GNIS: West Virginia Summits

Update Process, February 2012

Prepared by: Tyler R. Wylie, Kevin Kuhn

Goal:

The goal of this project was to update the GNIS summit elevations for West Virginia using current data. The need for this project arose from discrepancies between GNIS summit data and USGS NED 1/9th arc-second derived 10' contours; the summit location did not always fall within the expected contour interval. This project ensured that all GNIS summit locations correspond to the actual summit location of the USGS elevation data, while preserving both the GNIS elevation value and USGS NED elevation value.

Procedure:

This project involved creating a process that extracts elevation summit data from the WVSAMB / USGS DEM and assigns it to the related GNIS summit. This required creating a method of quality assurance of elevation, GNIS name, and horizontal point location. The USGS 7.5 minute quad was established as the working data unit due to the fact that both the elevation and GNIS data sets use the USGS quad index grid system. A script was written in Python that automated most of the project tasks performed in ArcGIS 10.

Generate summits: The first phase was to identify summit point locations using an analytical query on the new elevation data. This phase was broken into several tasks. The individual elevation quads were mathematically inverted, creating a negative value elevation raster. This step allowed the use of a hydrological model 'sink' to be identified, which is the location of all elevation peaks (referred to 'generated summits'. This phase resulted in only horizontal locations of all peaks. The generated summit points were then intersected with the original elevation data to assign the DEM elevation value. Due to the nature of the data, many additional summits were identified than previously identified by the GNIS data, therefore a second phase was used to narrow the point locations to those surrounding previously identified GNIS summits.

Focus generated summits: The second phase of the process was to utilize existing GNIS summits as a template to reduce the number of generated summit points. This was done by creating a 1000' buffer around existing GNIS points, and selecting all points that are within that buffer. The selected points were subset from the generated summits and further processed, the remaining points were not used.

Assign GNIS summit: The GNIS summit IDs were then assigned the best generated summit ID. If more than one generated summit was present, selection of the generated summit ID was based upon the horizontal location, closest to elevation summit, and maximum elevation value. If multiple locations of equal elevation were present, the ID was assigned manually during the QC. The resulting file could be joined by elevation summit ID to GNIS attributes. This meant points were never actually moved, rather the GNIS attributes were assigned to the new location (this preserved the horizontal location).

Quality Assurance: The third phase in the process performed a quality check of all automatically referenced points and addressed any GNIS summit not assigned a generated summit ID. Comments were then recorded regarding any information believed to be valuable for further efforts as a feature level metadata record.

Comments:

Figure A

Figure A is an example of a GNIS summit that was manually referenced to one of two generated summits that show the same elevation in feet (RED lettering). By referencing this GNIS point to one of the elevation IDs (GREEN lettering), the GNIS point is being pulled off of the topographic 'X'. The elevation in feet for the generated summits is different from the topographic elevation displayed as well. Therefore: "Spot elev. 1488**."** and "Moved off topo 'X/triangle'**. "** would be recorded in the "comments" section of the attribute table.

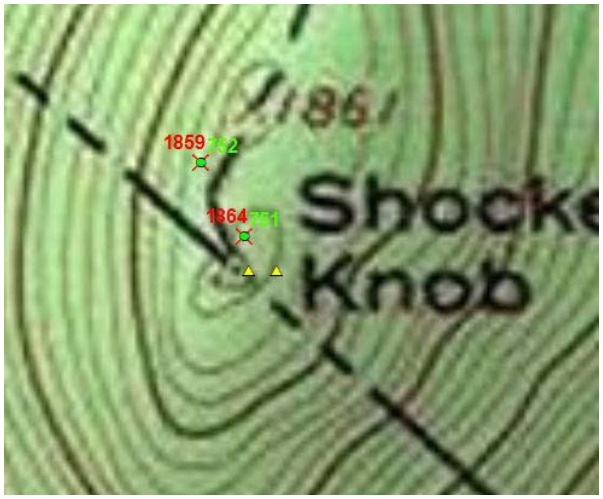

Figure B

Figure B demonstrates a situation where there were separately named GNIS points located for the same summit. In this specific case, the area shows the WV-VA border, so one point is referenced to VA while other point WV. In this situation the "comments" would read "Coincident with Shockeys Knob, VA". Similar situations also occurred with in-state points. In this situation the Feature ID # of the coincident point is written in parentheses following the name. The comments for Figure B would also give respect to the spot elevation to the North with comment: "Spot elev. 1861". Comments were also made for any GNIS points that were moved more than 300 feet from their original location. ("Moved over 300' ").

Comments:

OK: Most points were located adjacent to or directly on a spot elevation mark "X".

Spot elev. '*number*': recorded the elevation listed on the USGS Topographic map.

State Location: The GNIS data was extracted by query from the GNIS website. This resulted in two errors in the data. Points that were attributed to be in WV, but located outside the state boundary and points attributed outside of WV but located within the state boundary.

Point in '*state'*: Points located outside of the West Virginia state boundary.

Point listed in '*VA*': Several points were attributed in the 'State_Alpha' field as "VA", but location is within WV boundaries.

Coincident point: Many points were located adjacent to other GNIS Summits points and were therefore co-located.

Feature Type: It was noted that there were several GNIS summits that were not true summit locations. These features are generally shown as other features on the USGS Topographic map. For example, "Brushy Hill" is not located on a mountain summit, rather on the side of a mountain. Several of these locations were therefore not updated.

Data Results:
Source WV GNIS Summits : n= 988
(where "STATE_ALPH" = 'WV')
WV Summits: n=992

(988 + 4 = 992) 4 points listed as points stated as VA points, but fall within WV.
982 points in WV (10 points outside)

Appendix: Elevation Summit Script

```
# ---------------------------------------------------------------------------
# summits_1.py
# Created on: 2011-09-21 12:01:30.00000
#   (generated by ArcGIS/ModelBuilder)
# Usage: summits_1 <FlowDir__Raster_> <Output_raster__2_> <Name> <NEG__RASTER_>
# Description:
# ---------------------------------------------------------------------------

# Set the necessary product code
# import arcinfo


# Import arcpy module
import arcpy

# Check out any necessary licenses
arcpy.CheckOutExtension("spatial")

# Load required toolboxes
arcpy.ImportToolbox("Model Functions")

# Script arguments
FlowDir__Raster_ = arcpy.GetParameterAsText(0)
if FlowDir__Raster_ == '#' or not FlowDir__Raster_:
    FlowDir__Raster_ = "D:\\gis_data\\temp\\FlowDir_Time1" # provide a default value if unspecified

Output_raster__2_ = arcpy.GetParameterAsText(1)
if Output_raster__2_ == '#' or not Output_raster__2_:
    Output_raster__2_ = "D:\\gis_data\\temp\\Sink_FlowDir2" # provide a default value if unspecified

Name = arcpy.GetParameterAsText(2)
if Name == '#' or not Name:
    Name = "masontown_wv" # provide a default value if unspecified

NEG__RASTER_ = arcpy.GetParameterAsText(3)

# Local variables:
Output_drop_raster = NEG__RASTER_
Delete_succeeded = Output_drop_raster
Output_data_element__2_ = Output_raster__2_
RasterT_Sink_Fl3_shp = Output_data_element__2_
RasterT_Sink_Fl3_FeatureToPo_shp = RasterT_Sink_Fl3_shp
Extract_RasterT2_shp = RasterT_Sink_Fl3_FeatureToPo_shp
Output_data_element = FlowDir__Raster_
Input_raster_or_constant_value_2 = "-1"
DEMs = "D:\\gis_data\\GNIS\\Summits\\DEMs"
Any_value = ""
RASTER = "D:\\gis_data\\GNIS\\Summits\\DEMs\\RASTER"

# Process: Iterate Datasets
arcpy.IterateDatasets_mb(DEMs, "*.tif", "RASTER", "NOT_RECURSIVE")

# Process: Times
arcpy.gp.Times_sa(RASTER, Input_raster_or_constant_value_2, NEG__RASTER_)

# Process: Flow Direction
arcpy.gp.FlowDirection_sa(NEG__RASTER_, FlowDir__Raster_, "NORMAL", Output_drop_raster)
```

```
# Process: Rename
arcpy.Rename_management(FlowDir__Raster_, Output_data_element, "")

# Process: Sink
arcpy.gp.Sink_sa(Output_data_element, Output_raster__2_)

# Process: Rename (2)
arcpy.Rename_management(Output_raster__2_, Output_data_element__2_, "")

# Process: Raster to Polygon
arcpy.RasterToPolygon_conversion(Output_data_element__2_, RasterT_Sink_Fl3_shp, "SIMPLIFY", "VALUE")

# Process: Feature To Point
arcpy.FeatureToPoint_management(RasterT_Sink_Fl3_shp, RasterT_Sink_Fl3_FeatureToPo_shp, "INSIDE")

# Process: Extract Values to Points
arcpy.gp.ExtractValuesToPoints_sa(RasterT_Sink_Fl3_FeatureToPo_shp, RASTER, Extract_RasterT2_shp,
"NONE", "VALUE_ONLY")

# Process: Delete
arcpy.Delete_management(Output_drop_raster, "")
```

Appendix B:

```
# Import modules
import arcpy
import sys

# Check out any necessary arc/ESRI licenses
arcpy.SetProduct("ArcServer")
arcpy.CheckOutExtension("spatial")

# Load required toolboxes
arcpy.ImportToolbox

# Set to overwrite pre-existing files
arcpy.env.overwriteOutput = True

#########################################
## input data and processing variables
#########################################

gnisSummits = "G:\\workspaceTwo\\temp\\summits\\testing.gdb\\testData\\summits_forTest3"
elevPoints = "G:\\workspaceTwo\\temp\\summits\\testing.gdb\\testData\\elevPoints_forTest2"
newDataset = dict()
bufferValue = 200
output = "G:\\workspaceTwo\\temp\\summits\\results.txt"
tempBuffer = "G:\\workspaceTwo\\temp\\summits\\testing.gdb\\testData\\tempBuffer"
selectedElevPoints =
"G:\\workspaceTwo\\temp\\summits\\testing.gdb\\testData\\selectedElevPoints"

def newSummits():

   arcpy.MakeFeatureLayer_management(gnisSummits, "gnisSummits_lyr")
   arcpy.MakeFeatureLayer_management(elevPoints, "elevPoints_lyr")
   desc = arcpy.Describe("gnisSummits_lyr")
   gnisSummitsLyrShapeFieldName = desc.featureClass.shapeFieldName
   rows = arcpy.SearchCursor("gnisSummits_lyr")
   i = 0
   for row in rows:
      currentGnisId = row.FEATURE_ID
      print "I am working on GNIS ID: "
      print currentGnisId
      currentGnisId2 = int(currentGnisId)
      currentGnisId3 = str(currentGnisId2)
      arcpy.Buffer_analysis(row.getValue(gnisSummitsLyrShapeFieldName), tempBuffer, bufferValue,
"FULL", "ROUND", "NONE", "")
      arcpy.SelectLayerByLocation_management("elevPoints_lyr", "INTERSECT", tempBuffer, "",
"NEW_SELECTION")
      arcpy.CopyFeatures_management("elevPoints_lyr", selectedElevPoints)
      arcpy.MakeFeatureLayer_management(selectedElevPoints, "selectedFeatures_lyr")
      selectRows = arcpy.SearchCursor("selectedFeatures_lyr")
```

```
        testValues = []
        for selected in selectRows:
            val = selected.RASTERVALU
            testValues.append(val)
        localMax = max(testValues)
        print "This is the local maximum elevation: "
        print localMax
        whereClause = "RASTERVALU = "+(str(localMax)).strip()
        finalElevRow = arcpy.SearchCursor("selectedFeatures_lyr", whereClause)
        for row in finalElevRow:
            currentElevId = row.ORIG_FID
            print "This is the ID of the new summit feature: "
            print currentElevId
        newDataset[i] = currentGnisId3+","+str(currentElevId)
        print newDataset[i]
        i = i+1
    print "These are the results of my work: "
    print newDataset

##write to output file
    f = open(output,'wb')
    for row,value in newDataset.items():
        f.write(str(value)+'\n')
    f.close()

    return

#################
## main program
#################

newSummits()
```