Converting Avenue to VB/VBA

A “wraparound” Story

by Constantine N. Tonias, P.E. and Elias C. Tonias, P.E.

With the increased use and popularity of ArcGIS™, the need to migrate Avenue based applications to the ArcGIS™ environment has increased significantly. As such, many developers are faced with the dilemma of how to efficiently convert their Avenue code into a format that utilizes ArcObjects, and which is compatible with ArcGIS™. One approach, which this article addresses, is to develop a series of “wraparounds” that facilitates the conversion process. That is, to create a library of subroutines and functions that emulate the function of Avenue requests. By establishing a one to one correspondence between Avenue requests and “wraparounds”, the developer is able to substitute an Avenue request with the appropriate “wraparound”.

Like most new explorations, it is the pioneers that catch the first arrows. Having gone through the process of converting thousands of lines of Avenue code, the authors have learned, and wish to share, a few lessons which might benefit their fellow Avenue programmers who will be converting their applications to the ArcGIS™ environment.

First, before performing any conversion work, it is recommended that the:

- Visual Basic Environment (VB and VBA), and
- Visual Basic for Applications Development Environment

sections in the **ArcObjects Developer Help** be read. These sections can be found under the **Contents** tab, clicking on **Getting Started**, followed by clicking on the **Getting Started Start Page** item. These are not very long sections but they provide valuable information and insight into the conversion process.

Second, it is recommended that the initial conversion work be performed in the VBA, Visual Basic for Applications, environment and then, if appropriate, build an extension in the VB, Visual Basic, environment. The reason for doing so is that the VBA environment is similar to the Avenue environment in that the programmer can easily test and debug the application directly within ArcMap™. Once the application has been tested and is ready for distribution, the programmer can either:

- Package the application as a protected ArcMap template file (.mxt), or
- Build an ActiveX DLL in the VB, Visual Basic, environment.

Although there will be some duplicitous work, in building the ActiveX DLL (i.e. creating tools, menu items, etc.), any other approach would lead to a longer development/conversion cycle.

Since a complete discussion of building an ArcGIS™ extension is outside the scope of this article, we will focus on the specific aspects of converting Avenue code into VB/VBA. In converting Avenue code there are primarily three aspects, which the developer needs to address.

First, there is the issue of the syntax differences between Avenue and VB/VBA. This includes:

- differences in how statements are structured,
- names of intrinsic functions such as square root (sqrt vs. sqr),
- manipulation of string variables,
- handling of lists,

and so forth. These, however, are relatively easy to solve, and do not provide much of a problem.

Second, unlike Avenue, where variables *did not* have to be declared, in the VB/VBA environment variables *do* have to be declared. This issue, although not terribly difficult, is time consuming and tedious. For those who have developed programs in Fortran, C, C++ and the like know, variable declaration is not a glamourous aspect of programming, and is a stark departure from Avenue, where variables could be created without regard to specification of type.

Third, there is the issue of writing the appropriate ArcObjects code to perform the task of the Avenue request that is to be converted. This is where most of the conversion work will be spent, and where the “wraparounds” come into play. By using “wraparounds”, the developer no longer needs to worry about writing the appropriate ArcObjects code. This code is buried within the “wraparound”.

Available in the ESRI book store is a publication entitled *Avenue Wraps*. This publication discusses over 280 “wraparounds” for the most popular Avenue requests and includes a CD containing:

- The source listing for all “wraparounds”,
- Sample data,
- An ArcMap document file containing the “wraparounds”, and
- A VB project file for building a DLL implementation of the “wraparounds”.

The topics covered in the publication include: general Avenue to VB/VBA syntax differences, Views, Themes, Tables, Selection Sets, Graphic Elements, Querying, Calculating, File I/O operations, Message Boxes, Progress Bars, User-Document interaction, Manipulation of Feature Shapes, Legends, Classifications and Application deployment. In addition, *Avenue Wraps* contains numerous samples illustrating how Avenue code can be converted for use in the ArcGIS™ environment.

In Avenue

```
Msg = "Okay to continue ?"
Heading = "My Command"
Default = True
ians = MsgBox.YesNoCancel
      (Msg, Heading, Default)
if (ians = Nil) then
       .... do something
end
if (ians.Not) then
       .... do something
else
       .... do something
end
```

Using Avenue Wraps

```
Dim Msg, Heading As Variant
Dim Default As Boolean
Dim ians As Integer
Msg = "Okay to continue ?"
Heading = "My Command"
Default = True
Call avMsgBoxYesNoCancel _
      (Msg, Heading, _
       Default, ians)
If (ians = vbCancel) Then
       .... do something
End If
If (ians = vbNo) Then
       .... do something
End If
If (ians = vbYes) Then
       .... do something
End If
```

Figure 1

Shown in Figure 1 is an example of how “wraparounds” can be used. The Avenue code included in this figure demonstrates the use of the **YesNoCancel** request as applied to the **MsgBox** class. Below the sample Avenue code is the corresponding ArcObjects code using the “wraparound”, **avMsgBoxYesNoCancel**. In addition to the use of the “wraparound”, note:

- That the variables have been declared using the Dim statement,
- The syntax difference in the use of the “if ... then” statement, and
- The variable “ians” that is passed back by the YesNoCancel request is a Boolean, while the avMsgBoxYesNoCancel “wraparound” returns the same variable as an integer value, which is equal to one of the predefined VB constants.

An example of how the creation of a point feature can be converted from Avenue to ArcObjects with the **avPointMake** “wraparound” is shown in Figure 2. In this example note the use of the “Set” command in the ArcObjects code, which indicates that an object is being defined.

In Avenue

```
aPoint = 5000.0 @ 5000.0
```

Using Avenue Wraps

```
Dim aPoint As esricore.IPoint
Set aPoint = avPointMake _
             (5000.0, 5000.0)
```

Figure 2

In the example of Figure 1, the reader was alerted to the use of the “if..then” statement. Although the overall structure of this statement is similar in both environments, there are some minute differences that may prove bothersome during the conversion. While VB/VBA has an excellent feature in displaying syntax errors, it does not detect the “end” statement of the “if..then” statement as an error. This error is only detected when the code is executed. In addition, VB/VBA is not as simple as Avenue in concatenating statements in one line. Whereas, in Avenue one may write:

```
If (a.NE.B) then x = y End
```

In VB/VBA the above line would appear as:

```
If (a <> b) then
   x = y
End If
```

So that, perhaps one of the first steps performed in the conversion process would be to tackle these types of statements first.

Another troublesome issue is that of the iterative statements such as the “while”, “for each”, “for” and the like. Thus, once the “if...then” statements have been taken care of, the next thing to be tackled could be these iterative statements.

Once the syntax and variable declaration issues have been addressed, the developer can proceed with the conversion of the Avenue requests. The philosophy of writing equivalent procedures for the Avenue requests greatly simplifies the conversion process in that large blocks of code will not have to be rewritten. Instead, a simple name substitution can be made. In so doing, the time required to perform the conversion is dramatically reduced.

Although we have been speaking about the conversion of Avenue Code to ArcObjects, that is not to say that developers, who are used to programming in Avenue, cannot continue writing new code for ArcObjects using the same Avenue approach in conjunction with the “wraparounds” of *Avenue Wraps*.

Since a direct translator from Avenue to ArcObjects does not exist, the next best solution is to have a library of procedures that emulate Avenue requests. The authors hope that the information put forth in this article and in the Avenue Wraps publication help Avenue developers move forward to the ArcGIS™ environment. Like any new endeavor, the biggest hurdle to overcome is getting started. In the case of converting Avenue code, the “wraparound” approach makes that first step a little bit smaller to take.

In describing certain “wraparounds”, *Avenue Wraps* contains sample code demonstrating their use. Additional examples, such as that of Figure 3, can be found on the web at **www.cedra.com**. Once at that site, click the *Avenue Wraps* button, on the left side of the page, and then click the *Avenue Wraps Samples* link to access the additional examples.

About the Authors

Constantine N. Tonias, P. E. is president of The CEDRA Corporation, and overseer of the research and development operations of the corporation. He has been responsible for the development of The CEDRA System, an interactive design and drafting system tailored for the civil engineering practice, major portions of which have been ported to ArcView® GIS and ArcGIS™. His computer experience also includes the conversion of numerous computing programs between various computing hardware and operating system environments.

Elias C. Tonias, P. E. is a technical consultant to The CEDRA Corporation in the development of engineering computing software. He was a pioneer in the application of computers in Civil Engineering, and has written and managed the development of numerous software. Having been involved with software development since the very early days of the computer age, he was forced to go through a series of program conversions from computer environment to computer environment.

How to select a specific record in a table

```
'
Dim pMxApp As _
   esriCore.IMxApplication
Dim pmxDoc As _
   esriCore.IMxDocument
Dim pActiveView As _
   esriCore.IActiveView
Dim pMap As esriCore.IMap
Dim themeList As _
    New Collection
Dim aTable As Variant
Dim rec As Long
'
'  —Get the active view
Call avGetActiveDoc _
      (pMxApp, pmxDoc, _
       pActiveView, pMap)
'
'  —Get a list of only the
'  —tables
Call avGetThemes _
    (pmxDoc, 2, themeList)
'
'  —Get the first table
'  —in the list
aTable = themeList.Item(1)
'
'  —Set the record to be
'  —processed
rec = 12
'
'  —Clear the current
'  —selection set for the
'  —table
Call avClearSelection( _
      pmxDoc, aTable)
Call avUpdateSelection ( _
      pmxDoc, aTable)
'
'  —Update the display to
'  —reflect the new
'  —selection
Call  avGetDisplayFlush
'
'  —Select the specific
'  —record, 12, in the
'  —table
Call avBitmapSet ( _
      pmxDoc, aTable, rec)
Call avUpdateSelection ( _
      pmxDoc, aTable)
'
'  —Update the display to
'  —reflect the new
'  —selection
Call  avGetDisplayFlush
```

Figure 3