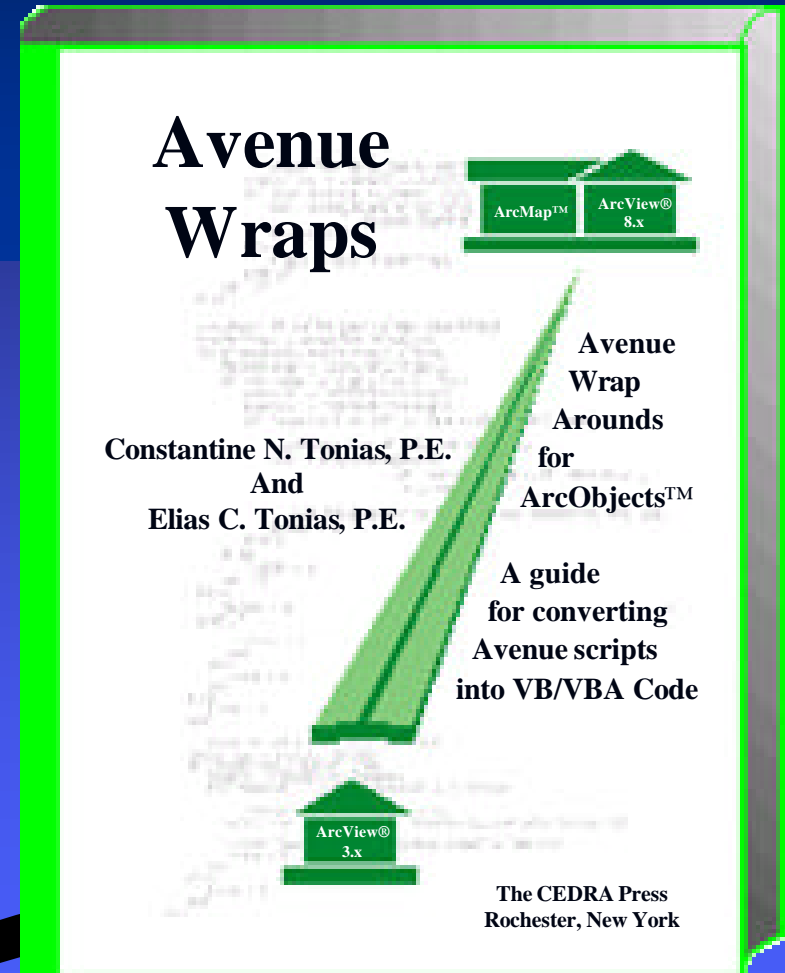


Avenue to ArcGIS

Is there an efficient process for converting Avenue scripts to ArcGIS?



Avenue to ArcGIS

Avenue Wraps is a library of
“wraparound” VB / VBA procedures
that provide a
one to one correspondence to **Avenue**
requests

Avenue to ArcGIS

Avenue



ArcGIS

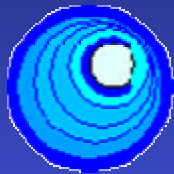
Like most
new
explorations



The pioneers
catch the
first arrows

Avenue to ArcGIS

Conversion of
over 500,000 lines of Avenue scripts



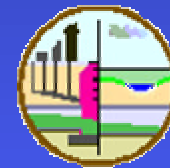
CEDRA-AVcad
General CAD



CEDRA-AVland
Road & Site Design



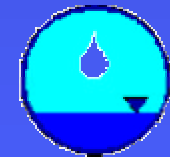
CEDRA-AVcogo
Comprehensive COGO



CEDRA-AVsand
Storm & Waste Water



CEDRA-AVparcel
Parcel maintenance



CEDRA-AVwater
Water Distribution

Avenue to ArcGIS

Development of the Application

Develop &
Test Within

ArcView

Avenue

ArcMap

VBA

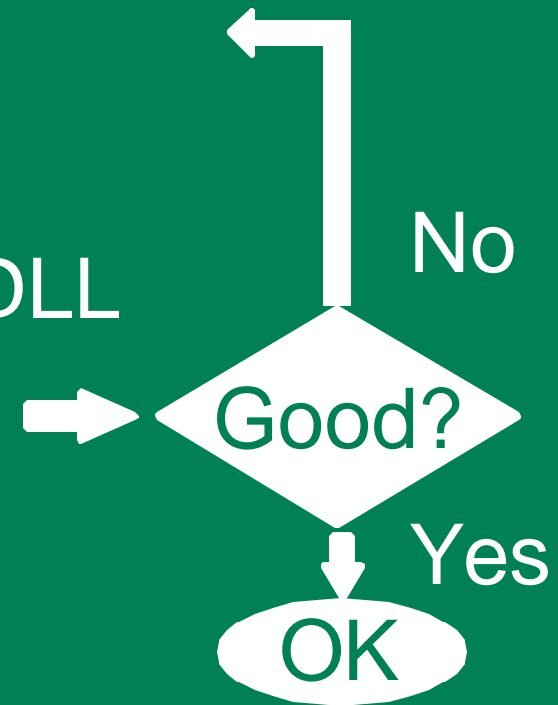
Other COM Languages

Develop

Compile

Create DLL

Test



Avenue to ArcGIS

Conversion Concerns

- ✍ **Syntax of Code**
- ✍ **Declaration of Variables**
- ✍ **Use of ArcObjects instead of Avenue Requests**

Avenue to ArcGIS

Conversion Concerns #1 - *Syntax of Code*

Things to be done first:

1. **Statement structure (if..then..end,
for each..end, etc.)**
2. **Intrinsic function names (sqrt vs. Sqr, etc.)
and position of the name**
3. **Manipulation of strings (count vs. len, etc.)**
4. **Concatenation statements**
5. **Lists vs. Collections**

Avenue to ArcGIS

Conversion Concerns #2 – *Variable Declaration*

Avenue

VB / VBA, C++, etc.

No

Yes

- ✍ Declare as you convert the code, then move to the top in the order of encounter.
- ✍ May declare more than one variable with one Dim statement, but
- ✍ Declare each variable individually

Avenue to ArcGIS

Conversion Concerns #3 - *Requests to ArcObjects*



`av.GetActiveDoc`

`avGetActiveDoc`

`Application.Document`

Avenue to ArcGIS

Avenue

```
Msg = "OK to continue ?"  
Title = "A command"  
Default = True  
ians = MsgBox.YesNoCancel(Msg,Title,Default)  
if (ians = Nil) then  
    ....do something  
else  
    ....do another thing  
end
```

Avenue Wraps

```
Dim Msg As Variant, Title As Variant
Dim Deflt As Boolean, ians As Integer
Msg = "OK to continue ?"
Title = "A command"
Deflt = True
Call avMsgBoxYesNoCancel(Msg, Title, Deflt, ians)
If (ians = vbCancel) then
    ....do something
Elseif (ians = vbNo) then
    ....do something
Elseif (ians = vbYes) then
    ....do another thing
End If
```

Concatenated if...then...end Statements

Avenue

```
if (a.NE.b) then x = y end  
if (c.GT.b) then x = z end
```

Recommended VB / VBA

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

Iterative Statements

Avenue

for each i in 1..55

.....

if (c.GT.b) then

 Break

end

.....

End

VB / VBA

For i = 1 to 55

.....

if (c > b) then

 Exit For

End If

.....

Next

Error Trapping

Public Sub ShowErrorTrapping()

..... Declaration Statements

On Error GoTo ErrorHandler

Call **avGetActiveDoc**(pMxApp, pmxDoc, _
pActiveView, pMap

....do some things

Exit Sub

ErrorHandler:

MsgBox "Error " & Err.Number & " " - " & _
Err.Description & Chr(13) & "Subroutine: _
ShowErrorTrapping"

End Sub

Application - Document Avenue Wraps

Avenue

theView = av.**GetActiveDoc**

vThemeList = theView.**GetVisibleThemes**

Avenue Wraps

Dim pMxApp As IMxApplication

Dim pmxDoc As IMxDocument

Dim pActiveView As IActiveView

Dim pMap As IMap

Dim vThemesList As New Collection

Call **avGetActiveDoc**(pMxApp, pmxDoc, _
pActiveView, pMap)

Call **avGetVisibleThemes**(pmxDoc, vThemesList)

Avenue to ArcGIS

Indexing of Lists and Collections

Avenue

```
firstVTheme = vThemesList.Get(0)
```

VB / VBA

```
firstVTheme = vThemesList.Item(1)
```

Avenue to ArcGIS

File I/O Avenue Wraps

Avenue

```
aLineFile = LineFile.Make (aFileName, _  
                             #FILE_PERM_WRITE)
```

Avenue Wraps

```
Dim aFileName As String  
Dim aLineFile  
Set aLineFile = avLineFileMake _  
                (aFileName, "WRITE")
```

Avenue to ArcGIS

Theme & Table Avenue Wraps

In Avenue, you Operate on an:

FTab or VTab (SetValue and ReturnValue)

With the Avenue Wraps, you use:

IFeature, when dealing with an FTab, and
IRow interface, when dealing with a VTab
with the Value property, *you do not use
IFields to store/extract attribute values !!*

Avenue to ArcGIS

Write the value 24 in record 12 (**Avenue**)

```
theView = av.GetActiveDoc  
theTheme = theView.FindTheme("L_0In")  
theFTab = theTheme.GetFTab  
col = theFTab.FindField("MAP")  
rec = 12  
theFTab.SetEditable(true)  
theFTab.SetValue(col, rec, 24)  
theFTab.SetEditable(false)
```

Write the value 24 in record 12 (**Avenue Wraps**)

```
Dim pMxApp As IMxApplication, pmxDoc As IMxDocument
Dim pActiveView As IActiveView, pMap As IMap
Dim theFTab As IFields, pFeatCls As IFeatureClass
Dim pLayer As IFeatureLayer
Dim col As Long, rec As Long
Call avGetActiveDoc(pMxApp, pmxDoc, pActiveView, pMap)
Call avGetFTab(pmxDoc, "L_0ln", theFTab, pFeatCls, pLayer)
col = theFTab.FindField("MAP")
rec = 12
Call avSetEditable(pmxDoc, "L_0ln", true)
Call avSetValue(pmxDoc, "L_0ln", col, rec, 24)
Call avSetEditable(pmxDoc, "L_0ln", false)
```

Store a 2 point line (shape) in record 12

Avenue

```
aShape = Polyline.Make({{20000.0, 20000.0, 30000.0, 25000.0}})
col = theFTab.FindField("SHAPE")
theFTab.SetEditable(true)
theFTab.SetValue(col, 12, aShape)
theFTab.SetEditable(false)
```

Avenue Wraps

```
Dim aShape As IPolyline, col As Long
Set aShape = avPolyline2Pt(20000#, 20000#, 30000#, 25000#)
col = theFTab.FindField("SHAPE")
Call avSetEditable(pmxDoc, "L_0ln", true)
Call avSetValueG(pmxDoc, "L_0ln", col, 12, aShape)
Call avSetEditable(pmxDoc, "L_0ln", false)
```

Avenue to ArcGIS

Undo / Redo ability with ArcObjects

avSetEditable flushes theme's buffered writes.

Use **avStartOperation** and **avStopOperation** to begin and terminate operations which are added to the Editor's operation stack thereby providing Undo and Redo capabilities.

Use **avStopEditing** to terminate the Editor saving all edits that may have been made.

Avenue to ArcGIS

Feature Selection in Avenue

```
theView = av.GetActiveDoc
theTheme = theView.FindTheme("L_0In")
theFTab = theTheme.GetFTab
col = theFTab.FindField("Deposits")
sel = theFTab.GetSelection
total = 0.0
for each rec in sel
    deposit = theFTab.ReturnValue(col, rec)
    total = total + deposit
end
```


Feature Selection with **Avenue Wraps**

..... (declaration statements)

Call **avGetActiveDoc**(pMxApp, pmxDoc, pActiveView, pMap)

Call **avGetFTab**(pmxDoc, "L_0In", theFTab, pFeatCls, pLayer)

col = theFTab.FindField("Deposits")

Call **avGetSelection**(pmxDoc, "L_0In", sel)

Call **avGetSelectionIDs**(sel, selList)

total = 0#

For iRec = 1 to selList.Count

 rec = selList.Item(iRec)

 Set pFeat = pFeatCls.GetFeature(rec)

 deposit = pFeat.Value(col)

 total = total + deposit

Next

Avenue to ArcGIS

Message and Menu Boxes with **Avenue Wraps**

Avenue

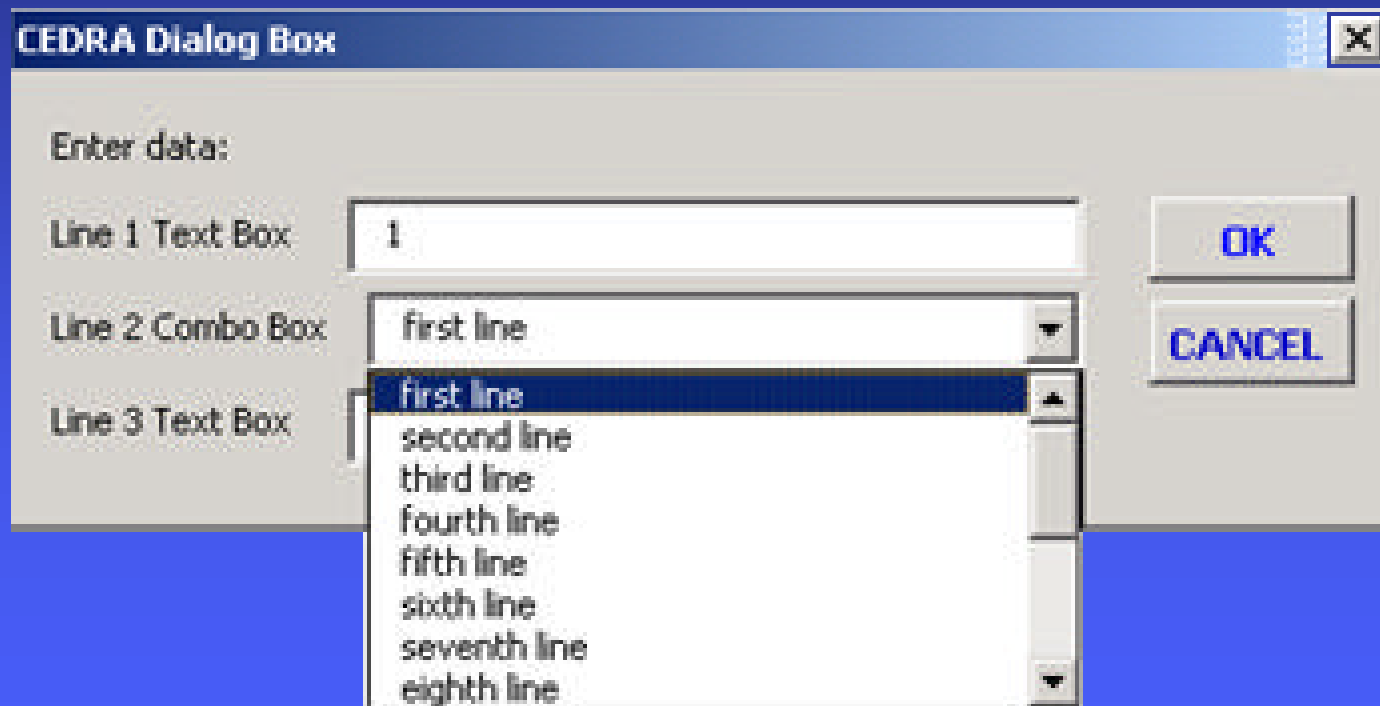
```
MsgBox.Info("A sample message string.",  
            "MsgBox Test")
```

Avenue Wraps

```
Call avMsgBoxInfo("A sample message string.", _  
                  "MsgBox Test")
```

Avenue to ArcGIS

Sample dialog box with 2 data lines and
1 combo box field using **VDBbuild**



The image shows a screenshot of a dialog box titled "CEDRA Dialog Box". The dialog box has a title bar with a close button (X) in the top right corner. The main area contains the text "Enter data:" followed by three input fields:

- "Line 1 Text Box" containing the number "1".
- "Line 2 Combo Box" with a dropdown menu showing "first line" as the selected item.
- "Line 3 Text Box" with a dropdown menu showing a list of items: "first line", "second line", "third line", "fourth line", "fifth line", "sixth line", "seventh line", and "eighth line". The "first line" item is currently selected.

On the right side of the dialog box, there are two buttons: "OK" and "CANCEL".

Avenue to ArcGIS

Sample multi-column & multi-row dialog box with data lines and combo box fields using **HDBuild**

Example of a horizontal dialog box

Enter annotation parameters:

Attribute	Font	Style	Size	Prefix	Suffix	Quadrant
PNT	Arial	Normal	10	PNT=		E
X	Arial	Italic	10	X=		N
Y	Arial	Bold Italic	10	Y=		S
Z	Arial	Bold	10	Z=		W

OK CANCEL

Avenue to ArcGIS

Geometry Avenue Wraps

Creating a point in Avenue

```
aPoint = 5000.0 @ 5000.0
```

Creating a point with Avenue Wraps

```
Dim aPoint As IPoint
```

```
Set aPoint = avPointMake(5000#, 5000#)
```

Create a polyline

Avenue

```
myLine = PolyLine.Make  
          ({{aPt1, aPt2, aPt3, aPt4}})
```

Create a polyline with **Avenue Wraps**

```
Dim shapeList As New Collection, partList As New Collection
Dim aPt1 As IPoint, aPt2 As IPoint, aPt3 As IPoint
Dim aPt4 As IPoint
Dim myLine As IPolyline
Call CreateList(shapeList)
Call CreateList(partList)
partList.Add aPt1
partList.Add aPt2
partList.Add aPt3
partList.Add aPt4
shapeList.Add partList
Set myLine = avPolylineMake(shapeList)
```

Create a 2-point line with **Avenue Wraps**

For 2-point lines, use `avPolyline2Pt` passing coordinates rather than `Point` objects

Dim myLine As IPolyline

Set myLine = **avPolyline2Pt**(xPt1, yPt1, _
xPt2, yPt2)

Avenue to ArcGIS

User Document Interaction

Unfortunately there is no correspondence in the creation of menu items and tools between the **Avenue** and **VB / VBA** environments.

Hence, the developer is basically starting from scratch using native **VB / VBA** functionality to create the GUI.

User Document Interaction

In **Avenue**, there were the:

ReturnUserPoint,
ReturnUserPolyLine,
ReturnUserPolygon,
etc. requests

which were applied to a Display object.

In **VB / VBA**, we need to write code for the type of "event" that is desired (MouseDown, MouseMove, MouseUp, etc).

User Document Interaction

Every tool has a certain set of events, for which the programmer can write code, if so desired.

The developer does not have to write code for every event that is supported by a tool.

Depending upon the operation of the tool, one or many events can be coded.

User Document Interaction

Since **VB / VBA** provides a far more robust environment for dealing with user-application interaction, this is an area where the developer can truly enhance the **Avenue** application in the **VB / VBA** environment.

Avenue to ArcGIS

Graphics and Symbols

The "wraparounds" that operate on graphic elements correspond to the requests used in the *Introduction to Avenue* training book.

One difference between the **Avenue** requests and the "wraparounds" is that the "wraparounds" require the developer to specify the type of graphic that is being processed.

To Create a Point Graphic Element

Avenue

```
aShape = Point.Make(5000.0, 5000.0)  
theShape = GraphicShape.Make(aShape)
```

Avenue Wraps

```
Dim aShape As IGeometry  
Dim theShape As IElement  
Set aShape = avPointMake(5000#, 5000#)  
Set theShape = avGraphicShapeMake _  
("MARKER", aShape)
```

To Create a Point Graphic Element

The first argument in `avGraphicShapeMake` denotes the graphic element type to be created, such as `PEN`, `MARKER` or `FILL`.

In **Avenue** all graphics were added to the view's graphic list.

In **ArcGIS**, a graphic is stored in an annotation target layer. This provides greater flexibility in managing graphic elements.

Avenue to ArcGIS

Classifications and Legends

The terminology between **Avenue** and **ArcObjects** is so completely different. It is difficult to correlate **Avenue** requests to **ArcObjects** (Legend vs. Renderer)

The **classification "wraparounds"** such as **avInterval**, **avUnique**, etc. operate on the theme name rather than on a Legend object.

Classifications and Legends

In **Avenue** the code below assigns a single symbol classification to a theme:

```
thmName = "SomeName"  
aTheme = theView.FindTheme(thmName)  
aLegend = aTheme.GetLegend  
aLegend.SingleSymbol
```

Classifications and Legends

Avenue Wraps

```
Dim pMxApp As IMxApplication, pmxDoc As IMxDocument
Dim pActiveView As IActiveView, pMap As IMap
Dim thmName As String
Dim aDesc As String, aLabel As String
Dim pSym As ISymbol
Call avGetActiveDoc(pMxApp, pmxDoc, pActiveView, pMap)
thmName = "SomeName"
Call avSingleSymbol(pmxDoc, thmName, aDesc, _
                    aLabel, pSym)
```

Classifications and Legends

In the statement

Call **avSingleSymbol**(pmxDoc, thmName, _
aDesc, aLabel, pSym)

The last three arguments allow the user to control:

- the name of the renderer,
- the TOC classification name, and
- the symbol used in displaying the features.

Should default values be used, the statement would appear as:

Call **avSingleSymbol**(pmxDoc, thmName, _
NULL, NULL, NOTHING)

Avenue to ArcGIS

Utility Macros

ArcView Project

**Source
Code**

ArcMap Document

*VBA
only*

**Source
Code**

*No source
code in VB*

Avenue Wraps

ExportVBAcode and LoadVBAcode

Avenue to ArcGIS

Custom Form Creation

In **Avenue**, we could use the **Dialog Designer**.
In **VB / VBA**, we use the **Form Designer**.

Forms created in **VBA** will *need to be recreated* in **VB**.

In addition, some **VB** form controls *contain different properties* than their **VBA** counterparts.

Application Deployment Methods

Application

```
graph TD; Application[Application] --- Testing[Testing]; Application --- VBAcode[VBAcode]; Application --- DLL[DLL];
```

Testing

appl.mxd

VBAcode

module1.bas
module2.bas
form1.frm
form1.frx

DLL

Module1.cls
Module2.cls
form1.frm
form1.frx
appl.vbp
appl.vbw
appl.dll
appl.exp
appl.lib

Avenue to ArcGIS

In Summary

- ✍ A direct translator from **Avenue** to **ArcObjects** does not exist.
- ✍ With **Avenue Wraps**, large blocks of code do not have to be rewritten.
- ✍ Developers used to programming in **Avenue** can develop new code for **ArcGIS** using the same **Avenue** approach they are accustomed with.

Avenue to ArcGIS

In the **Avenue Wraps** book there are several examples on the use of Avenue Wraps.

Additional examples can be found on the web at:

www.cedra.com

On the left side of the page, click the **Avenue Wraps** button, then click the **Avenue Wraps Samples** link.