

# MINE SIGHT®

in the Foreground

Volume 22, Number 10 , November 2006

## Current Affairs

# New Scripts Available in MineSight® V3.60 for Processing Multiple Objects

This article reviews two general purpose scripts available on this year's Update CD, `contoursFromDir.py` and `partialsFromDir.py`. These two scripts process multiple geometry objects that reside in a specified directory.

### Contour all surfaces in a directory – `contoursFromDir.py`

The script, `contoursFromDir.py`, creates geometry objects containing contour lines for all surfaces and solids in a specified directory. This is useful when there are multiple surfaces to be contoured. The surfaces can reside in one or many geometry objects. The script will create geometry objects containing contours of each surface or solid element for each geometry object in the directory. This is an EM-GRAIL script that can only be run inside MineSight® 3-D.

To run the script, use the **Run Script...** option from the MineSight® menu bar (**File | Scripts | Run Script**) and browse for `contoursFromDir.py`:

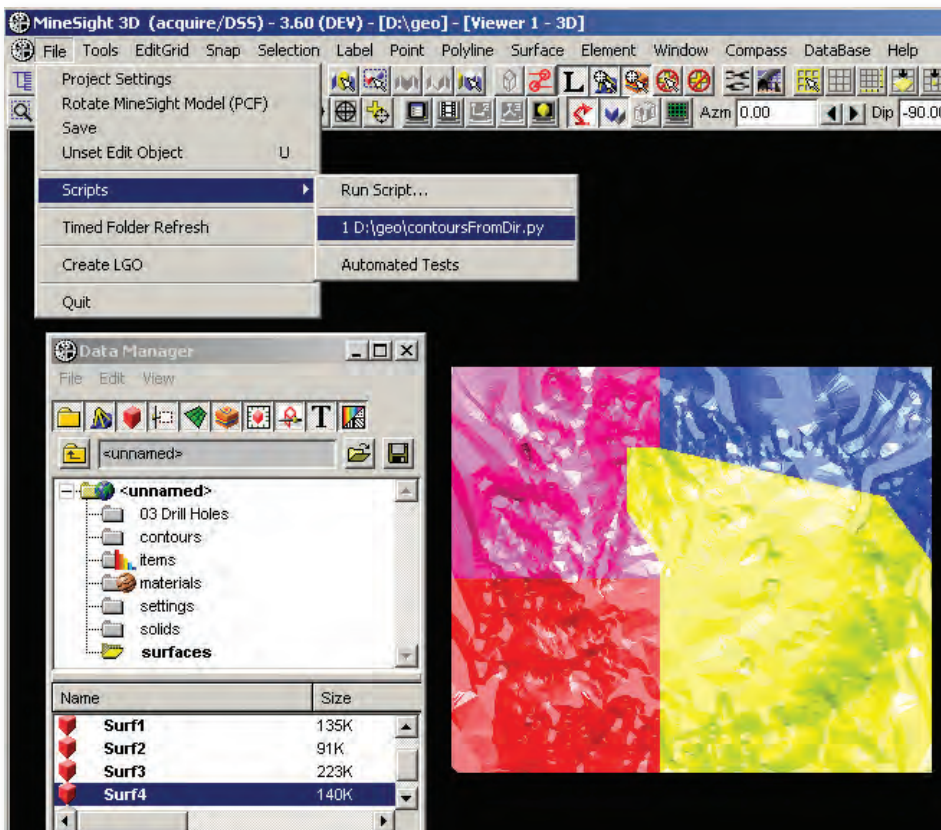


Figure 1 Run script from MineSight® 3-D

(continued on page 3)

### Inside This Issue:

- 2006 Training Schedule... page 11
- 24th Annual Mintec Seminar... page 12
- Current Affairs: New Process Scripts Available in MineSight® v.3.60 for Processing Multiple Objects... page 1
- MineSight® License Price Increase... page 2
- Mintec Directory... page 2
- Mintec to Open New Australasia Office in Perth... page 2
- Tips from Tech Support: Loading Composite Data to MineSight®... page 7
- Trade Shows and Seminars... page 12
- Web-based Training... page 11

# Current Affairs

*A Window on Software Engineering*

(New Scripts Available in MineSight® V3.60 for Processing Multiple Objects continued from front page)

The script presents a graphical user interface (GUI) where required parameters, as well as some additional options, are entered (Figure 2).

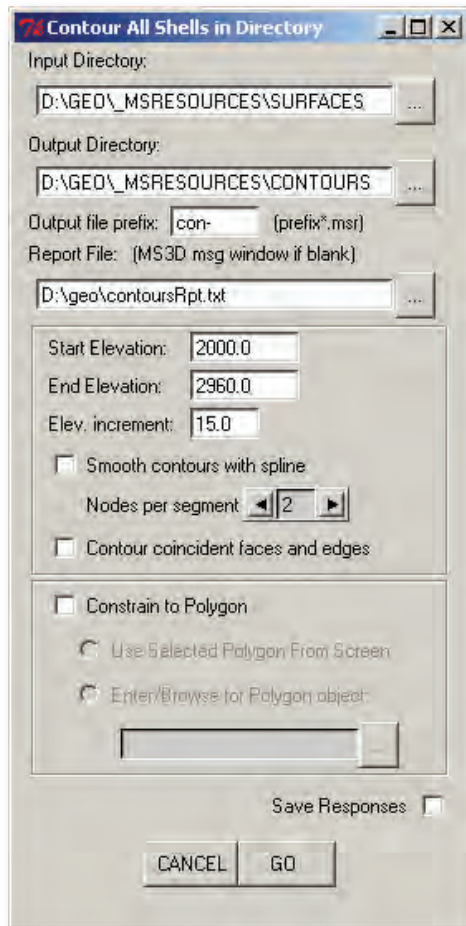


Figure 2 contoursFromDir.py GUI

Required parameters include **Input Directory**, **Output Directory**, **Output file prefix**, **Start and End Elevation**, and **Elevation Increment**.

**Input Directory** is a directory where the geometry objects containing surfaces and solids reside.

**Output Directory** is a directory where the new contour objects are created.

**Output file prefix** is used to construct the name of the object with contours for each input **msr**. If not entered, the prefix defaults to "con-". The output file name is constructed as follows: *prefix + input file name*.

**Start Elevation**, **End Elevation**, and **Elevation Increment** are used to specify elevations for contouring.

**Report File** indicates the path where an audit report of the run, with details of the processing of the objects, is written. If this field is left blank, the audit information is printed to the MineSight® 3-D Message Window.

The options **Smooth contours with spline**, **Contour coincident faces and edges**, **Constrain to Polygon** and **Save Responses** are discussed later in this article.

In the examples shown in Figures 1 and 2, the **Input Directory** is called **SURFACES** and it contains four surfaces in objects **Surf1-Surf4**. The requested **Output Directory** is called **CONTOURS**. Contours are requested for elevations from 2000 to 2960 with an interval of 15. The audit report will be written to the file **contoursRpt.txt**.

After entering the desired parameters, press **GO** to start the contouring operation. When the contouring is complete, the message **Processing Complete** is displayed at the bottom of the window.

The files with contours (**con-Surf1.msr**,... **con-Surf4.msr**) were created in the **\_msresources\CONTOURS** directory. Refresh the output directory (**CONTOURS**) in the Data Manager to make these objects available in the current session of MineSight® 3-D (Figure 3).

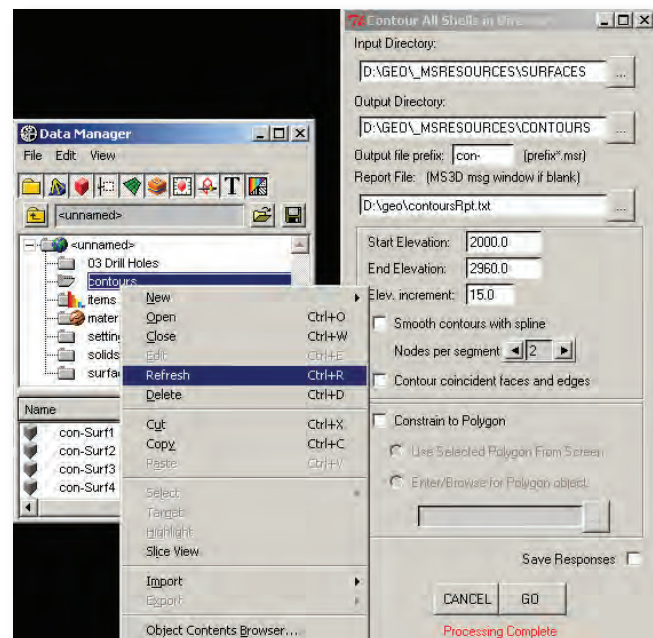


Figure 3 Refresh output directory after processing complete

(continued on page 4)

(New Scripts Available in MineSight® V3.60 for Processing Multiple Objects continued from page 3)

Figure 4 shows the results of the contouring operation, with all four new objects, **con-Surf1-con-Surf4** opened. Figure 5 displays the contents of the report/audit file, **contoursRpt.txt** opened in Notepad.

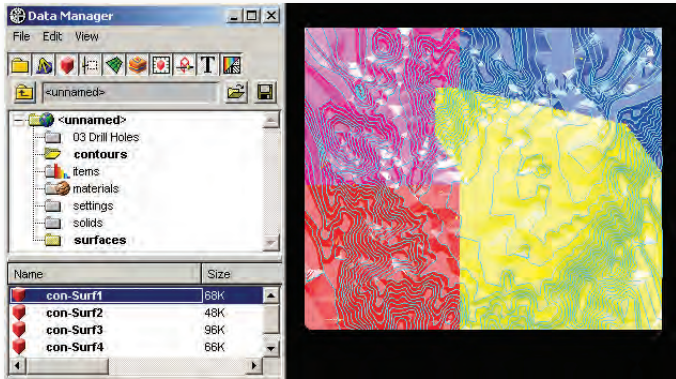


Figure 4 Resulting contours



Figure 5 Report/audit messages

Additional options are equivalent to the corresponding options in the MineSight® 3-D Contour dialog:

### Smooth contours with spline

If **Smooth contours with spline** is checked, the resulting element is smoothed by applying the Spline smoothing algorithm to the polyline/polygon for all nodes within the element. The Spline smoothing algorithm does not respect nodes while it smoothes the curve. The Nodes per segment integer specifies how many nodes to add during the smoothing process.

### Contour coincident faces and edges

Since the script generates exact contours, if a contour falls on an edge exposed in the surface, the general algorithm will bump the contour up a small amount to find a location that does not correspond to an edge. In some cases you may wish to perform an

exact contour of your data. For exact contours, check the box **Contour coincident faces and edges**, which means it is okay to contour along edges. This option may be necessary for triangulations that have been imported from other systems, or if you have created a triangulation, without the new **Minimize Flat Area** option in the triangulation dialog.

### Constrain to Polygon

To limit the resulting contours by a polygonal boundary, check **Constrain to Polygon**, then choose to use the selected polygon from the screen, or **Enter/Browse** for a geometry object that contains one polygon element. Note that if there are many surfaces to process, constraining by a polygon can be a lengthy process. The MineSight® 3-D progress meter displays progress.

Figure 6 shows the results of contouring with all additional options.

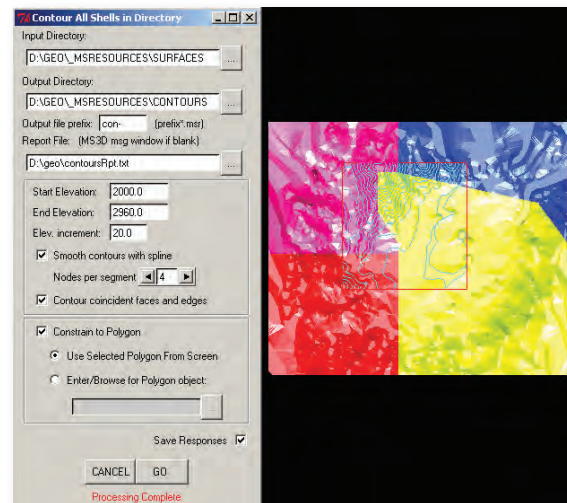


Figure 6 Contours with constrain to a square area over four different surfaces

### Save Responses

If **Save Responses** is checked, the information supplied to the GUI is saved to a configuration file named **contoursFromDir.ini**, in the same directory where the script resides. This information is used to populate the GUI the next time the script is run. Figure 7 shows the contents of the **contoursFromDir.ini** file.

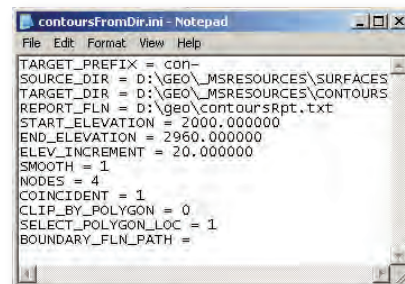


Figure 7 Example .ini file

(continued on page 5)

(New Scripts Available in MineSight® V3.60 for Processing Multiple Objects continued from page 4)

## Create Partials for all Solids in Directory – `partialsFromDir.py`

The script `partialsFromDir.py` loops through a directory inspecting all `.msr` files for solids. For each solid element found, a corresponding partials file is created. The script can be run from MineSight® 3-D, independently from the command line or from another program. It can run silently, using parameter information read from the `partialsFromDir.ini` file, or it can present an interface (GUI) where parameters are entered.

The script, `partialsFromDir.py`, uses information from a MineSight® Project Control File (PCF) to define how the partials files are created. The PCF values are used as arguments to the voxel standalone engine to compute the partials files.

If the script is run from MineSight® 3-D, the user interface always appears, as shown in Figure 8. If running from the command line or if the script is executed from another program, the GUI can be suppressed by deploying the script with an argument of one: `C:\> partialsFromDir.py 1`.

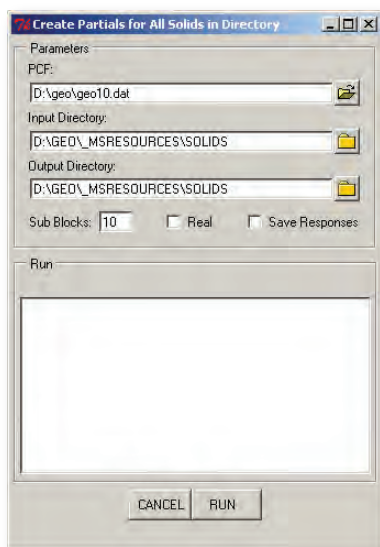


Figure 8 `partialsFromDir.py` GUI

To run the script you have to specify PCF, **Input** and **Output Directories**. In the example shown in Figure 8, the PCF file is `D:\geo\geo10.dat`, the **Input Directory** is `D:\geo\_msresources\solids` and the **Output Directory** is `D:\geo\_msresources\solids`.

The partials are always output as block percents (values from 0 to 100), and by default they are output as whole integers, but you can optionally request the real-values partials (with 3 decimals precision).

There is also an option to alter the default (10) Sub Blocks count used to compute integrated partials' values.

If the parameters are saved, the scripts can be run without a GUI. The parameters will also be used to initialize the GUI on subsequent runs. Select **Save Responses** option and the parameters will be saved to the `partialsFromDir.ini` file in your current directory.

In the example shown in Figure 9, the **Input Directory** contains two objects: `solid1` and `solid3`. The object `solid3` contains two solid elements and the object `solid1` contains one solid element.

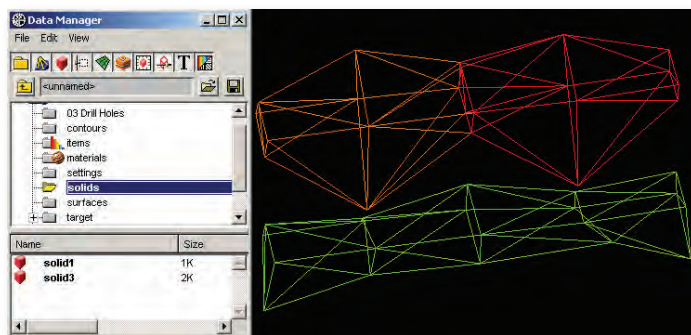


Figure 9 Two objects with three solids total

After defining the Parameters, as shown in Figure 10, and pressing **RUN**, the script creates partials files for each solid in the **Input Directory**, placing them in the **Output Directory**.

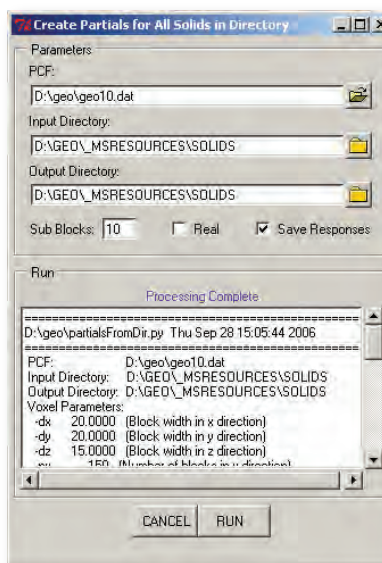


Figure 10 Processing complete

When the process completes, details (audit) of the run are displayed in the **Run** window of the GUI. Figure 11 shows an example of the audit, which includes the parameters used to create the partials

(continued on page 6)

(New Scripts Available in MineSight® V3.60 for Processing Multiple Objects continued from page 5)

and information about each operation for every solid processed.

```

=====
D:\geo\partialsFromDir.py Thu Sep 28 15:05:44 2006
=====
PCF:                D:\geo\geo10.dat
Input Directory:    D:\GEO\_MSRESOURCES\SOLIDS
Output Directory:  D:\GEO\_MSRESOURCES\SOLIDS
Voxel Parameters:
  -dx      20.00 (Block width in x direction)
  -dy      20.00 (Block width in y direction)
  -dz      15.00 (Block width in z direction)
  -nx      150 (Number of blocks in x direction)
  -ny      125 (Number of blocks in y direction)
  -nz       64 (Number of blocks in z direction)
  -xmin   1000.00 (Min model limit in x direction)
  -ymin   4000.00 (Min model limit in y direction)
  -zmin   2000.00 (Min model limit in z direction)
  -sb      10 (Sub blocks)
-----
Input geometry object:  solid1.msr
Output partials file 1: solid1-1.prt
voxel ...
Reading data from disk ... done
-----

Input geometry object:  solid3.msr
Output partials file 1: solid3-1.prt
voxel ...
Reading data from disk ... done

Output partials file 2: solid3-2.prt
voxel ...
Reading data from disk ... done

-----
Total geometry objects processed: 2
Total elements processed: 3
=====

```

Figure 11 Example audit information for completed run

A separate partial file is created for each solid in the directory. The partials file name is created as follows: (object name) + (-element number) + (.prt). For the three solids in our example resulting partials files are named **solid1-1.prt**, **solid3-1.prt**, and **solid3-2.prt**.

Figure 12 shows the contents of one of the created partials (**solid-1.prt**) files. Figure 13 shows the **.ini** file created when **Save Responses** is checked.

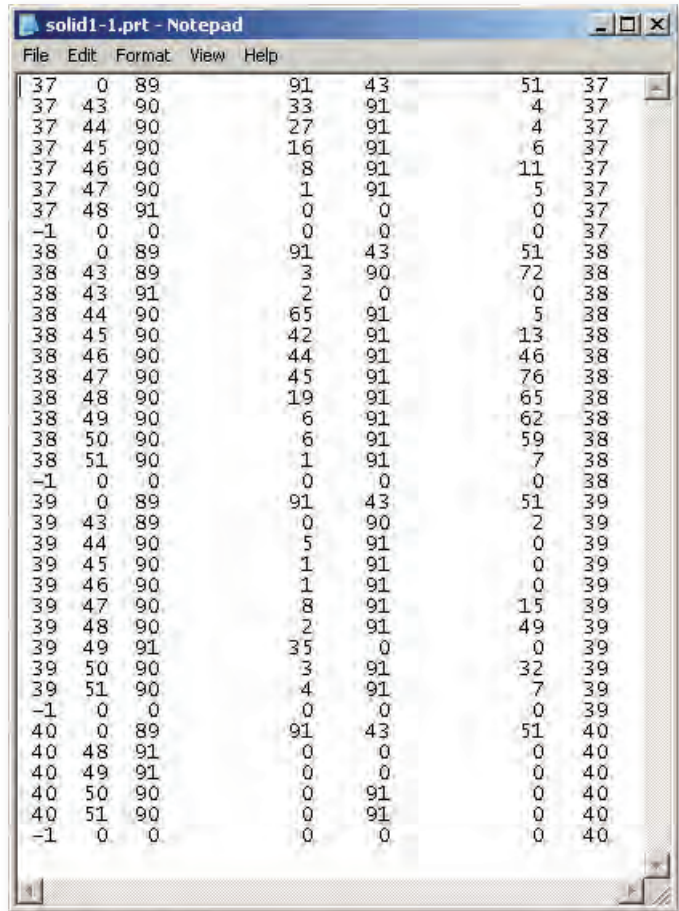


Figure 12 Example partials file content

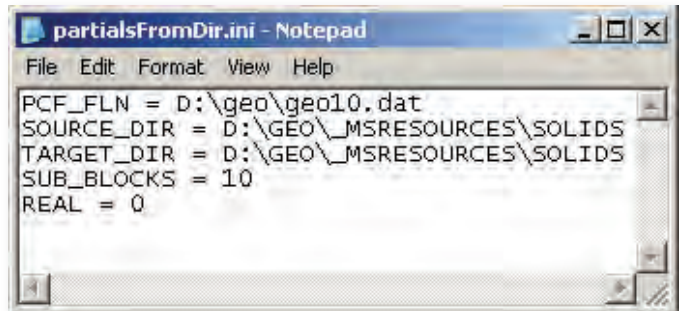


Figure 13 Example .ini file

To run the above two scripts, the Grail libraries must be in the PYTHONPATH environment variable, and the path to the standalone engines **Clip.exe** and **Voxel.exe**, must be in the PATH environmental variable.