ASCENT®
CENTER FOR TECHNICAL KNOWLEDGE

CATIA V5-6R2018
Visual Basic Automation

Learning Guide

1st Edition

ASCENT - Center for Technical Knowledge®
CATIA V5-6R2018
Visual Basic Automation
1st Edition

Prepared and produced by:

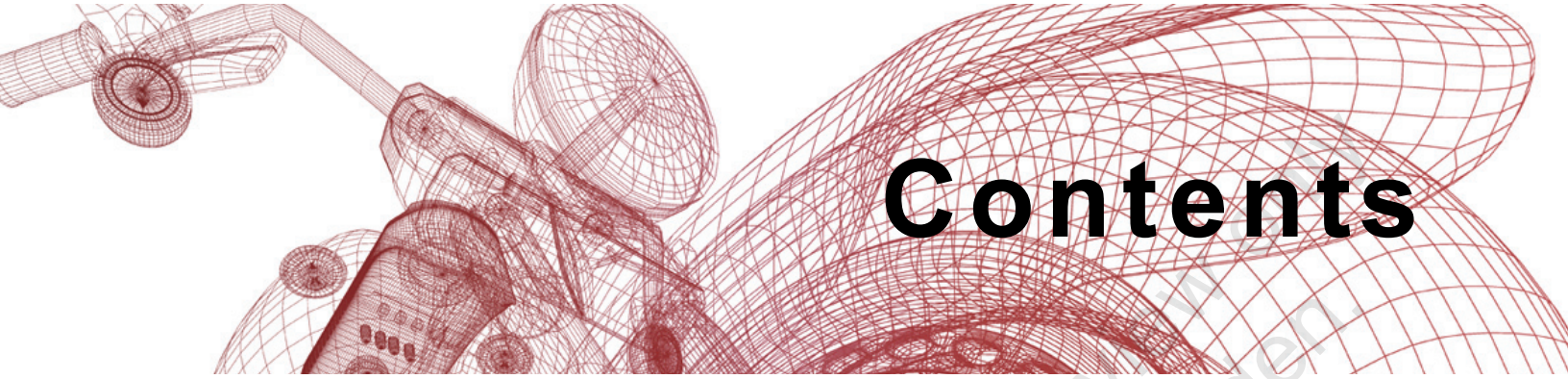
ASCENT Center for Technical Knowledge
630 Peter Jefferson Parkway, Suite 175
Charlottesville, VA 22911

866-527-2368
www.ASCENTed.com

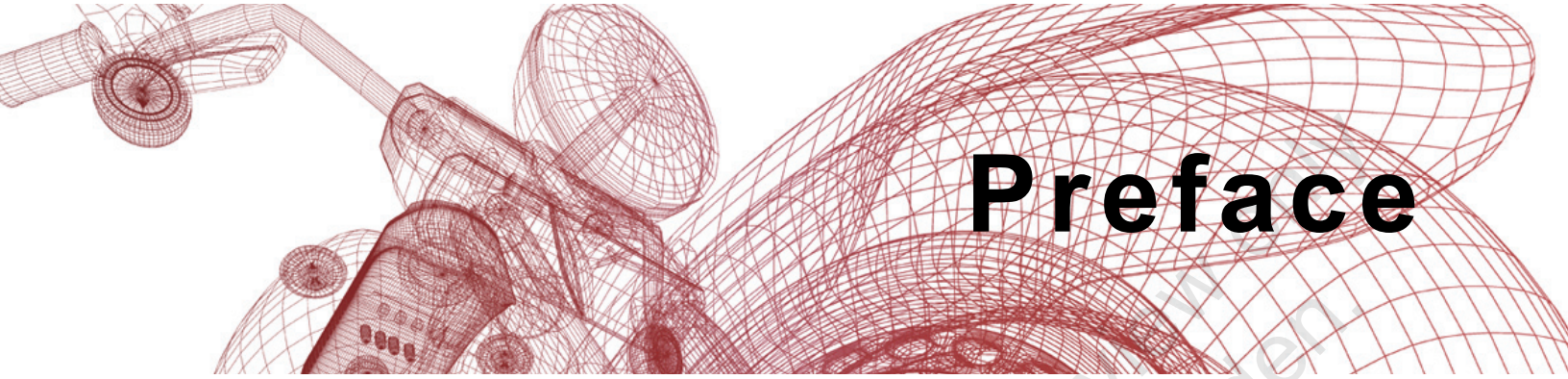
Lead Contributor: Scott Hendren

ASCENT - Center for Technical Knowledge is a division of Rand Worldwide, Inc., providing custom developed knowledge products and services for leading engineering software applications. ASCENT is focused on specializing in the creation of education programs that incorporate the best of classroom learning and technology-based training offerings.

We welcome any comments you may have regarding this guide, or any of our products. To contact us please email: feedback@ASCENTed.com.



Contents



Preface

The *CATIA V5-6R2018: Visual Basic Automation* learning guide provides you with a good understanding of the different ways to automate tasks using CATIA macros and Visual Basic programming. Using hands-on practices, you will use VB pro-gramming to work with parts, assemblies, drawings, selections, parameters and formulas, graphic properties, and to exchange data with Microsoft Excel.

Topics Covered:

- CATIA V5 Object Model
- Creating Part Design and Shape Design features
- Working with Product Structure and Assembly Design
- Scripting Drawing Views, Frames, and Title Blocks
- Deleting, Cutting, Copying, Pasting CATIA objects
- Interactive Selections
- Communication with MS Office

Note on Software Setup

This learning guide assumes a standard installation of the software using the default preferences during installation. Lectures and practices use the standard software templates and default options for the Content Libraries.

Lead Contributor: Scott Hendren

Scott Hendren has been a trainer and curriculum developer in the PLM industry for over 20 years, with experience on multiple CAD systems, including Pro/ENGINEER, Creo Parametric, and CATIA. Trained in Instructional Design, Scott uses his skills to develop instructor-led and web-based training products.

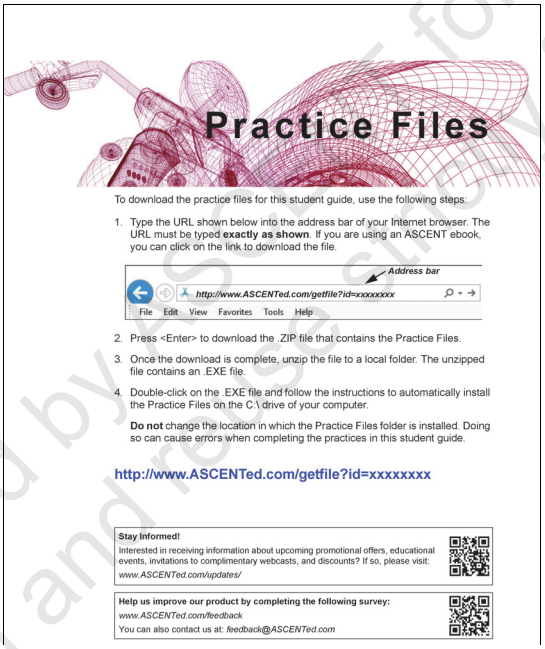
Scott has held training and development positions with several high profile PLM companies, and has been with the ASCENT team since 2013.

Scott holds a Bachelor of Mechanical Engineering Degree as well as a Bachelor of Science in Mathematics from Dalhousie University, Nova Scotia, Canada.

Scott Hendren has been the Lead Contributor for *CATIA: Visual Basic Automation* since 2013.

In this Guide

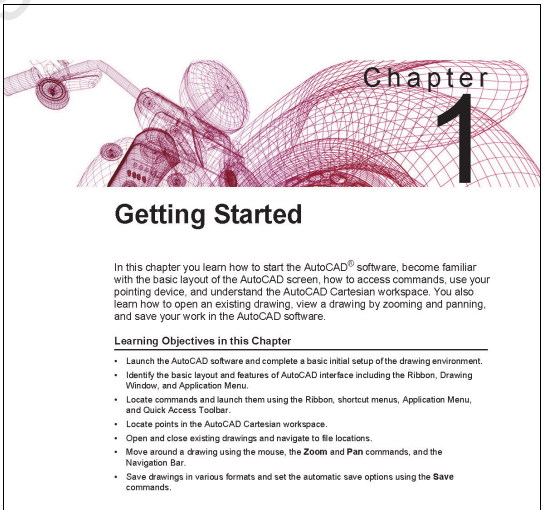
The following images highlight some of the features that can be found in this guide.



Link to the practice files

Practice Files

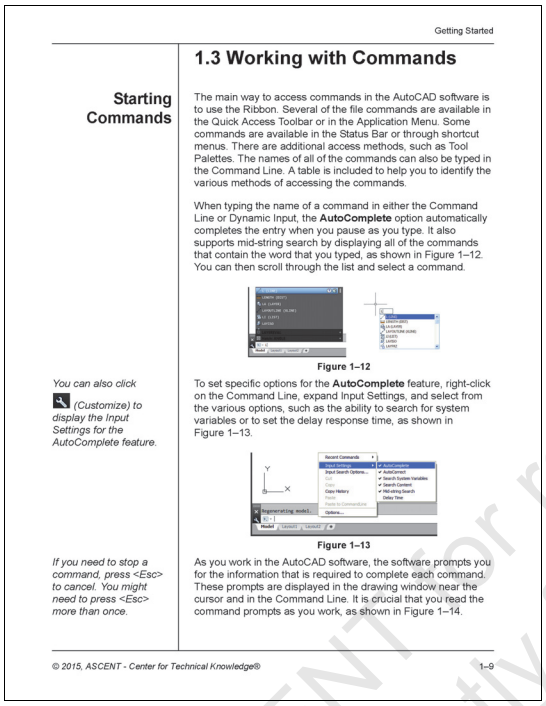
The Practice Files page tells you how to download and install the practice files that are provided with this guide.



Learning Objectives for the chapter

Chapters

Each chapter begins with a brief introduction and a list of the chapter's Learning Objectives.



Getting Started

1.3 Working with Commands

Starting Commands

The main way to access commands in the AutoCAD software is to use the Ribbon. Several of the file commands are available in the Quick Access Toolbar or in the Application Menu. Some commands are available in the Status Bar or through shortcut menus. There are additional access methods, such as Tool Palettes. The names of all of the commands can also be typed in the Command Line. A table is included to help you to identify the various methods of accessing the commands.

When typing the name of a command in either the Command Line or Dynamic Input, the **AutoComplete** option automatically completes the entry when you pause as you type. It also supports mid-string search by displaying all of the commands that contain the word that you typed, as shown in Figure 1–12. You can then scroll through the list and select a command.

Figure 1–12

You can also click (Customize) to display the Input Settings for the AutoComplete feature.

To set specific options for the **AutoComplete** feature, right-click on the Command Line, expand Input Settings, and select from the various options, such as the ability to search for system variables or to set the delay response time, as shown in Figure 1–13.

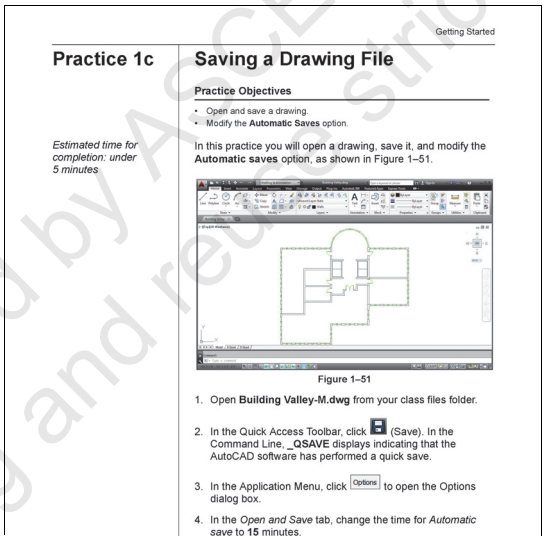
Figure 1–13

If you need to stop a command, press <Esc> to cancel. You might need to press <Esc> more than once.

As you work in the AutoCAD software, the software prompts you for the information that is required to complete each command. These prompts are displayed in the drawing window near the cursor and in the Command Line. It is crucial that you read the command prompts as you work, as shown in Figure 1–14.

Side notes

Side notes are hints or additional information for the current topic.



Getting Started

Practice 1c Saving a Drawing File

Practice Objectives

- Open and save a drawing.
- Modify the **Automatic Saves** option.

Estimated time for completion: under 5 minutes

In this practice you will open a drawing, save it, and modify the **Automatic saves** option, as shown in Figure 1–51.

Figure 1–51

1. Open **Building Valley-M.dwg** from your class files folder.
2. In the Quick Access Toolbar, click (Save). In the Command Line, **_QSAVE** displays indicating that the AutoCAD software has performed a quick save.
3. In the Application Menu, click Options to open the Options dialog box.
4. In the *Open and Save* tab, change the time for *Automatic save* to **15** minutes.

Practice Objectives

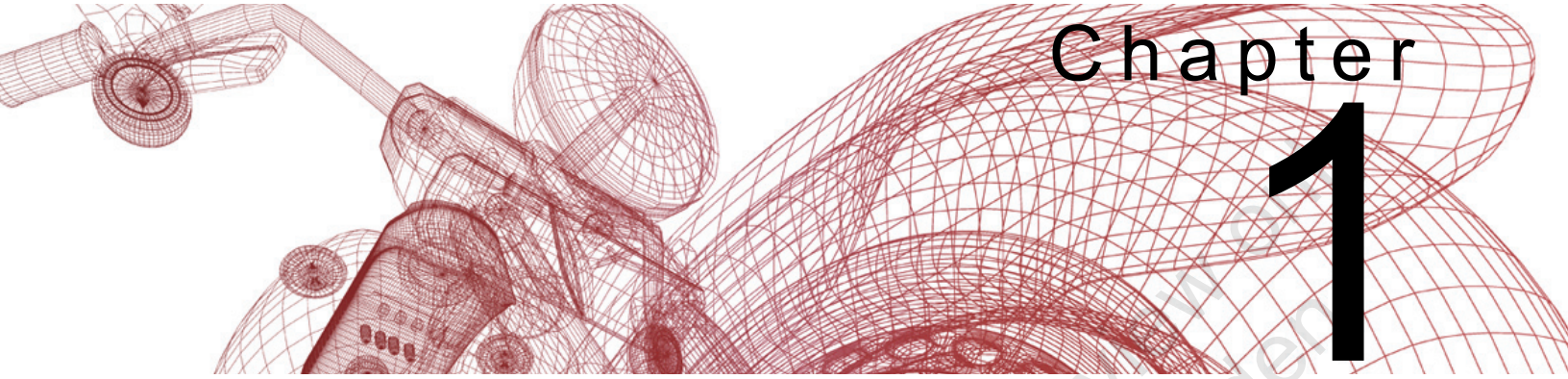
Instructional Content

Each chapter is split into a series of sections of instructional content on specific topics. These lectures include the descriptions, step-by-step procedures, figures, hints, and information you need to achieve the chapter's Learning Objectives.

Practices

Practices enable you to use the software to perform a hands-on review of a topic.

Some practices require you to use prepared practice files, which can be downloaded from the link found on the Practice Files page.



Chapter 1

Fundamentals

Learning Topics in this Chapter

- Visual Basic Evolution
- VBA
- VBScript and CATScript
- VB.NET
- In-process and Out-of-process Applications
- Registering CATIA Libraries

1.1 Visual Basic Evolution

Each incarnation of Visual Basic has had a substantially different syntax, as shown in Figure 1–1

Figure 1–1

- The B.A.S.I.C. (Beginner's All-purpose Symbolic Instruction Code) is a programming language that was originally designed in 1963.
- Visual Basic is Microsoft's dialect of BASIC including controls, events, and IDE. It was very popular in 1980s and 1990s. CATIA V5 Automation was originally designed for VBScript, VBA, and VB6.
- VB.NET is a completely reworked version of VB. It replaces VB6, which Microsoft no longer officially supports. However, it does not replace VBA or VBScript, both of which are still supported by Microsoft. Therefore, VB.NET does not make VBA or VBScript scripts or macros obsolete.

CATIA Programming Platforms

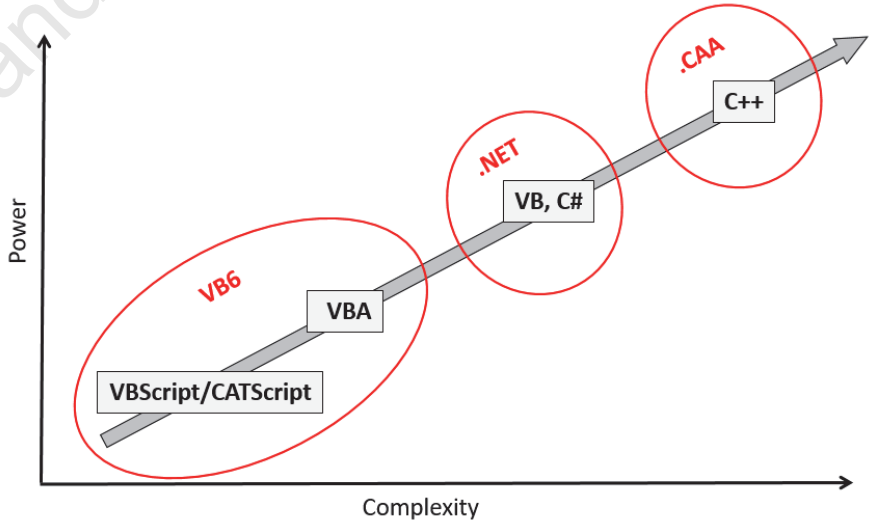


Figure 1–2

1.2 VB6 Platform

Visual Basic 6 (VB6) is an event-driven programming platform and an associated integrated development environment (IDE) for building graphical user interfaces (GUI) in the Microsoft Component Object Model (COM) programming model.

Event-driven Programming

In event-driven programming, the flow of the program is determined by user actions (i.e., mouse clicks, key presses, etc.), or messages from other programs, as shown in Figure 1–3.

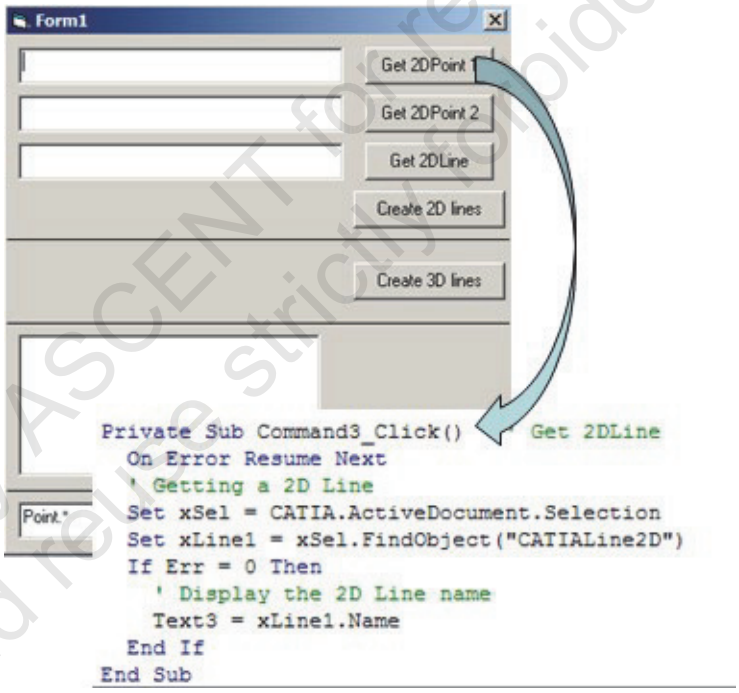


Figure 1–3

In contrast, in sequential programming the flow of the program is determined by the programmer using conditional statements (If...Else If ...Else, etc.), as shown in Figure 1–4.

```
Sub CATDrw_Create()

'MsgBox "Creating FTBs..."

   '--- Init Drawing-scope public variables
   On Error Resume Next
   FTBIA_InitDocument
   If Err.Number <> 0 Then Exit Sub
   On Error Goto 0

   '--- Set Drawing parameters
   FTBIA_SetDrwParams

   Dim DrwSheetCur As DrawingSheet
   Set DrwSheetCur = poDrwSheets.ActiveSheet ' memorize last active sheet

   '--- Loop over all sheets
   For piSheet = 1 To pnSheet
'MsgBox "Creating FTB: Sheet " & piSheet & " of " & pnSheet
      Set poDrwSheet = poDrwSheets.Item(piSheet)
      poDrwSheet.Activate
      FTBIA_CreateSheet
   Next

   DrwSheetCur.Activate                    ' return to the last active sheet

End Sub
```

Figure 1–4

IDE (Integrated Development Environment)

IDE (Integrated Development Environment) is a type of computer software that assists computer programmers in developing applications. It includes a source code editor, compiler or interpreter, build-automation tools, debugger, and object browser, as shown in Figure 1–5.

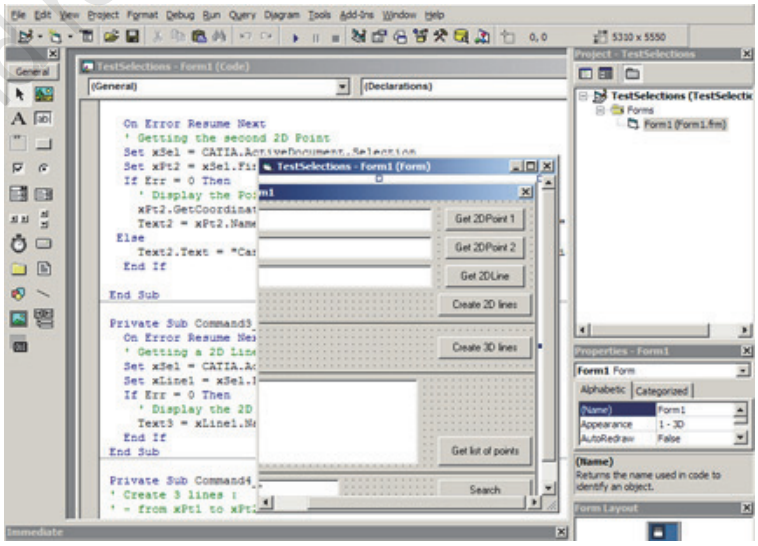


Figure 1–5

GUI (Graphical User Interface)

GUI (Graphical User Interface) is a type of user interface that enables you to interact with the computer using graphical elements (i.e., toolbars, icons, visual indicators, texts, labels, etc.), which represent the available information and actions. The actions are usually performed through direct manipulation of the graphical elements.CATIA software is GUI-enabled, as shown in Figure 1–6.

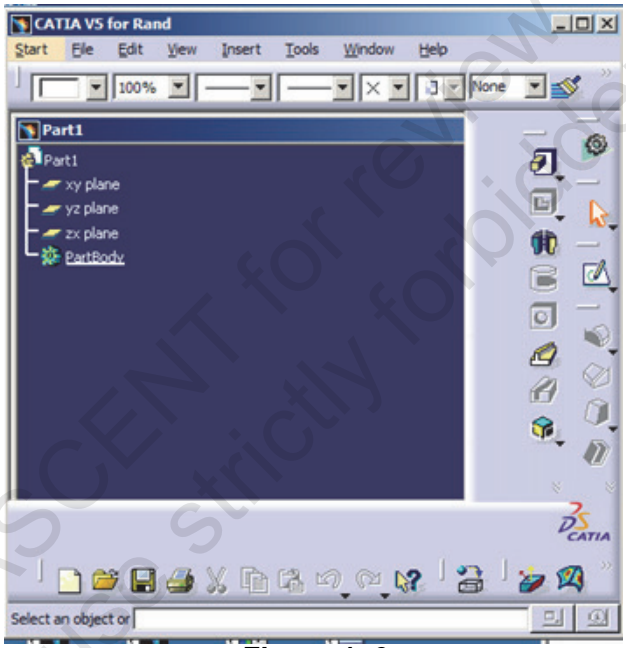


Figure 1–6

COM (Component Object Model)

COM (Component Object Model), shown in Figure 1–7, is a Microsoft technology designed to enable communication between software components written in any programming language. The term COM is often used as an umbrella term that encompasses OLE, Automation, ActiveX, COM+, and DCOM technologies.

Figure 1–7

The essence of COM is a language-neutral way of implementing software components. COM standard permits the reuse of components with no knowledge of their internal implementation because it forces software programmers to provide standard binary interfaces that are separate from the implementation.

CATIA V5 is COM-enabled software. Codes for COM objects or components are stored in DLL libraries and registered in the Windows registry. Therefore, they can be called, initiated, or created at any time.

A software component can work as either a client, or a server. For example, if object A calls object B, object A is then the client and object B is the server (the one that provides services to the client). The client does not have to know the functions or methods (i.e., interface) that the server contains in advance. COM provides for a way to discover the server's interface when the program executes. This is called *late binding*. If the client is made aware of the server's interface before the program begins executing, this is called early binding.

1.3 VBA

The VB6 environment can be hosted in another application, such as Excel, Word, CATIA, etc. and is event-driven and GUI-oriented. It has full IDE (i.e., it contains an editor, debugger, type checker, object browser, etc.) as shown in Figure 1–8. However, it cannot run without the host application running. The source code can only be interpreted (i.e., executed line-by-line), and might be slower than a compiled VB application.

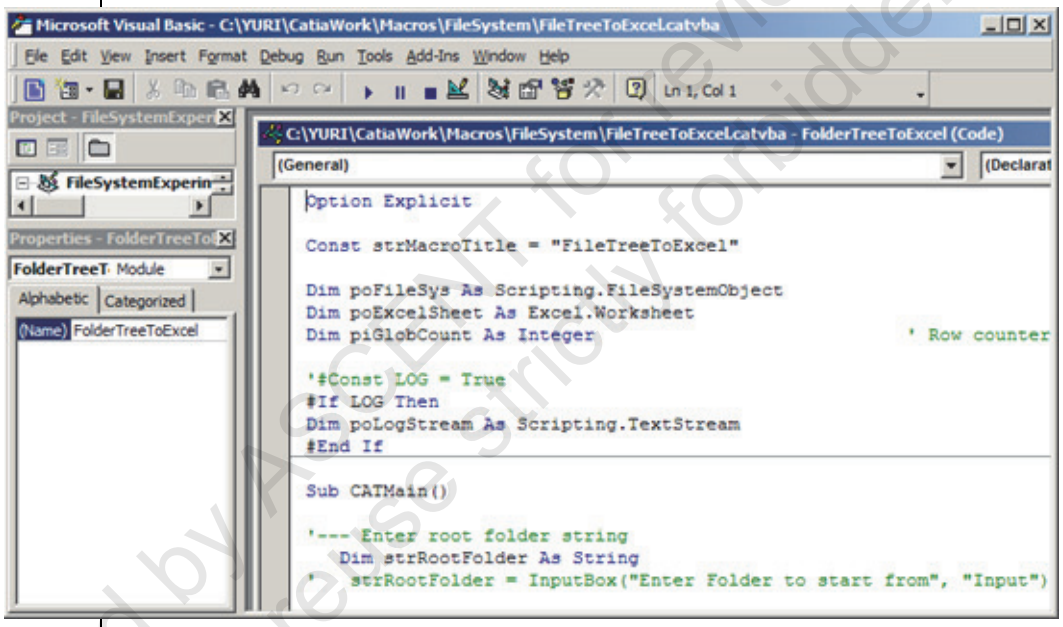


Figure 1–8

1.4 VBScript and CATScript

VBScript is a subset of full VB syntax (some programming features are omitted). It is designed to run in Web applications (i.e., Internet Explorer, etc.) and is an extremely simple and limited tool. It does not contain IDE, debugging etc., and regular text editors are used for coding. It is non-GUI-oriented and only a couple of very simple interactive dialog boxes are available. It is non-event-driven and its code represents a flow with a sequence of actions that is determined by the programmer, as shown in Figure 1–9.

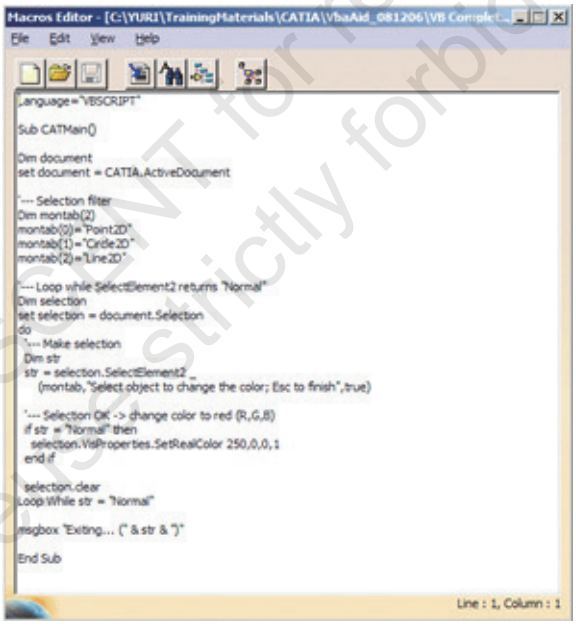


Figure 1–9

CATScript is a portable version of VBScript. CATScript macros can run on Unix.

1.5 VB.NET

VB.NET is the latest incarnation of the popular Microsoft programming environment. It is also event-driven, has an integrated development environment (IDE), and is used for building graphical user interfaces (GUI). However, it is not based on COM (although it can call COM objects) and the syntax is quite different from VB6.

The following difficulties are normally encountered when switching from VB6 to VB.NET:

- New syntax
- New IDE
- New GUI controls
- New InstallShield

When using VB.NET for CATIA automation, CATIA V5 is still a COM application. There is no change to how CATIA objects are called and used.

1.6 In-process and Out-of-process Applications

There are two ways in which a VB application or macro can communicate with CATIA: as an in-process or an out-of-process application.

In-process Applications

The VB application runs within the CATIA process in the computer memory. No inter-process communication is involved. CATIA freezes while the VB application is running. The VB application's allocation memory is wiped out after each run. Therefore, passing data between two subsequent runs is not possible.

- Use the **Tools>Macros** interface (VBScript, CATScript, or VBA).

Out-of-process Applications

The VB application runs in its own process in the computer memory. It communicates with CATIA using inter-process messaging. CATIA is fully active while the VB application is running. For example, these applications could be VB.NET or C# programs, VBA macros in Excel, Word, etc., or Windows Explorer (i.e., embedded VBScript or Java scripts).

1.7 Registering CATIA Libraries

CATIA libraries are automatically registered in the Windows registry when CATIA is installed. The registration enables you to double-click on a CATPart or CATProduct to automatically start CATIA. A VB application or macro can also locate and communicate with CATIA objects.

Since CATIA libraries might change from release to release, a VB application that is developed or compiled with one CATIA release might or might not run correctly with another CATIA release. For example, a VB application developed for V5R21 might not work correctly with V5-6R2012, etc. Therefore, it is recommended that you have the correct CATIA release registered before the VB application is used.

If several CATIA releases are installed on a computer, registering a specific release can be done by executing the following command from the Command Prompt:

<catia_executable> /regserver -env <catia_envir_file> -direnv <catia_envir_directory>

For example, the following command registers CATIA V5-6R2018 libraries on a Windows 10 64-bit system:

"C:\Program Files\Dassault Systemes\B28\win_b64\code\bin\CNEXT.exe" /regserver -env CATIA.V5-6R2018.B28 -direnv "C:\ProgramData\DassaultSystemes\CATEnv"