Developing Applications with ArcInfo™: An Overview of ArcObjects™

An ESRI® Technical Paper • April 2000

ESRI 380 New York St., Redlands, CA 92373-8100, USA • TEL 909-793-2853 • FAX 909-793-5953 • E-MAIL info@esri.com • WEB www.esri.com

J-8462

Developing Applications with ArcInfo: An Overview of ArcObjects

An ESRI Technical Paper

Contents **Page**

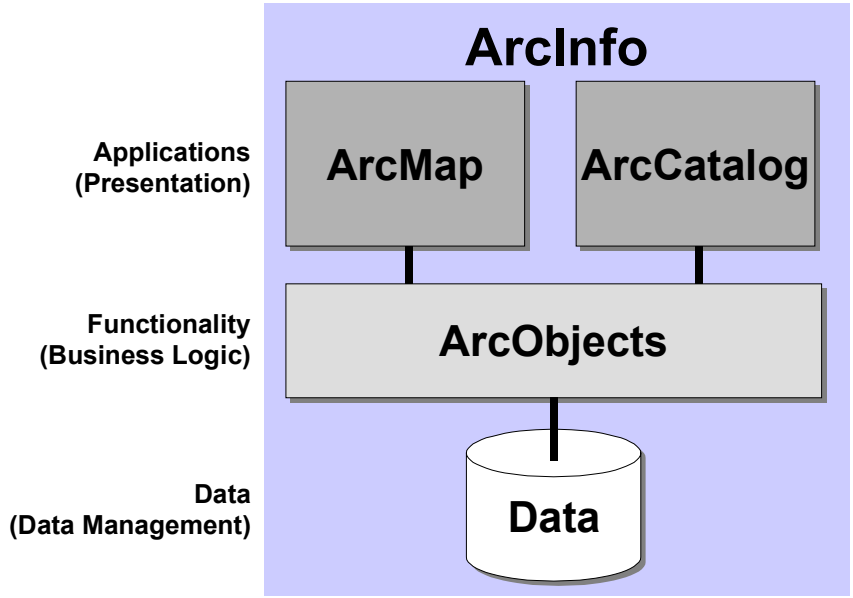
J-8462

Developing Applications with ArcInfo: An Overview of ArcObjects

Introduction

ArcObjects™ is the technology framework that forms the foundation of ArcInfo™ 8 software (Figure 1). Developers can use the ArcObjects framework to enhance and extend ArcInfo programmatically. With ArcObjects, developers can, for example, add new tools or work flows to the ArcInfo, ArcMap™, and ArcCatalog™ applications, or extend the ArcInfo data model into new application domains by adding new custom feature types. These are only two examples of the many ways in which developers can build on, embed, or extend ArcInfo ArcObjects.

Figure 1
Simple View of ArcInfo 8 Showing the Role of ArcObjects



ArcObjects is not sold separately; rather, it is included with ArcInfo software much like Avenue™ is an integral part of ArcView® GIS 3 software. To build applications using

ArcObjects, you must obtain a copy of ArcInfo. In addition, any derivative applications require a fully licensed ArcInfo seat.

This paper provides an introduction to ArcObjects, focusing on the general characteristics and examples of applications that can be created with ArcObjects.

When to Use ArcObjects

ArcInfo 8 includes three levels of customization, two of which utilize ArcObjects directly.

The first and simplest level of ArcInfo customization involves no programming and, therefore, no direct use of ArcObjects. All users can easily change the look and feel of the ArcInfo applications using standard user interface capabilities. For example, toolbars can be turned on and off using the customize dialog. Many other application properties can be changed using simple menu-driven actions like this.

The second level involves using the in-built Visual Basic® for Applications (VBA) application scripting capabilities to add new menus, tools, and work flows to the ArcInfo applications. The ArcMap and ArcCatalog applications include a fully working VBA development environment, including an integrated debugger, that provides access to all the ArcObjects technology. VBA allows the addition of modules, class modules, and user forms to ArcInfo project files. With VBA, it is possible to create extensive and sophisticated applications based on ArcObjects that run within the ArcMap and ArcCatalog application frameworks. It is not, however, possible to create new custom feature classes or build applications that run outside ArcMap and ArcCatalog. VBA is a very good choice for small-to-medium-size applications that use or extend the existing ArcInfo applications or functions. VBA is useful for many of the same types of tasks that ARC Macro Language (AML™) has been used for in the past.

Serious software developers who want to create reusable software building blocks, new applications, or custom feature additions to the geodatabase object model will prefer to work with ArcObjects directly using Visual Basic, or—especially in the latter case—Microsoft® Visual C++® or Delphi®.

What Is ArcObjects?

ArcObjects comprises a technology framework, an object-oriented geographic data model, an integrated library of software components, and a rich collection of developer-oriented resources (for example, a printed developer guide, online developer help, and a series of class diagrams). The ArcObjects software components are delivered as an organized collection of object–components. The ArcInfo object model describes the scope and organization of the components. Object–Components offer a number of key advantages over earlier software development methodologies.

Early, first-generation object-oriented GIS technology was built using C/C++ and suffered from several limitations, notably difficulty of sharing parts of the system (it is very difficult to share binary C++ components; most attempts have only shared source code), problems of persistence and updating C++ components without recompiling, lack of good modeling languages and tools, and proprietary user interfaces and customization tools. To counteract these and other problems, many software engineers have adopted component-based approaches to system development. A software component is a binary

unit of reusable code. The key to the success of components is that they implement in a very practical way many of the object-oriented principles now commonly accepted in software engineering. Components encapsulate behind a well-defined interface the manipulation methods and the data that characterize each instantiated object. This promotes structured and safe system development. Components facilitate software reuse because they are self-contained building blocks that can easily be assembled into larger systems. They also support inheritance and polymorphism. Inheritance is the ability to reuse functionality in other components by including reference to another object's state and behavior. For example, a new type of water valve can easily be created by overwriting or adding a few properties or methods to an existing type of valve that is similar. Polymorphism describes the process whereby each object has its specific implementation for operations such as draw, create, and delete. One example of the benefit of polymorphism is that a GIS can have a generic object creation component that issues generic object create requests to be processed in a specific way by each type of object class factory (e.g., gas pipes, valves, and service lines).

The object–component approach to software development provides a framework for anyone to extend a data model or customize a geographic information system. In the basic object model approach, only the original GIS vendor has complete customization capabilities and is free from performance and functionality bottlenecks. Also, because of the closed nature of object implementations, users are bound to proprietary macro languages when they want to undertake customization. With the ArcObjects object–component model, users extend the data model with exactly the same technology as ESRI used to build ArcInfo. As a consequence, users have more options and their objects will perform just as well. To the user, there is absolutely no difference between the ESRI base object–components and third party developer custom components.

Although there are several different object–component standards (COM, JavaBean, and CORBA), Microsoft's Component Object Model (COM) has emerged as the most viable technology for developing high-performance, interactive desktop and client/server applications. COM is well-defined and mature, is understood reasonably well, is prevalent in the developer community, and has excellent supporting materials (books, training resources, Integrated Development Environments [IDE], and modeling languages [e.g., UML, the Unified Modeling Language]). Because COM is a binary specification, it delivers good performance and is secure. ArcObjects is based on the COM specification.

What Is COM?

COM is a protocol that connects one software component or module with another. By making use of this protocol, it is possible to build reusable software components that can be dynamically interchanged in a distributed system. There are many terms in circulation that refer in part to COM; OleDB, ActiveX®, and DirectX® are all technologies based on the COM specification.

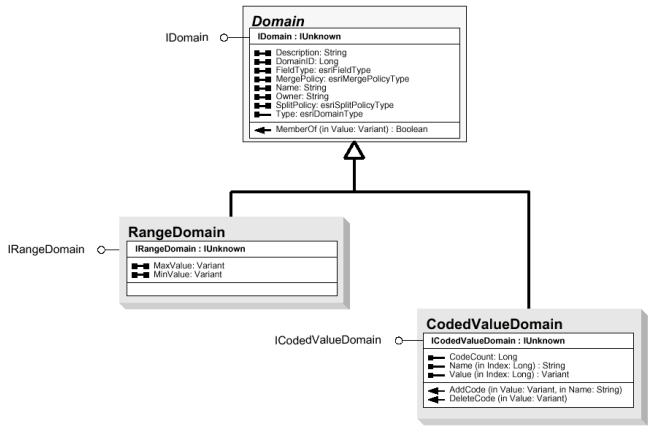
COM defines a binary specification that is not reliant on any specific language. Any language capable of manipulating computer memory can implement the COM specification. Since the COM specification deals with communication between components within a system, there are no restrictions placed on how the components are developed or what language is used to develop them.

COM allows for components to be reused at a binary level, meaning that third party developers do not require access to source code, header files, or object libraries in order to extend the system even at the lowest level.

Interface-Based Programming

Developing with COM means developing using interfaces. All communication between COM components is made via the components' interfaces. COM interfaces are abstract, meaning there is no implementation associated with an interface, and the code associated with an interface comes from a class implementation. The interface sets out what requests can be made of an object. COM interfaces are how COM objects communicate with each other. When working with COM objects, the developer never has ahold of the COM object directly but holds a reference to it via one of its interfaces. COM interfaces are designed to be a grouping of logically related functions. The virtual functions are used in abstract by the client and implemented by the server. Figure 2 shows examples of ArcObjects.

Figure 2
The Interfaces, Properties, and Methods of the Domain, RangeDomain, and CodedValueDomain ArcObjects



Every COM object has a default interface that is returned when the object is created if no other interface is specified. With some implementations of COM, a class may choose to implement the IDispatch interface, a so-called dual interface. Object classes that do not implement the IDispatch interface cannot be used with scripting languages, such as Java Script and VB Script, since to work they require all COM servers accessed to support the IDispatch interface.

Extensibility

When an interface has been published, it is not possible to change the external signature of that interface. It is, however, possible at any time to change the implementation details of an interface. This change may be a minor bug fix or a complete reworking of the underlying algorithm; the clients of the interface do not care, because the interface appears the same to them. This means that when upgrades to the servers are deployed in the form of new DLLs and EXEs, existing clients do not need to be recompiled to make

J-8462

use of the new functionality. If the external signature of the interface must be changed, a new interface is created to hold the new functions. Thus, when ESRI releases updates to ArcInfo, developers can be assured that the existing interfaces will always be supported, even if the internal class code changes.

A developer extends a COM-based system by creating new COM servers that serve up new or improved functionality. These new servers can be created from scratch, or they can use COM containment or COM aggregation to reuse existing functionality. For a third party developer to make use of existing components using either containment or aggregation, the only requirement is that the component that is being contained or aggregated is installed on both the developer and target client machines.

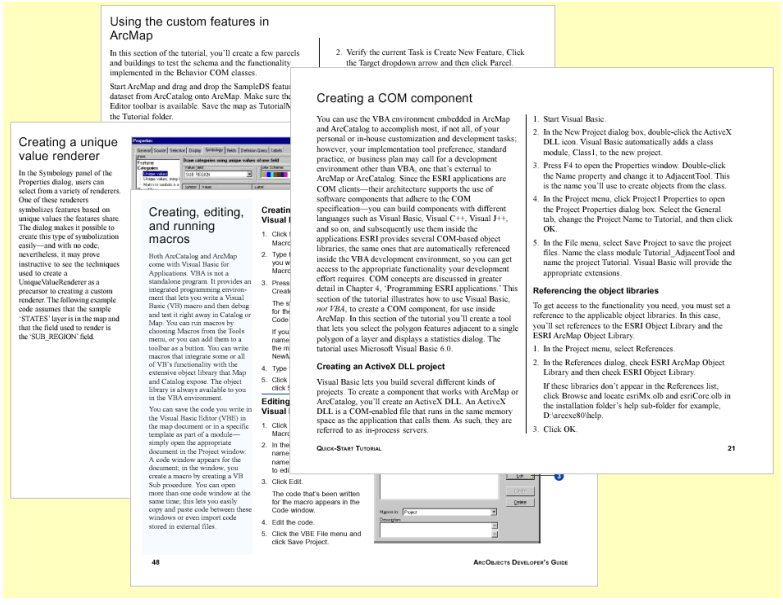
ArcObjects Developer Resources

There are a number of resources available to the developer that help with the learning process as well as act as an invaluable reference for experienced ArcObjects developers.

ArcObjects Developer's Guide

The *ArcObjects Developer's Guide* is for anyone who wants to learn the basic techniques and principles of customizing or adding functionality to the ArcInfo end user applications by working graphically with the user interface or by referencing the ArcObjects object model in a development environment. This book assumes you have some familiarity with the end-user applications, ArcMap and ArcCatalog, and it would be helpful but not essential to have had some experience designing and implementing applications or additional functionality for another ESRI® software program or other Microsoft Windows-based program. This book can be viewed as the starting point for all developers working with ArcObjects.

Figure 3
Extract from the *ArcObjects Developer's Guide*



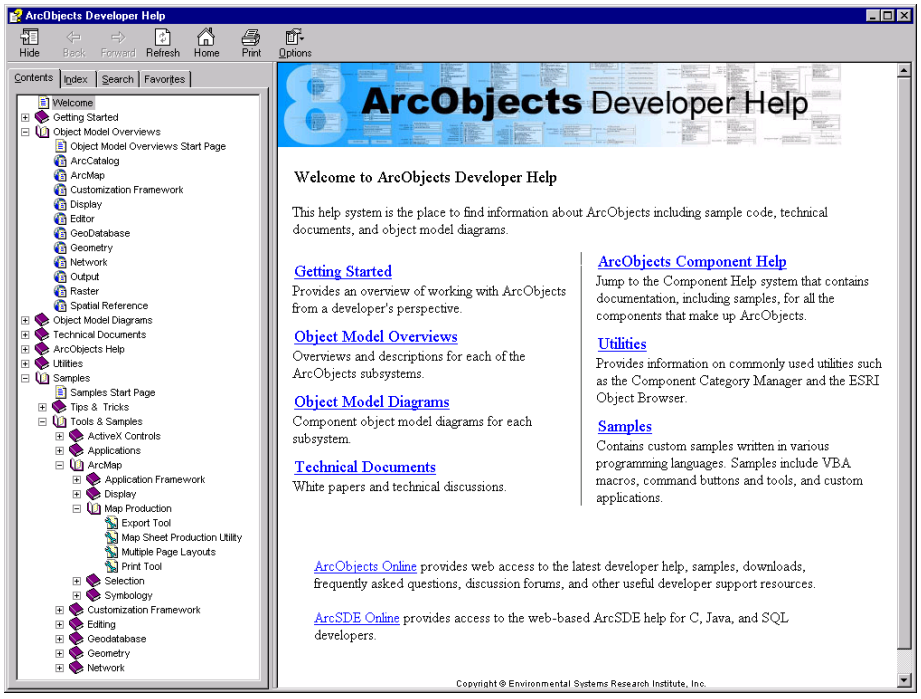
ArcObjects Developer Help

The ArcObjects Developer Help provides the developer with a compiled HTML file containing reference information, coding tips and tricks, and a number of complete code samples written in both Visual Basic and Visual C++. The help system viewer is shown in Figure 4.

The ArcObjects Developer Help is split into several sections. The first section is Getting Started. This section covers the basics of COM technology and, in greater detail, covers working with VBA and Visual Basic.

The next two sections cover the object models. These models document the many classes that make up ArcObjects and the relationships between these classes. In addition to the actual model diagrams, each diagram has a description of what components can be found on the diagram, and many of the common tasks performed on these objects are shown with examples. In this way, the developer is led through the process of working with these diagrams.

Figure 4
Extract from the ArcObjects Developer Help



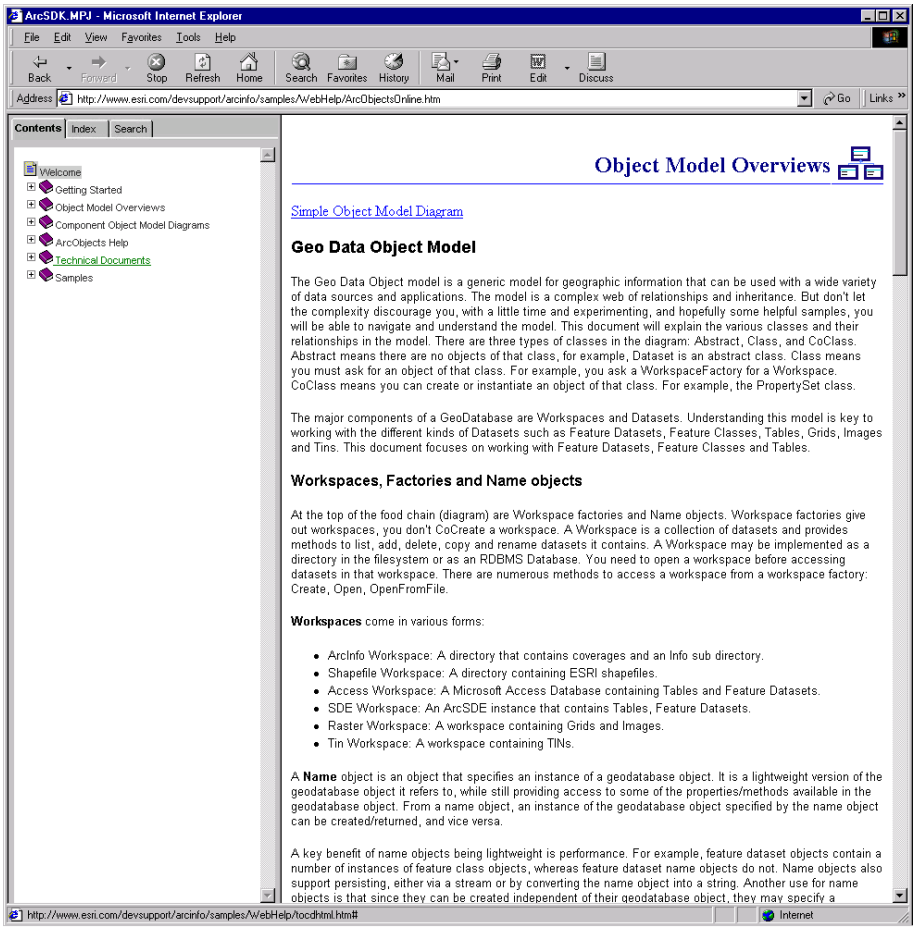
Some utility applications and technical papers are included in the next section, but it is the last section where developers will gain the majority of information. Samples included with ArcObjects Developer Help contain more than fifty tips and tricks and 100 full code examples.

J-8462

ArcObjects Online

The online version of the help system is accessed via a link from the top level page of the ArcObjects Developer Help system. This online help resource has the same structure as the help system supplied on the CD, but the content is updated on a regular basis.

Figure 5
Extract from ArcObjects Online Developer Help

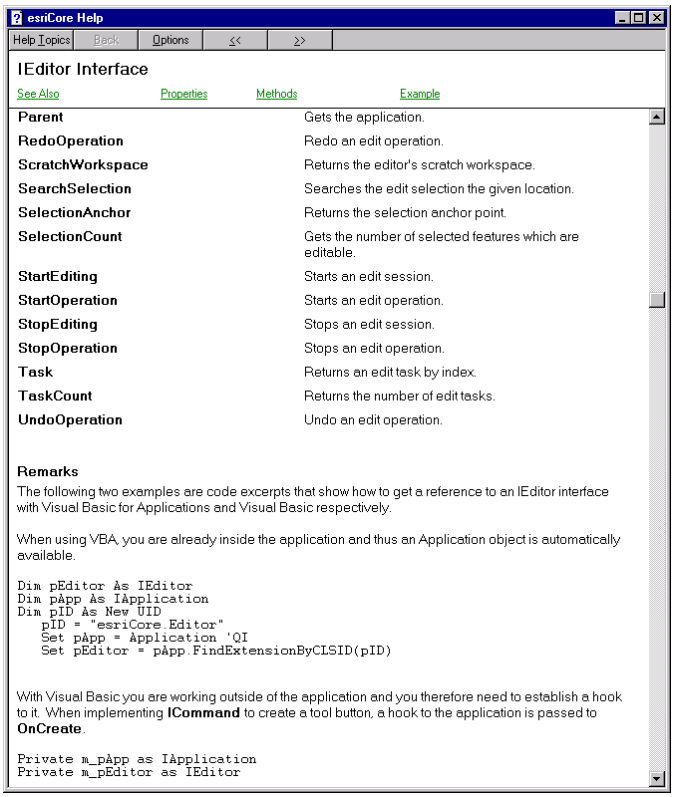


These updates include the addition of more tips and tricks and tools and samples, as well as help "hot topics." If there are areas of functionality that after feedback from users require more detailed discussion, the ArcObjects Online Developer Help system is the place where these enhancements will be posted.

ArcObjects Component Help

All the resources mentioned so far provide the developer with information and guidance when working with ArcObjects as a whole. For detailed reference information of the methods and properties of the classes that compose ArcObjects, these are two main resources.

Figure 6
Example Help for IEditor Object–Component Interface



The definitive resource is the ArcObjects Component Help system. This documents all the ArcObjects, with the details required by developers to work with the individual functions. This help system is accessed via ArcObjects Developer Help. In addition to detailed descriptions of classes and their interfaces, with the functions and arguments, many of the more common classes have detailed example code snippets to show the developer how to work with the component.

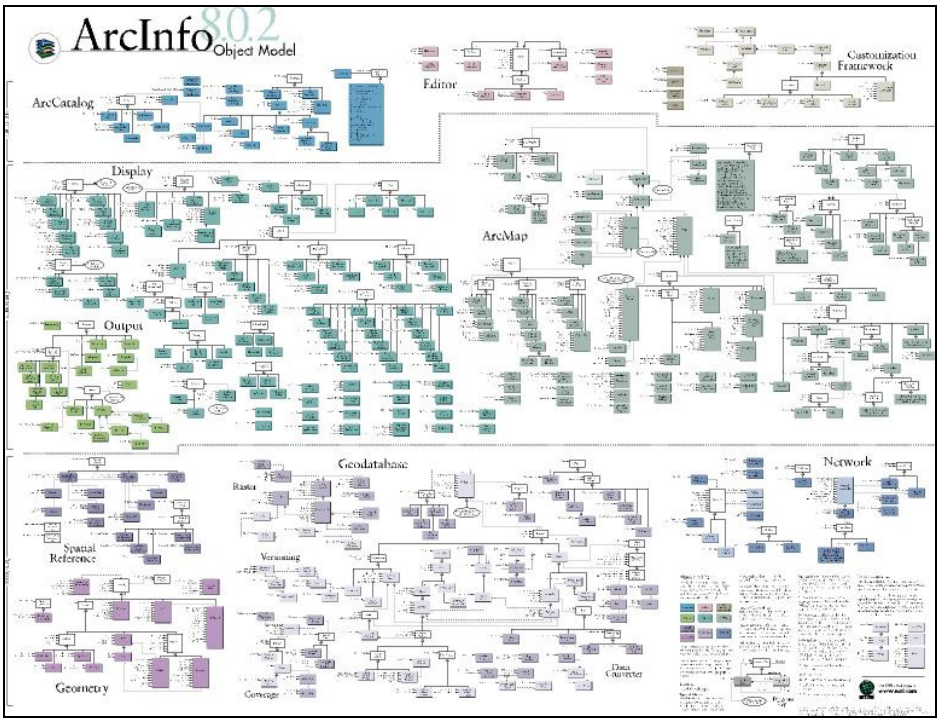
Class Diagrams

For the developer starting work with ArcObjects, the class diagrams are perhaps the resource that is most useful. Not only do they document many of the classes and interfaces, but more importantly they show clearly the relationships that individual classes have with each other. These relationships are not shown in any other form of documentation.

The class diagrams divide the ArcObjects object–components into distinct functionality groups. For instance, there are diagrams depicting the object–components used by the ArcMap and ArcCatalog applications, along with diagrams for the geometry, display, output, and geodatabase subsystems, to name a few. The ArcInfo Object Model (Figure 7) provides an overall picture of the class diagrams. More detailed diagrams are available for each functionality area.

J-8462

Figure 7
The ArcInfo Object Model, the Highest Level Overview of ArcObjects



Other Resources

There are many other resources available to the developer. These resources range from object browsers that allow the object model to be viewed in a form familiar to the developer (for example, the object browser that ships with Visual Basic or OLEView, a small utility application available from the Microsoft Web site), to books on COM and Windows application development. Many of the popular publishing houses have Web sites that support their books with additional material and many source code examples in a variety of languages.

What Is Possible with ArcObjects?

ArcObjects offers the developer four main opportunities: extend existing ESRI applications (ArcMap and ArcCatalog); extend the data model used by applications based on ArcObjects; embed ArcObjects within an existing application; and create entirely new applications. What follows are some examples that illustrate these different opportunities.

Extending ArcMap and ArcCatalog

These two new ArcInfo software applications provide the developer with an application framework that can be extended with little effort. Extensions can range from simple commands and tools to complete industry-focused solutions comprising customized tools, renderers, and user interfaces.

Commands and Tools

Commands and tools are most commonly accessed via pushbuttons on toolbars. A very simple command that displays a dialog box with the number of selected objects would

have code like this associated with the OnClick method. This method is called when the user presses the pushbutton associated with the command.

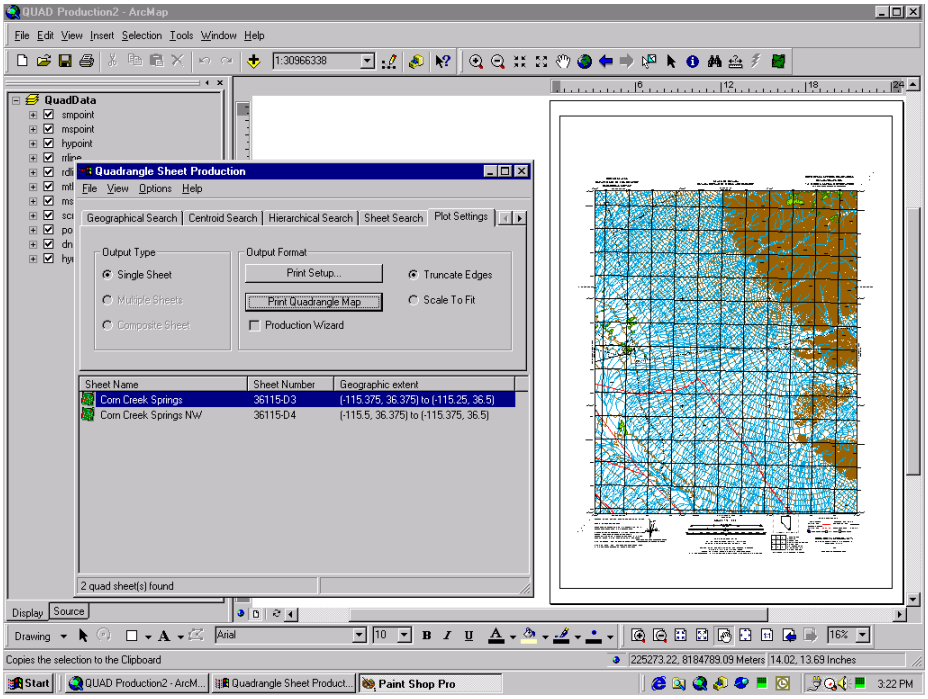
```
  ' Declare an ImxDocument variable

        Dim pMxDoc As ImxDocument

        ' QI the Idocument for the IMxDocument

  Set pMxDoc = m_pApp.Document


'Access the property SelectionCount held on the Imap interface

  MsgBox "Number of selected objects = " & pMxDoc.FocusMap.SelectionCount
```

The document object of class has an interface, ImxDocument, that has a property that refers to the current map that the user is referencing; this in turn has a property that holds the number of selected features. The value of this property is displayed to the user using the message box dialog that is supplied by Visual Basic.

The selection count example is one of the simplest commands possible, but that does not mean there is a limitation to the complexity these commands can have. Figure 8 shows an example of a more complex command that has been developed for the ArcMap application. This command, when activated, displays a dialog that supports the automatic generation of USGS quadrangle sheets.

J-8462

Figure 8
Screen Shot Showing an Example of a New Command Added to ArcMap



In this example, a new Form Menu (Quadrangle Sheet Production) has been added to ArcMap along with some new functionality.

This command not only supports the selection of the appropriate sheet, but it also controls the generation of all the marginalia required for the sheet, in addition to the printing of the sheet on a suitable device.

Both the above example commands have one thing in common: they implement the ICommand interface. The application framework recognizes this and is able to interact with the command and, in this way, the command is plugged into the application with little effort. When interfaces were covered in the COM section, it was stressed that the interface acts as a contract between client (ArcMap) and server (command). The ICommand interface has the properties and methods required by the application framework in order for the command to be integrated within ArcMap. In Visual Basic, it is a simple matter of using the Implements keyword to specify the interface to implement. In order to fulfill the contract between client and server, the developer must then implement all the methods and properties for that interface.

Below is an example of a command that zooms the display by a factor of two every time the command is selected. As can be seen, many of the interface's properties are simple to implement. The only function that has any amount of code is the OnClick. This function grabs the envelope defining the extents of the current map view and then halves these extents and redraws the screen.

J-8462

```
Implements ICommand

'Member variable holding the ArcMap application interface
Dim m_pApp As IApplication

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
End Property

Private Property Get ICommand_Caption() As String
  'Text label displayed on the command button
  ICommand_Caption = "Zoom In"
End Property

Private Property Get ICommand_Category() As String
  ' Category of command - used to order command in customize dialog
  ICommand_Category = "Sample Commands"
End Property

Private Property Get ICommand_Checked() As Boolean
  ' Does the command have a check mark next to it - usual for menu entrys
  ICommand_Checked = False
End Property

Private Property Get ICommand_Enabled() As Boolean
  'Enable our command button
  ICommand_Enabled = True
End Property

Private Property Get ICommand_HelpContextID() As Long
End Property

Private Property Get ICommand_HelpFile() As String
End Property
```

J-8462

```
Private Property Get ICommand_Message() As String
  ' Message that appears on the Status Bar while crossing the tool with
the cursor
  ICommand_Message = "Zooms the display x2"
End Property

Private Property Get ICommand_Name() As String
  ' Unique name given to the command - makes finding it in VBA easy with
the ArcID
  ICommand_Name = "Our Tools_Zoom In"
End Property

Private Sub ICommand_OnClick()
  ' Method called when user clicks button
  Dim pMXDoc As IMxDocument

  ' QI for ImxDocument interface from IDocument
  Set pMXDoc = m_pApp.Document

  ' Get the Active view extents envelope
  Dim pEnv As IEnvelope
  Set pEnv = pMXDoc.ActiveView.Extent
  ' Shrink the envelope by half
  pEnv.Expand 0.5, 0.5, True

  'Set the activeview to this new envelope extent
  pMXDoc.ActiveView.Extent = pEnv
  ' Redraw the map display to reflect the new extents
  pMXDoc.ActiveView.Refresh
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  'Hold onto the application for use in subsequent calls to my dll
```

```
  Set m_pApp = hook
End Sub

Private Property Get ICommand_Tooltip() As String
  ' Tooltip text for bubble help
  ICommand_Tooltip = "Zoom In x2"
End Property
```

The code associated with the quadrangle sheet production command is similar for many of the methods that support the tool tips, descriptions, help, and so forth. The main difference is the OnClick method. In the case of the quadrangle sheet command, this displays the dialog developed in Visual Basic. Subsequent user interaction is with this dialog.

To support different types of interaction with the application, the developer must implement other types of interfaces. All these interfaces are explained in the *ArcObjects Developer's Guide*, and there are numerous examples within the help system.

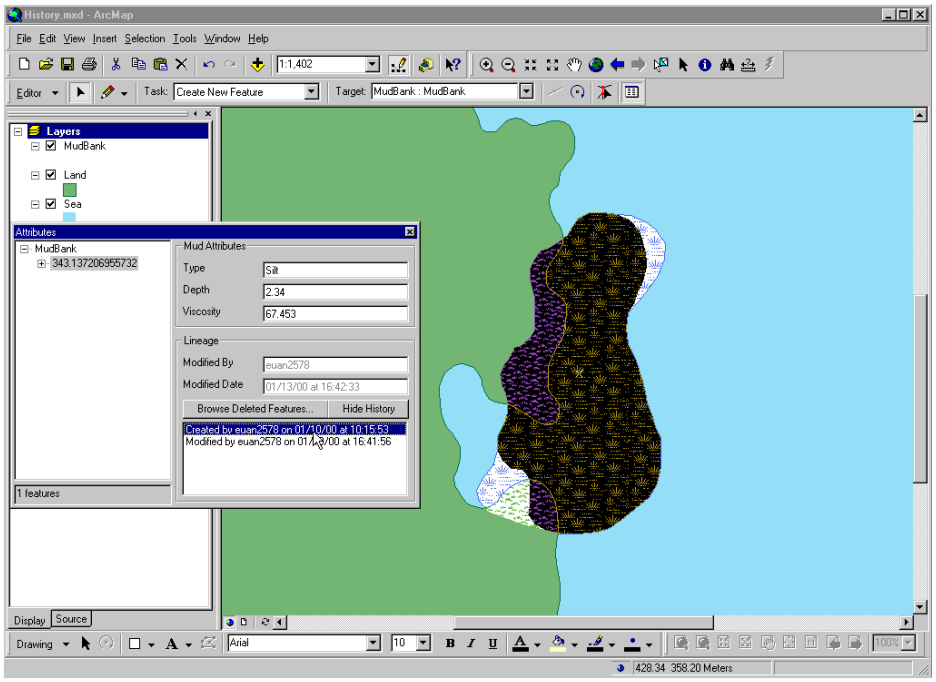
Extending the Data Model

The ArcInfo data model can be extended in one of two ways. Specific behavior relevant to individual object classes of the data model can be modeled within the system framework using custom features, relationships, and so forth. More generic support for new data types can also be added by extending the data formats supported by the geodatabase.

Figure 9 illustrates a custom feature class that implements a drawing method. This drawing method ensures that the Mud Bank polygon feature draws itself differently depending on what type of features lies beneath it. If the Mud Bank feature is on land, a swamp symbol is used, whereas if it is on water, the symbol changes to marsh. This decision is taken every time the symbol is drawn, meaning that changes to any of the related features automatically update the representation. Figure 9 also illustrates a custom property inspector. This property inspector has a history list that enables the object's history to be browsed. When a list entry is selected, the geometry state at that time in history is displayed in black in the map window, and the geometry when the object was created is clearly seen.

J-8462

Figure 9
A Screen Shot Showing an ArcObjects Application Example That Implements a Data Model Extension



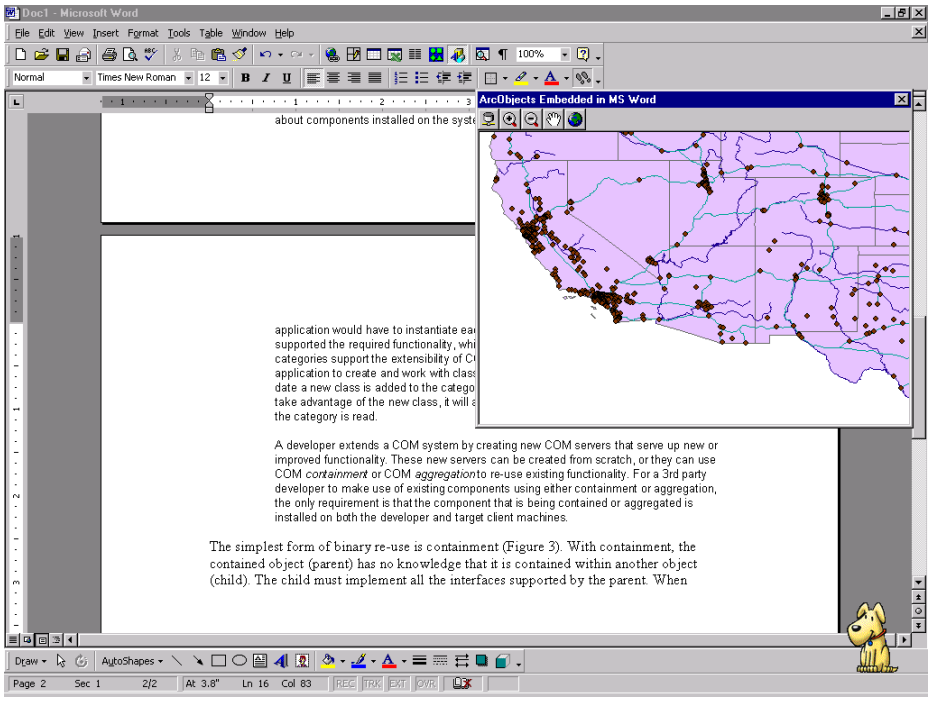
The new feature type "Mud Bank" is rendered based on logic that extends the standard polygon draw method.

Embedding Within an Application

Since ArcObjects is based on COM, any application that is capable of integrating COM components can embed ArcObjects. There is no limitation to what can be done with ArcObjects when used in this context compared to working with ArcObjects in the context of ArcMap and ArcCatalog. Figure 10 illustrates embedding ArcObjects inside Microsoft Word to provide a simple map viewer within Word documents.

Like ArcMap, Microsoft Word comes with VBA as its native development environment, meaning that the developer will have a similar experience working with ArcObjects inside Microsoft Word to that of working with the objects within ArcMap.

Figure 10
A Map Viewer Embedded Within a Microsoft Word Document

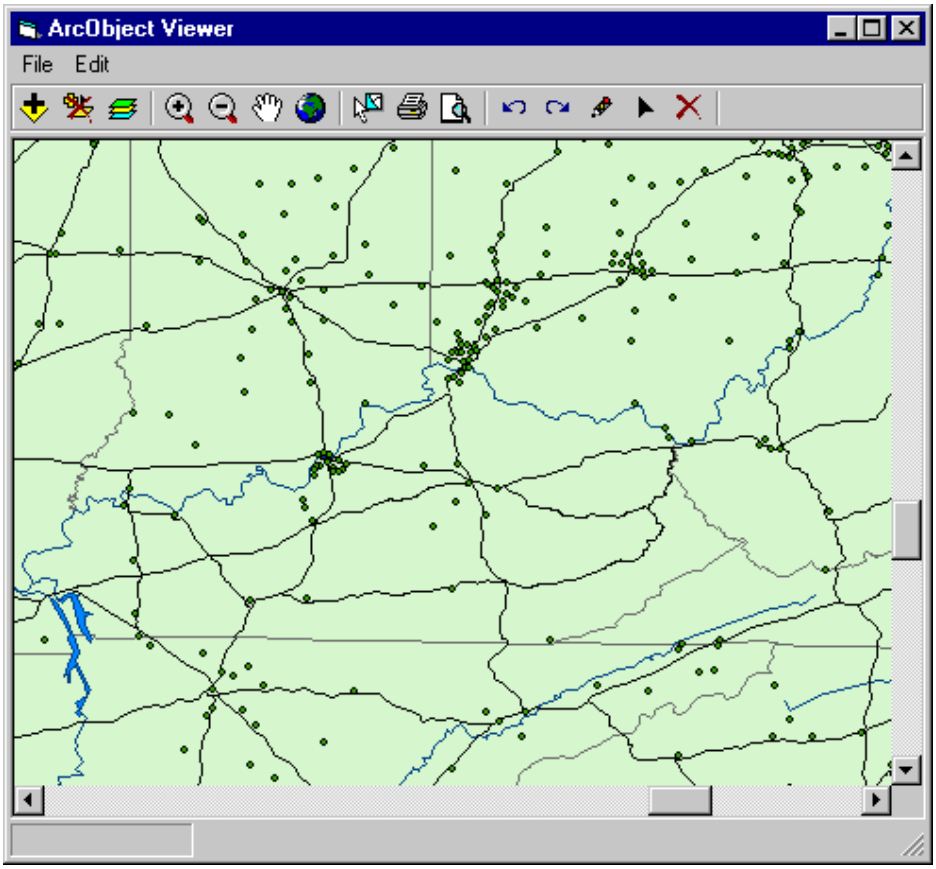


Creating New Applications

The previous examples have focused on extending the existing application frameworks. It is also possible to create completely new applications that embed ArcObjects. Here the developer is responsible for all aspects of the application framework, graphic user interface (GUI), program control, and so on. Figure 11 is an example of a stand-alone application created with ArcObjects and Visual Basic. This simple map viewer uses ArcObjects to draw map layers, perform geographic and attribute queries, and edit and print maps.

J-8462

Figure 11
A Screen Shot of a Completely New Stand-Alone Application Built with Visual Basic and ArcObjects



Conclusion: Where Next?

The goal of this paper is to give an overview of the ArcObjects technology and to illustrate the type of projects that can be undertaken. Hopefully you have gained an understanding of the openness of ArcObjects and the tremendous wealth of GIS functionality it exposes to developers.

To begin using ArcObjects, you will need to obtain a copy of ArcInfo software. More details about ArcObjects can be found in the ArcInfo documentation; especially valuable are the developer samples. Before starting on an ArcObjects project, it is advisable to attend an ESRI developer training class.

J-8462