The following was presented at DMT’09 (May 10-13, 2009).

The contents are provisional and will be superseded by a paper in the DMT’09 Proceedings.

See also earlier Proceedings (1997-2008) [http://ngmdb.usgs.gov/info/dmt/]
Improving ArcGIS workflow: Automation using VBA

Andrew L. Wunderlich
gibbon@utk.edu

Tectonics & Structural Geology Research Group
Department of Earth & Planetary Sciences
and Science Alliance Center of Excellence
University of Tennessee, Knoxville
Introduction

Many tasks associated with editing and QCing GIS datasets can be tedious, repetitive, and time consuming.

Automation of some of these tasks, such as systematic panning around a map during editing and inputting feature attribution, is very desirable.

BUT HOW?
Customizing ArcGIS

• Many forms of customization:
  – Model Builder
  – Layer definition files
  – Simple macros (keystroke combinations)
  – Coding with VBA
  – Stand-alone applications

• All ArcGIS applications (ArcMap, ArcCatalog, etc.) include the Visual Basic for Applications development environment.
VBA in ArcGIS

- From the ArcGIS Desktop Help:
  “Using VBA, you author macros that are stored within the document/template structure of the application you are extending by writing code using the Visual Basic language. You can also create custom commands and tools, called UIControls. UIControls are macros that also contain hooks into the application framework so that you can respond to actions that happen on the buttons or commands you create.”
Programming can be scary...

Don’t panic!

– Many resources available for examples and help with VBA development.
– Huge amount of sample applications and scripts already available for common tasks.
– Simpler syntax makes understanding VB code easier than other languages.

REMEMBER:

Scripting does not have to be complex!
Programming is like illustration…

Imagine that you have been asked to create an illustration of a dog…
Programming is like illustration…

Imagine that you have been asked to create an illustration of a dog…
Programming is like illustration…

Imagine that you have been asked to create an illustration of a dog…
It’s about time!

• Both illustrations are fundamentally the same: they are dogs!

• Vary the complexity and functionality of scripts based on criteria and TIME

• Ask yourself:
  – Does the script need to be bare-bones or robust?
  – How much time can be budgeted for development?
  – How much time will be saved? In the short-term? Long-term?
  – How can I justify the time I need to develop a custom solution to management/supervisor?
Scenario for automation…

Features digitized from georeferenced scan of an old paper map need attribution lifted from labels near points.

1) Start edit session.
2) Open attribute table, select features for attribution.
3) Right-click entry, pan/zoom to feature.
4) Type attribute in table window or feature attribute window and press <Enter>.
5) Pan to or click next feature and repeat.
Pan, zoom to, or select manually, type attribute value in table or window.

Manual attribution of features.
Time the process

- Attribute some features manually (maybe a few dozen) and record the time.
- Divide the time it took by the number of features attributed.
  - In this scenario it took about me about 25 seconds per feature.
- Extrapolate using total number of features to get the total time estimate.
  - In this scenario there are about 1500 features to attribute so…

\[ 25 \times 1500 = 37500 \text{ seconds} / 3600 = \sim 10.5 \text{ hours!} \]
Break the process down

- In its simplest form, the script needs to:
  - See the selected features in a map layer
  - Pan/zoom to the first selected feature
  - Pop up a window to allow user input
  - Find the field for storing the user input and write the value to that field
  - Move to the next feature
- Also:
  - Must allow user to cancel
  - Must handle basic errors without scaring the user too badly
What I came up with…

• Using tutorials and code samples I was able to develop a *very* basic attribute tool in a day.
  – Not sophisticated, worked with the selected features in the highlighted layer in the TOC.
  – Handled features one by one using a simple feature cursor:
    • Loops through features
    • Sets view to the current feature “envelope”
    • Pops up an “InputBox”
    • Writes value to predefined field in table
  – Very simple error handling: script simply quits (gracefully) when a problem is encountered

• With this simple script, time was cut down to around 4-5 seconds per feature! (~2 hours)
With a bit more effort…

• Additional code samples and research (and another day):
  – User can choose layer and attribute field, and query features if none are selected
  – Makes sure user input is appropriate for the target field
  – Cleaner, more efficient code
  – Better error handling

• Further developments (another day or two):
  – Ability to trigger edit session (undo!)
  – Ability to edit features classes that participate in relationships
  – Improved interface: buttons for zooming in and out and flashing current feature
' Pan to each feature, show the form, and store user input
Do Until (pFeature Is Nothing)
    ' Trigger the Editor to create an edit operation
    If m_pEditor.EditState = esriStateEditing Then
        m_pEditor.StartOperation
    Else
    End If
    ' Get the current extent of the map
    Set pEnv = pActiveView.Extent
    ' Get the extent of the current feature
    Set pEnvCurFeat = pFeature.Shape.Envelope
    ' Store the shape of the feature envelope
    Set pAreaEnvCurFeat = pEnvCurFeat
    If (frmAddAttrib.cmdZoomDefault.Cancel = False) Then
        ' User clicked one of the zoom buttons on the form, so keep that zoom level (state)
        ' Center the map on the coordinates of the centroid of the current feature envelope
        ' without changing the zoom level (map scale), effectively panning to the feature
        pEnv.CenterAt pAreaEnvCurFeat.Centroid
        pActiveView.ScreenDisplay.DisplayTransformation.VisibleBounds = pEnv
        pActiveView.Refresh
    Else
        ' Default setting, or user previously clicked zoom to feature on form,
        ' so zoom to the envelope of the current feature and expand it a little.
        ' Set the view envelope to the feature envelope
        Set pEnv = pFeature.Shape.Envelope
        Set pEnvCurFeat = pEnv
        ' Expand the envelope by a distance (False = map units, True = ratio)
        pEnv.Expand 500, 500, False
        ' Set the view to the envelope and refresh, effectively zooming to the feature
        pActiveView.Extent = pEnv
        pActiveView.Refresh
    End If
    ' Make sure refresh is finished before flashing feature for user
    DoEvents
End Do
Here, the custom script ‘AttributeFeatures’ is used to quickly populate the ‘Dip’ attribute for the selected features from the image in the background.

User types in the number, hits <Enter>, and the script automatically moves to the next feature.
Other useful automations…

• Pan current view up, down, left or right with a small amount of overlap (~10%).
• Create data frame current view “footprints”.
• Densify features (lines and polygons) with additional vertices (part of footprint script, will not be discussed separately)
• Write layer symbology reports for map layers that utilize the unique value renderer.
Systematic panning…

- Pan current view up, down, left or right with a small amount of overlap (~10%).
  - Speeds panning around a map for doing editing or QA/QC work.
Data frame footprints

• Create a polygon shapefile that represents the current view extent.
  – Useful for creating data frame “footprints” for inset map locations or graphical indexing.
Example of graphical indexing
Layer symbology report

- Writes a comma separated values (CSV) text file containing the graphical attributes of layer symbology.
  - Useful for checking colors, line weights, etc. in layers that use a unique value renderer:
    - Line and Fill color in RGB and CMYK
    - Line (or outline) weight
    - Rotation (point classes)
    - Symbol size (point classes)
    - X, Y Offset (point classes)
Layer symbology report

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Symbol</th>
<th>Value</th>
<th>Graphic</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>M</th>
<th>Y</th>
<th>K</th>
<th>Width</th>
<th>Angle</th>
<th>Size</th>
<th>XOffset</th>
<th>YOffset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Label</td>
<td>Point</td>
<td>Point</td>
<td>Label50</td>
<td>ADCOH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Line</td>
<td>Line</td>
<td>Line</td>
<td>Line</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Polygon</td>
<td>Polygon</td>
<td>Polygon</td>
<td>Polygon</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Label</td>
<td>Label</td>
<td>Label</td>
<td>Label</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Symbol</td>
<td>Symbol</td>
<td>Symbol</td>
<td>Symbol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Symbol Label</td>
<td>Symbol</td>
<td>Symbol</td>
<td>Symbol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Symbol Color</td>
<td>Symbol</td>
<td>Symbol</td>
<td>Symbol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Symbol Size</td>
<td>Symbol</td>
<td>Symbol</td>
<td>Symbol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
New developer tips…

• Comment code well; it makes scripts easier to follow and to edit.
• New tools can be based on old ones
  – Code that is needed in many of these scripts can be made into “functions” that can be called by any script.
• Save versioned copies of scripts.
  – If a mistake is made in a working script and it causes a crash, old code can be used for comparison or reversion.

And remember: great artists steal…
…so use these free resources!

– Get started with ArcGIS Desktop help:
  • Tutorials, code samples, and instructions for creating custom UIControls (Search for “vba”)

– ESRI Support website (support.esri.com)
  • Search key words for solutions: chances are, it’s been done already!

– ESRI Developer Network (edn.esri.com)
  • Advanced object reference and code samples

– Just Google it!
Final Thoughts

• Time needed to develop custom solutions can be justified through long-term savings.
• Basic programming skills can be achieved in a relatively short period of time with help from tutorials and the bevy of free code samples.
• All code is FREE, taken from PUBLIC resources and forums.
• Contact me if interested in getting source code for any of the scripts discussed: gibbon@utk.edu
Thank you for your time and attention!

Questions?