A GIS Approach to Charting Terrain
On Instrument Approach Procedures

Brent M. Baumhardt
Esri Users Conference
July 23, 2015

Pennsylvania State University
Masters in Geographic Information Systems
Volpe Instrument Approach Procedure (IAP)

- Organizes information into logical flow
- Less time searching… more time flying
- Human factors study

Pilot Briefing Bar – Notes/Comm
Planview – Complete IAP extent
Profile – Altitude stepdowns
Minima – Defined minimum descent heights
Technology navigational enhancements:

- Terrain Awareness and Warning System (TAWS)
- Electronic Flight Bags (EFB)
- FMS/GPS
- Improved ATC RADAR coverage
Tool Objectives

➢ Improve IAP situational awareness
➢ Minimize interactions
  ➢ User defines extent
  ➢ ICAO Code and IAP name
➢ Let Python do the work…
  ➢ Transform/import IAP
  ➢ Clip elevation data
  ➢ Mathematically calculate intervals
  ➢ Manage permanent/intermediate data
Three Core Functions

- **IAP Transformation**
- **Terrain Manipulation**
- **User Interactive Extents**

### File Folder

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Origination</th>
<th>Spatial Resolution</th>
<th>Spatial Reference</th>
<th>Format</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>National Elevation Dataset</td>
<td>United States Geological Survey</td>
<td>1 arc second (approx 30 meters)</td>
<td>North America Datum 83 (NAD83) Geographic</td>
<td>GRID Float</td>
<td>Source to generate terrain shaded intervals</td>
</tr>
<tr>
<td>Global Airports</td>
<td>Global Hazards Information Network</td>
<td>N/A</td>
<td>World Geodetic System 1984 (WGS84)</td>
<td>Esri Shapefile</td>
<td>Source providing airfield elevation to mathematically derive terrain shaded intervals</td>
</tr>
</tbody>
</table>
Add-In Manager

My Add-Ins
- Terrain Generator
  Created by: Penn State
- Instrument Approach Procedure Terrain Editor
  Created by: Penn State

Shared Add-Ins
- ArcGIS Online
  Created by: Esri

To install Add-Ins and configure the user interface with Add-In components, use the customize dialog.

Customize

Customize

Customize
IAP GeoPDF to GeoTIFF Import:

- TerraGo functions costly
  - Required for Adobe
  - Required for ArcMap

- Open Source translator works well with numerous languages
def __init__(self):
    self.enabled = True
    self.checked = False

def onClick(self):
    # Obtain the GeoPDF file to transform into a GeoTiff
    currentDir = os.getcwd()
    GeoPDFDir = currentDir + '\\GeoPDF'
    for fileName in os.listdir(GeoPDFDir):
        if fileName.endswith('.pdf'):
            inputFile = (fileName)
            # Define the parameters to pass to command line for GDAL
            sourceFile = currentDir + '\\GeoPDF\\' + inputFile
            outputFile = inputFile.split('.')[0]
            destinationFile = currentDir + '\\GeoPDF\\' + outputFile + '.tiff'
            # Run the GDAL command line
            subprocess.call(['gdal_translate', '-of', 'GTiff', sourceFile, destinationFile])

    env.workspace = currentDir + '\\GeoPDF'

    # Make the raster a raster layer to be added to the dataframe through arcpy.mapping
    tempRaster = "rasterLayer"
    arcpy.MakeRasterLayer_management(destinationFile, tempRaster, "\", "\", "\")

    # Add the transformed GeoTiff to the dataframe
    mxd = arcpy.mapping.MapDocument("Current")
    df = arcpy.mapping.ListDataFrames(mxd, "Layers")[0]
    addLayer = arcpy.mapping.Layer(tempRaster)
    arcpy.mapping.AddLayer(df, addLayer, "AUTO_ARRANGE")
    del mxd

    for dirpath, direname, files in os.walk(GeoPDFDir):
        if not files:
            pythonaddins.MessageBox("There is no PDF available in GeoPDF directory", "GeoPDF To GeoTiff")
Enter ICAO code:

Enter Procedure Name:

Enter Highest Elevation:

Enter Middle Elevation:

Enter Lowest Elevation:

Create Terrain

Add Terrain

Draw Planview Extent
class EnterIcaoCode(object):
    
    
    def __init__(self):
        self.items = []
        self.editable = True
        self.enabled = True
        self.dropdownWidth = 'WWWW'
        self.width = 'WWWW'

    def onEditChange(self, text):
        global icaoParameter
        icaoParameter = text
        
class EnterProcedureName(object):
    
    
    def __init__(self):
        self.items = []
        self.editable = True
        self.enabled = True
        self.dropdownWidth = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
        self.width = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'

    def onEditChange(self, text):
        global procNameParameter
        procNameParameter = text
def __init__(self):
    self.enabled = True
    self.shape = "Line"  # Can set to "Line", "Circle" or "Rectangle" for interactive shape drawing and to activate the mouse cursor
    self.cursor = 3
    self.width = 'WWWWWWWW'

def online(self, line_geometry):
    # Check to see if ICAO Code parameter equals 3 characters, if not return a message informing the user to change the input
    if len(icaoParameter) == 3:
        array = arcpy.Array()
        part = line_geometry.getPart(0)
        for pt in part:
            array.add(pt)
        array.add(line_geometry.firstPoint)
        closePoint = array[0]
        array.add(closePoint)

        # Define the feature class and insert cursor
        currentDir = os.getcwd()
        planViewExtents = currentDir + '\Data\TerrainGeneratorData.gdb\PlanViewExtents'

        cursor = arcpy.da.InsertCursor(planViewExtents, ("SHAPE@", "ICAO", "PROC_NAME"))

        # Build the polygon extent from the array
        polygon = arcpy.Polygon(array)

        # Insert the polygon into the feature
        cursor.insertRow((polygon, icaoParameter, procNameParameter))

        del cursor

    else:
        pythonaddins.MessageBox("ICAO Code greater than 3 letters, enter in a 3 letter ICAO Code", "ICAO Code Error")
*** Circling not authorized E of Rwy 3R-21L.
When Rwy 3L VGS IOP, circling to Rwy 3L NA at night.

Minimum nav aid reception at
ARCOF 15,000 MSL.

NOT FOR CIVIL USE

RADAR OR DME REQUIRED
EMERG SAFE AIT 100 NM 14,000
Terrain Generation

Mosaic Dataset/Planview Extents

Clipping

#Extracting the field elevation of the airport
airports = currentDir + "\Data\TerrainGeneratorData.qdb\US_Airports"
locIDField = "LOCID"
elevField = "ELEV"
airfieldCursor = arcpy.da.SearchCursor(airports, elevField, "" + locIDField + 
for row in airfieldCursor:
    fieldElevVal = int(row[0])
    print "Extracted field elevation is : " + str(fieldElevVal)
#Defining the terrain interval range
lowElevBand = int(math.ceil((fieldElevVal/1000.0))*1000.0 + 1000.0)
diffLowElevMaxElevCalc = int(rasterMaxElevVal - lowElevBand)
bandIntervalCalc = int((diffLowElevMaxElevCalc / 3.0 - 3.0))
midElevBand = lowElevBand + bandIntervalCalc
topElevBand = midElevBand + bandIntervalCalc
print "High elevation interval is: " + str(topElevBand) + " ft"
print "Mid elevation interval is: " + str(midElevBand) + " ft"
print "Low elevation interval is: " + str(lowElevBand) + " ft"
#Define remapping for terrain intervals
terrainRemap = RemapRange([[lowElevBand, midElevBand, int(lowElevBand)], [midElevBand, topElevBand, int(midElevBand)]], [3.0, 3.0, 3.0])

Mathematical Reclassification Remap

Terrain Polygon

Field Calculations
Symbolize with standard layer properties
Export map to PDF
IAP Terrain Editor

- Quickly update revised IAPs
- Re-import procedure
- Pre-populated combo boxes
- Retrieves stored terrain
Future Enhancements:

- Consumption within electronic flight bags
- Managed within web service
  - User accounts
  - Independent of unique software

- Topographical additions:
  - Transportation
  - Populated Areas
  - Landcover
Summary:

- Performance based navigation will skirt the landscape
- Personalizes navigational situational awareness
- Alternative to expensive avionic systems