Geoprocessing in FalconViewTM Using ArcGIS

Mr. Joel Odom
Georgia Tech Research Institute
Atlanta, GA
joel.odom@gtri.gatech.edu

Mr. Chris Bailey
Georgia Tech Research Institute
Atlanta, GA
chris.bailey@gtri.gatech.edu

ABSTRACT

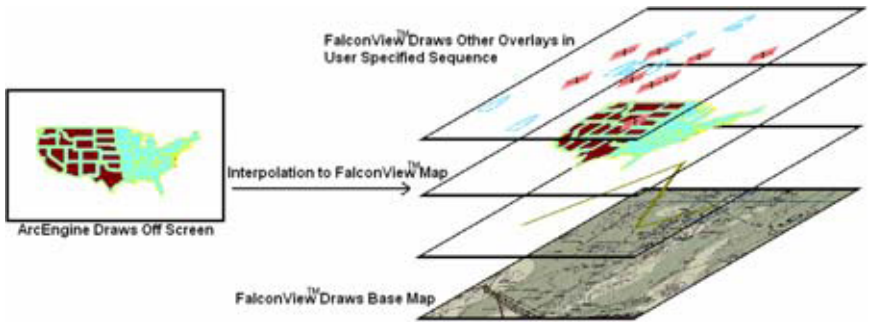
FalconViewTM is a government-off-the-shelf application for displaying and analyzing geographical data crucial to the warfighter. Its ease of use and variety of applications has made it the mission-planning application of choice for the warfighter and the standard for geographic mission data interchange in Iraq and Afghanistan. As part of FalconViewTM 4.2, Georgia Tech Research Institute (GTRI) is expanding the ESRI technology from the Commercial Joint Mapping Toolkit (CJMTK) introduced in 4.1 to allow FalconViewTM users to display ArcGIS data in the same context as other information used by the warfighter such as aeronautical routes, navigation information, and threat information. New features being integrated include ArcGIS features such as feature selection, feature editing, geoprocessing, and tighter integration with native FalconViewTM data. This paper will outline the techniques used to integrate these features and will present the features of the finished product, its user interface, and its utility to the warfighter.

BACKGROUND

FalconViewTM is the mapping and visualization component of the Portable Flight Planning System (PFPS). Initially, the system was used primarily as a pre-mission planner, but later enhancements have added support for in flight situational awareness, re-planning, and after action review. The US Department of Defense and allied nations use FalconViewTM extensively. Developed by the Georgia Tech Research Institute (GTRI), the software displays a wide variety of map and imagery products from the National Geospatial Intelligence Agency (NGA) and from commercial sources. On top of these map backgrounds, FalconViewTM draws a variety of geospatial data as overlays.

US Special Operations Command (USSOCOM) realized the potential of leveraging ArcGIS (via the CJMTK program) in FalconViewTM and funded its integration into FalconViewTM 4.1. The initial desire was to read Shapefiles and personal geodatabases. Once the initial implementation was complete, it became apparent that drawing other data formats was a natural extension of the GIS Overlay capabilities. FalconViewTM 4.1 offers support for web services, map documents, layer files, SDE geodatabases and a host of other formats enabled by ArcGIS.

The FalconViewTM 4.1 GIS Overlay is essentially an ArcGIS viewer. It draws map documents (.mxd files) onto the FalconViewTM map along with other FalconViewTM overlays in the order specified by the user.



FalconViewTM 4.1 also offers limited options to change the way that layers in map documents are symbolized and ordered.

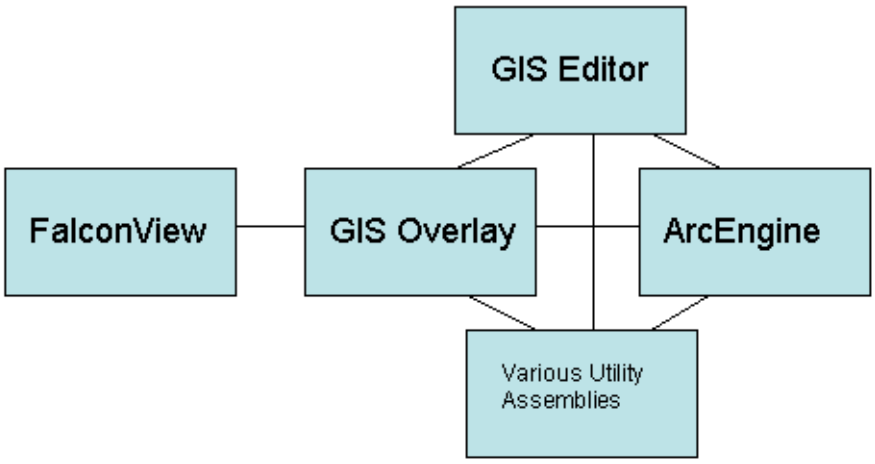
Pleased with the FalconViewTM 4.1 GIS overlay, USSOCOM has funded GTRI to take the ArcGIS capabilities of FalconViewTM a leap further with what is now being called the GIS Editor. The GIS Editor not only allows users to display and symbolize a variety of ArcGIS compatible formats, but users may view feature class attribute data, select feature class data by attribute or by spatial filters, edit feature classes, and access all of the geoprocessing tools. Additionally, FalconViewTM integration allows users

to use FalconViewTM data (tactical graphics, local points, threats, topography data, airports data, etc.) in geoprocessing via the GIS Editor. All of these features are enabled by ArcEngine technology, utilizing many of the capabilities introduced in version 9.2.

GENERAL APPROACH

The FalconViewTM GIS Overlay is implemented as FalconViewTM plug-in. This allows FalconViewTM users who do not have an ArcEngine license to use FalconViewTM without the ArcGIS capabilities. When FalconViewTM detects a licensed install of ArcGIS 9.2, the GIS Overlay becomes active and available in FalconViewTM.

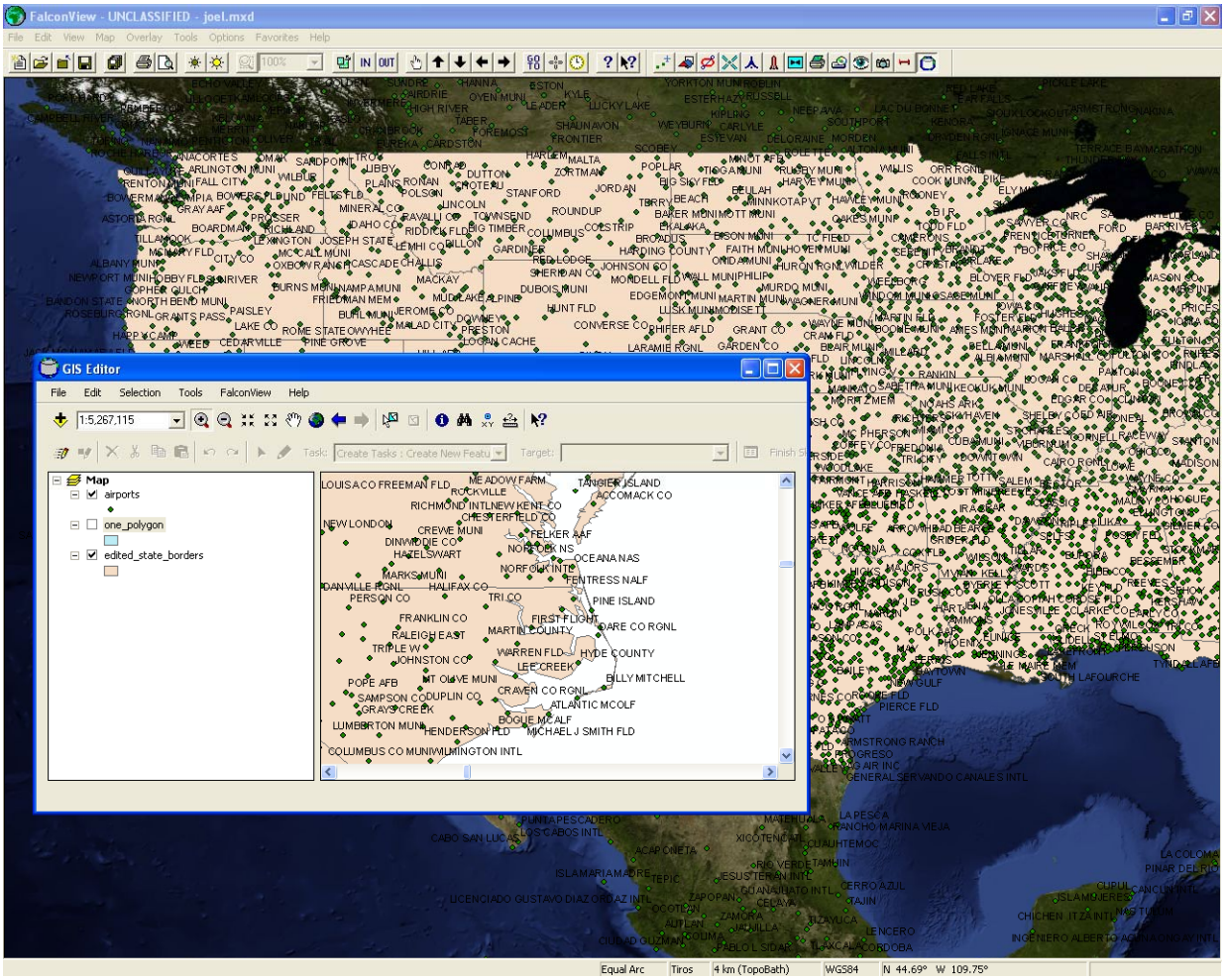
The overall architecture of the GIS Overlay is simple:



The GIS Overlay component utilizes the FalconViewTM COM APIs to draw to the FalconViewTM map and to integrate with FalconViewTM data. The GIS Editor itself is a stand alone assembly where the GIS Editor GUI and most of the geoprocessing logic is housed. The GIS Editor components link to several utility assemblies built for the GIS Overlay. All GIS Overlay components utilize ArcEngine in some fashion. All components of the overlay are implemented in Microsoft .NET 2.0.

When a FalconViewTM user browses to a map document, FalconViewTM calls into the GIS Overlay, which opens the map document. FalconViewTM users may choose to open multiple map documents at once, or they may create empty map documents. When FalconViewTM calls the GIS Overlay to draw, the overlay uses the exporting capability of ArcGIS to output to an in-memory bitmap. The bitmap is then stretched using an interpolation algorithm to geographically match the FalconViewTM map. FalconViewTM may make other calls to the GIS Overlay for purposes such as hit testing mouse clicks and opening the GIS Editor window.

The GIS Editor window, shown below, may be opened or closed, depending on whether the user desires to interact with the map document or not. The general layout of the GIS Editor is comparable to the general layout of ArcMap. This allows users experienced with ArcMap an easy transition to the GIS Editor.



The GIS editor includes a menu of functions, a general purpose toolbar, a feature editing toolbar, a table of contents, and a map control. There is also a user toolbar control which may be customized to add features not supplied in the default control, or to use ArcGIS extension tools.

CHALLENGES

As there was a rather extensive amount of research and development that has gone into the FalconViewTM GIS Overlay, it is beyond the scope of this paper to describe the GIS Editor in its entirety. We will describe here some of the particular challenges of the effort and explain how we solved the related problems.

Drawing to the FalconViewTM Map

Much of the value of the GIS Editor to the warfighter is its ability to draw ArcGIS data seamlessly on top of the FalconViewTM map. We will here describe the particular steps that take place during the rendering process and the notes of interest related to each step.

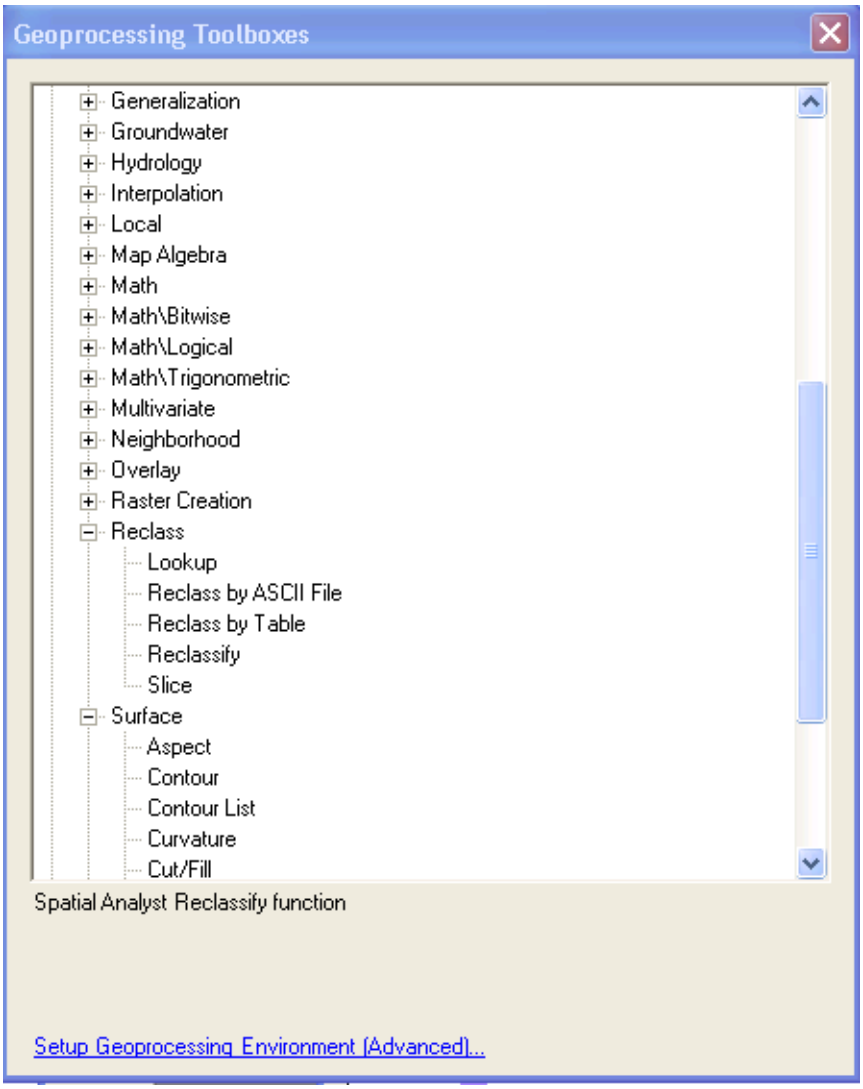
1. FalconViewTM calls the OnDraw method of the GIS Overlay, passing in a reference to the COM object that describes the FalconViewTM map and the particular number of the layer to be drawn (recall that the GIS Overlay may be drawing multiple map documents).
2. The GIS Overlay collects the data necessary for the export from the FalconViewTM COM API. The GIS Overlay then signals the GIS Editor associated with the particular map document being drawn to begin drawing. As some ArcGIS map documents take a long time to render, especially those whose data sources are web services, this rendering process starts in a separate thread from the FalconViewTM main thread. Having the export happen in a separate thread allows the GIS Overlay to return execution to FalconViewTM so that FalconViewTM remains responsive to the user while the ArcGIS layer is rendering. It is of particular note here that, to allows multiple GIS overlays to draw at different times and to keep FalconViewTM responsive, *each GIS Editor actually runs in a separate thread*. (The entire GIS Editor must run in a separate thread because ArcObjects must not be passed across thread boundaries.)

3. The GIS Editor constructs a memory bitmap and uses the Output method of the ActiveView property of the map control to draw to the bitmap. (If the FalconViewTM map spans the date line, the GIS Editor must perform two exports, one on each side of the date line.)
4. Once rendering is complete, the GIS Editor calls back to the GIS Overlay which invalidates the FalconViewTM map. FalconViewTM calls back into the OnDraw method from its own thread. The exported bitmap is stretched and rotated to match the FalconViewTM projection and the final projected output is alpha blended onto the FalconViewTM map.

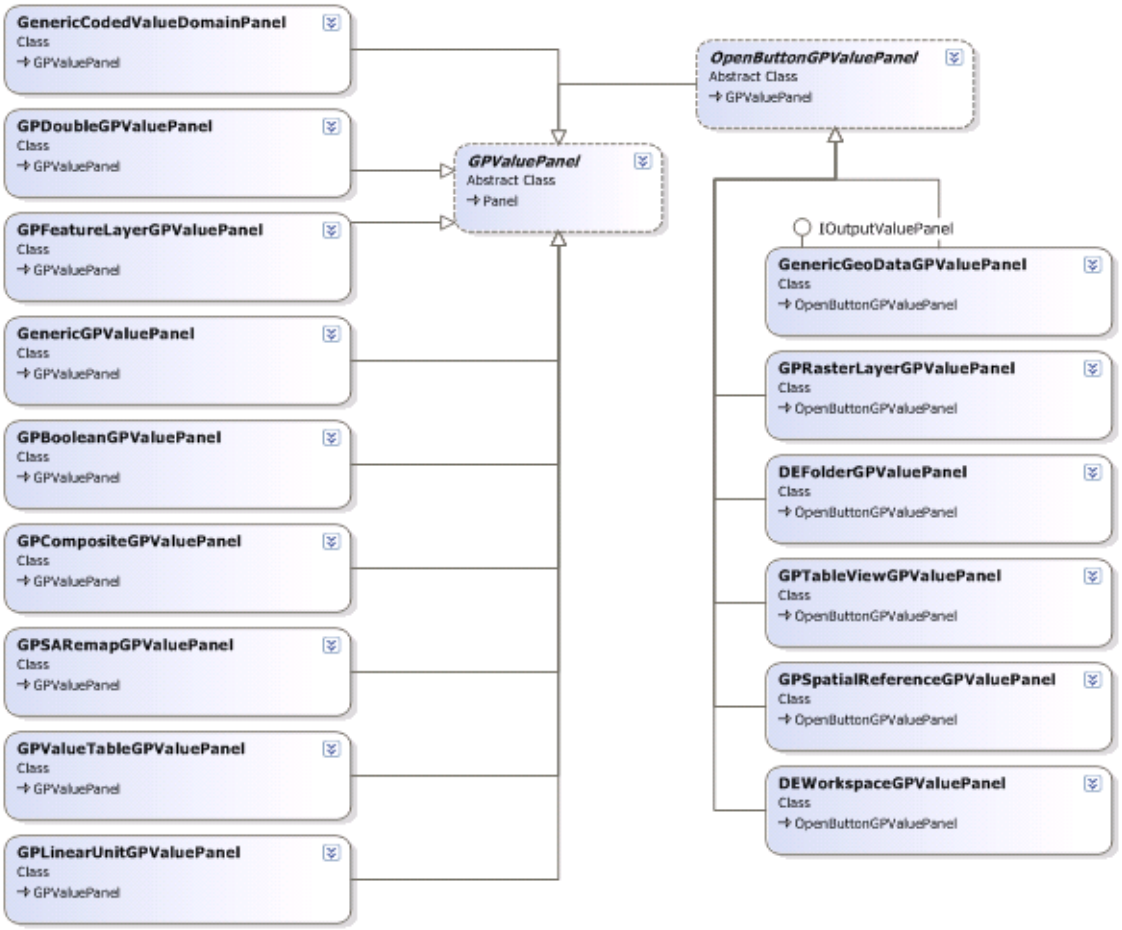
In the development of the GIS Overlay, some debate went into the decision to interpolate the exported bitmap over the FalconViewTM map. Though we considered matching the FalconViewTM and ArcGIS projection parameters, we eventually decided on the rubbersheeting because this would *guarantee* the geographic correlation between the two applications, regardless of projection, zoom, rotation, and any changes that GTRI or ESRI may eventually make in their respective mapping engines.

Geoprocessing

The key feature requested by the government in the FalconViewTM 4.2 GIS Overlay is the ability to conduct geoprocessing tasks using the full power of ArcGIS. In order to present the user with a list of toolboxes, we use an IEnumGPToolbox and a IEnumGPTool to build this tree with a list of licensed, valid tools.

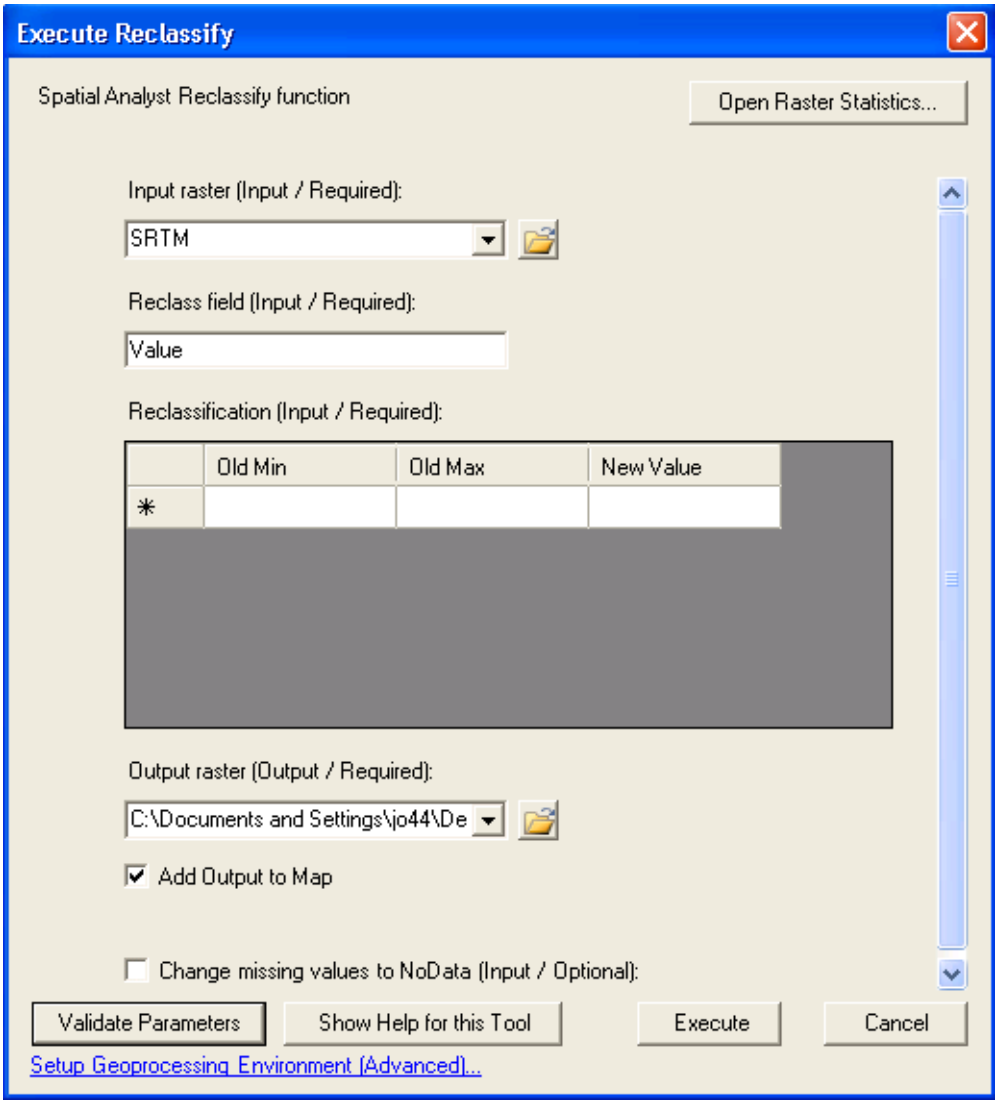


Every geoprocessing tool implements an IGPTool interface which allows us to get the parameter descriptions for the tool. ArcGIS tool parameters may require one of dozens of different value types, and every tool takes a different set of parameters. In order to accommodate the numerous parameter types and combinations, we build each tool's parameters dialog programatically from the ParameterInfo property of the tool. The controls that allow the user to adjust parameters are built from a class hierarchy which allows us to reuse algorithms shared between various parameter types.



The above class diagram shows the abstract base class, GPValuePanel, which itself is a child of the .NET Panel class. All of our geoprocessing value panels derive from the GPValuPanel class. Each value panel has methods and properties which understand their value types, how the user can easily interact with those value types, and how to format their data into a generic IGPValue object for validation and execution. Value panels define their own GUI layout. We have a static method that creates and populates each parameter panel based on the information in ParameterPanel class. If a particular parameter type is not explicitly associated with a class in the static method, we use a GenericGPValuePanel to allow the user to get and set the value as text.

Since we had a limited allotment of time to complete the GIS Editor research and development effort, we concentrated on building user-friendly value panels for the most frequently used value types and for those that were of the greatest interest to our sponsor. The class hierarchy above is designed to be extensible so future versions of the GIS Editor can easily improve the geoprocessing user interface by extending existing classes. For example, the IOutputValuePanel, which defines methods that allow the GIS Editor to add output data to the user's map, could be added to other value panel types, as needed. A sample execute geoprocessing tool form is shown below.

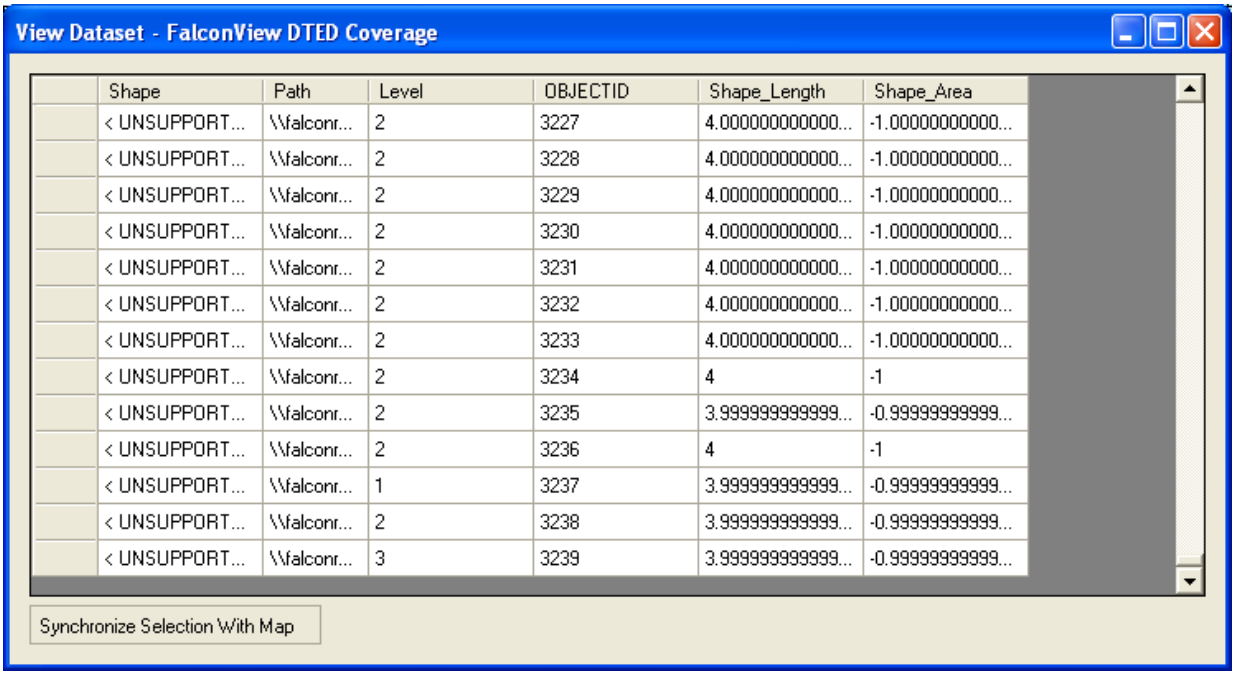


Consuming FalconViewTM Data

The ability to consume data on the FalconViewTM map is an important feature of the GIS Editor. This allows the warfighter to create familiar overlays directly in FalconViewTM and then to use the data for advanced geoprocessing tasks. There are three general categories of data that may be read from FalconViewTM into the GIS Editor: digital terrain elevation data (DTED), static overlays (*e.g.* airports), and user created overlays (*e.g.* tactical graphics).

The design challenge in importing FalconViewTM DTED is how to have the user specify which DTED tiles are of interest. ESRI Military Analyst solves this problem by creating raster catalogs from DTED and then combining those catalogs into something of a "virtual mosaic." While this is a good solution, we wanted our product to function with ArcEngine only, apart from any extensions. In the end, we settled on a two step process for importing DTED which should be friendly even to novice users.

First, the user creates a feature class that shows all of the DTED tiles available on the system. This is a one-click process from the FalconViewTM menu. When the user chooses this option, the GIS Editor creates a polygon feature class to hold the DTED tiles and then uses the FalconViewTM Map Data Server API to enumerate all of the FalconViewTM DTED tiles, adding them to the new feature class. The output of this operation is shown in the GIS Editor view attributes form below.



Once the new feature class is added to the GIS Editor map control, the user selects the tiles of interest and we use the CreateRasterDataset and Mosaic geoprocessing tools to create the mosaic of DTED. The newly created mosaic may be fed to other terrain analysis geoprocessing tools.

To import vector data from FalconViewTM, we created an abstract utility class called PointData which implements the .NET IEnumerable interface. Objects which derive from PointData may be imported into the GIS Editor via another utility class, PointFeatureLayerFactory. This class exposes a static method which takes a PointData object and enumerates all of the points, adding them to feature class which is then added to the map document being edited. It is relatively simple to wrap FalconViewTM data in a PointData object, thus a variety of FalconViewTM data types will likely be exposed to the GIS Editor in future versions of FalconViewTM. (There is currently no support for geometries other than points, but an abstraction of PointData may be created which will allow for differing geometries.)

ACKNOWLEDGEMENTS

Georgia Tech would like to thank Philip Heede, Ben Conklin, and Adam Reedy - all of ESRI - for their support of this effort.

ABOUT THE AUTHORS

Mr. Joel Odom (joel.odom@gtri.gatech.edu) is a Research Scientist at the Georgia Institute of Technology. Before joining Georgia Tech in 2003, he was a consultant with Accenture and a founding partner of the engineering consulting and prototyping firm Juggernautics. Mr. Odom maintains a professional blog about FalconViewTM, ArcGIS, and other GIS notes of interest at http://giscoder.blogspot.com/.

Mr. Chris Bailey (chris.bailey@gtri.gatech.edu) is a Principal Research Engineer at the Georgia Institute of Technology and the Program Manager for mission planning applications at the Georgia Tech Research Institute (GTRI). Before joining Georgia Tech in 1998, he was Director of Software Development at Firearms Training Systems.