Chapter 1
Introduction

Welcome to CA-Visual Objects!

CA-Visual Objects is a fully object-oriented application development system that allows you to quickly and easily create sophisticated applications that run under Microsoft Windows. Its entry into the world of application development offers new opportunities and technology to programmers of all levels and backgrounds.

With CA-Visual Objects, you can create full-featured, "thoroughbred" applications that deliver everything Windows users have come to expect, including:

- Multiple document interfaces (MDI), with no constraints on simultaneously opening several documents (such as databases or text files) or the same document in several different windows
- Event-driven operation, with no limitations on user flexibility and control
- Top-flight graphical appearance and full-fledged annotation, prompting, and help
- Support for Windows conventions and subsystems, such as the Clipboard, drag-and-drop, and Help

All of this is achieved by bringing together the worlds of object-oriented programming (OOP), graphical user interfaces (GUIs), visual design tools, and traditional business languages—all in a single, integrated desktop.

CA-Visual Objects Features

CA-Visual Objects has lots of exciting features designed to make the application development process fast and easy, as well as to free you from dealing with technical details that can hinder you from using your time productively.

Visual Programming Tools and a Complete IDE

The integrated development environment (IDE) provides tools that enable you to *visually* design the windows, menus, reports, icons, etc., for your applications using point-and-click, drag-and-drop techniques. These tools let you see the result of your design as it progresses.

The IDE is an intuitive and powerful environment for creating applications—for example, the most frequently used features of the IDE (like Save, Build, and Execute) are available at the touch of a button. This allows new users to start developing applications quickly.

The IDE offers a sophisticated and powerful environment for the advanced developer, with features that allow you to spend more time on the business logic aspect of application development. For example, CA-Visual Objects automatically tracks and maintains the relationships between the various pieces of an application for you, determining which components need to be compiled to build an application. Things like make files and compiler and linker script files are, therefore, obsolete.

Additionally, the IDE offers the capability of running applications without generating an executable file (.EXE). This feature enables fast prototyping and quick feedback when you make changes to the application and enables you to test and debug your applications efficiently using the debugger in the IDE.

After developing, testing, and debugging your application, distributing it as a stand-alone .EXE is easy. You simply click a button to generate an .EXE, which can be distributed royalty-free to your end users.

A Fully Object-Oriented Language

The CA-Visual Objects language is fully object-oriented. Some may question: Why object-orientation? There are many reasons, the most fundamental of which is that programming for event-driven, GUI environments presents a set of challenges that are aptly met by OOP.

As you read through the CA-Visual Objects documentation (in particular, this guide and the *Programmer's Guides*), you will see how OOP naturally lends itself to GUI environments by giving you the capability to develop complex systems through standard, reusable components, in a manner that models the real world.

To facilitate object-oriented programming, CA-Visual Objects includes extensive *class libraries* for:

- GUI programming
- Database management
- Reporting

These libraries provide very powerful building blocks for your applications. In addition, the visual tools in the IDE exploit the strengths of object-orientation by using these class libraries to generate object-oriented code based on your designs.

Note: Class libraries are no different from other libraries you would use in your applications—instead of containing functions, for example, they contain class and method definitions.

The language also features a structured superset of the Xbase language. (*Xbase* is the industry standard term for those programming languages that inherit from the original dBASE system, including CA-Clipper, CA-dBFast, dBASE III PLUS, dBASE IV, and FoxPro.)

The Xbase superset contains extensions for Windows and its environment, including the ability to access all Windows Application Programming Interface (API) functions for low-level, system programming.

Open Database Access

CA-Visual Objects gives you a wide variety of choices in terms of database access. It supports:

- Both procedural and object-oriented access to Xbase databases

  CA-Visual Objects supports the procedural database commands and functions—like SKIP and EOF()—that are traditional to Xbase languages.

  However, it also includes an object-oriented interface to Xbase database management. Both semantically and syntactically, the object-oriented interface is akin to the commands and functions traditionally used in procedural access. For example, instead of commands like APPEND, COMMIT, and ZAP, you'll use methods named Append(), Commit(), and Zap() to perform the same operation.

  Note: With these new methods, all the capabilities of the traditional Xbase approach are provided, but have been enhanced to fit the event-driven, multi-tasking nature of GUI applications.

- Access to both Xbase and SQL databases

  When using an object-oriented approach to database management, both Xbase and SQL databases can be accessed.

Furthermore, access to these two different types of databases is accomplished using a single, compatible protocol. This allows an application to manage Xbase and SQL databases with the same code.

- Several different Xbase/SQL database formats

  When accessing Xbase databases (using either a procedural or object-oriented approach), you can choose from a variety of file formats. This is accomplished through replaceable database driver (RDD) technology. With RDDs, a single application can access different database file formats using a common language interface. This allows you to tailor your applications so that migrating from one database format to another is simple and straightforward.

  CA-Visual Objects supplies several popular RDDs, and through its open architecture allows for development of third-party RDDs. See the Replaceable Database Drivers section in the "Using DBF Files" chapter in the *Programmer's Guide, Volume II* for more information about RDD technology. Also refer to the "RDD Specifics" appendix in the same volume for detailed information about specific RDDs.

  Similarly, support for SQL databases is accomplished using Open Database Connectivity (ODBC), a widely used API for SQL access under Windows. This technology also uses replaceable drivers, supplied as dynamic link libraries (DLLs), which standardize the interface to the various database formats. CA-Visual Objects comes bundled with DLLs for many of the popular ODBC formats, and provides language support for a superset of the standard ODBC API, as well as an object-oriented interface compatible with that used for Xbase database files.

An Active Repository

CA-Visual Objects is a repository-based system. The *repository* is where the IDE stores all application components, and it automatically manages the relationships between the various components of an application. For example, if you make a change to a library component, the repository automatically marks every application with that library in its search path, indicating that it should be rebuilt.

A Native Code, Incremental Compiler

CA-Visual Objects can compile your applications down to native machine code. This gives you the flexibility of using OOP without sacrificing runtime performance.

Also, to support iterative development, the compiler works with entity-level granularity. *Entities*, as explained in greater detail later in this guide, are the smallest pieces of an application (like a function or a global variable declaration). *Entity-level granularity* means that when you make a change to an application and then build the application, the compiler determines which entities of the application have changed (or are affected by the change), and automatically recompiles only those pieces, as opposed to recompiling entire modules.

Entity-level granularity is a powerful feature because it speeds development—you spend less time waiting for your application to be built and more time designing, enhancing, and fine-tuning. Additionally, it makes prototyping fast and easy.

Reporting with CA-RET

CA-RET—the Computer Associates Report Engine Technology program component—has been integrated with CA-Visual Objects to provide powerful reporting capabilities for your applications.

CA-RET offers a sophisticated database publishing interface, allowing you to design and produce custom database reports at the press of a button. While in CA-RET, you can use its intuitive, GUI environment to define the structure and specifications of the report. For example, you can add fields, text, tables, and pictures to a report, and format the various sections (like headers, footers, and titles).

An Open Architecture

CA-Visual Objects features extensible subsystems that facilitate the integration of third-party tools within the product. It also supports a number of powerful features that allow your applications to interact with other applications and exchange data, as well as use routines written in other languages, such as C, C++, Pascal, and COBOL.

For example, you can:

- Access the functions stored in a DLL

  A DLL is a library of functions in which only the interface definitions are visible, not the source code. CA-Visual Objects has the necessary language support (such as pointers and structures) to access standard Windows DLLs, including the Windows API. Additionally, not only can you use DLLs in your applications, but you can *build* them in CA-Visual Objects. Furthermore, in addition to standard DLLs, you can build special CA-Visual Objects DLLs that can contain classes, methods, etc.

- Use Dynamic Data Exchange (DDE) to exchange information with other DDE-compatible applications (such as CA-RET)

- Interface with your system's Clipboard facility to transfer different types of data between applications

Where to Go from Here

In conclusion, CA-Visual Objects' strength and ease-of-use offer you great potential for developing GUI applications that run under Windows. Its intuitive, robust language and visual development tools simplify the complexities of GUI programming for every user, yet are powerful enough to satisfy all development needs.

To gain a better understanding of CA-Visual Objects features and capabilities, all users—both novice and advanced—should work through the remainder of this *Getting Started* guide.

After working through this guide, you may want to go on to the *IDE User Guide*, which provides more detailed information about the IDE, especially the browsers and visual editors, editing and debugging applications, and creating .EXEs.

To learn more about programming, read through the three volumes of the *Programmer's Guide*. Once you are more familiar with the language, refer to the *Class Reference*, *Function Reference*, and/or *Command Reference Guides* for complete information about syntax and parameters. (Of course, at any time, you can display the online Help reference for information about the environment and language right on your screen as you work.)

Tip: If you are a CA-Clipper user, you may want to consult *Migrating Your CA-Clipper Programs to CA-Visual Objects* for tips and techniques on how to migrate your existing CA-Clipper code.

In This Guide

This *Getting Started* guide is your introduction to CA-Visual Objects. It contains all the information you need to get a quick and productive start.

This guide is organized into the following chapters:

1. Introduction
2. Installing and Starting CA-Visual Objects

   Provides the information you need to install and start CA-Visual Objects.
3. Object-Oriented Programming Concepts

   Intended for a person new to object-oriented programming, this chapter describes the basic principles of OOP.
4. An Overview of the IDE

   Presents an overview of some of the features of the IDE.
5. Learning the Basics

   Provides a hands-on tutorial that you can work through to build a sample MDI application.

What You Need to Know

In addition to an understanding of basic programming concepts, this guide assumes that you are familiar with Windows terminology and navigational techniques, including how to work with standard Windows items like menus, dialog boxes, the Clipboard, and the Control Panel.

If you are unfamiliar with Windows, please refer to your Windows documentation before using CA-Visual Objects.

General Typographic Conventions

This guide also employs several typographic conventions (such as capitalization or italic formatting) to distinguish between language elements and discussion of them.

Key Names

The names of keys, such as Enter, Ctrl, and Del, appear in the document as they do on your keyboard, where possible.

Note that when referring to the four arrow keys as a group, they are referred to as Direction keys; however, the name of each Direction key (for example, Up arrow or Left arrow) is used when referring to them individually.

Key Combinations

Whenever two keys are joined together with a plus (+) sign (for example, Ctrl+R), you should hold down the first key while pressing the second key to complete the command. Release the second key first.

Key Sequences

When keys are separated by a comma (,), press them in the sequence indicated. The keystroke sequence Alt+E, C, for example, indicates that you should hold the Alt key down while pressing the E key, release them both, and then press and release the C key.

User Input Examples

The following conventions are used for user input:

- Literal information (text that the user must enter exactly as shown) is shown in bold:

  Insert the diskette into drive A and type **a:\install**.

- Placeholder text (variable information a user must enter) is denoted by a bold and italic typeface:

  Enter **login *username***.

UPPERCASE

The following appear in uppercase:

- Commands (like CLEAR MEMORY)
- Keywords (for example, AS, WORD, and INT)
- Reserved words (for example, NIL, TRUE, and FALSE)
- Constants (for example, NULL_STRING and MAX_ALLOC)

Mixed Case / Initial Capitalization

The following are displayed using mixed case:

- Function, method, and procedure names (like SetDoubleClickTime() and Abs())
- Class names (for example, TopAppWindow and DBServer)
- Variable names (for example, oTopAppWindow and nLoopCounter)

Italic

Variable names are displayed in italic in syntax (for example, Abs(<*nValue*>)) and when referring to them in the discussion text.

Cross References

The following conventions are used:

- Guide name in italic:

  See the *IDE User Guide.*

- Part name in single quotes:

  See ‘Database Programming’ in the *Programmer’s Guide, Volume II*.

- Chapter name in double quotes:

  See “Creating an Application” in the *IDE User Guide*.

- Section name as it appears in the document:

  Also see the Saving a Program section.

Getting Help

CA-Visual Objects provides online Help, which can be used to display information on your console as you work. You can use any of the following Help menu commands:

Menu Command	Description
Index	Displays an index of available help topics about the CA-Visual Objects language and IDE.
Context Help	Allows you to get context-sensitive help for an item or area currently displayed on your screen.
How to Use Help	Describes how to use the Windows online Help system.

In the IDE you can also receive context-sensitive help for a menu or menu command by pressing either the F1 key or the Shift+F1 key combination. Press Shift+F1 to receive context-sensitive help for most dialog boxes and windows.

Additionally, when the Source Code Editor is open, you can receive context-sensitive help for the keywords, commands, classes, and functions in a selected module or entity. Simply highlight the keyword, command, class, or function and press the Shift+F1 key combination.