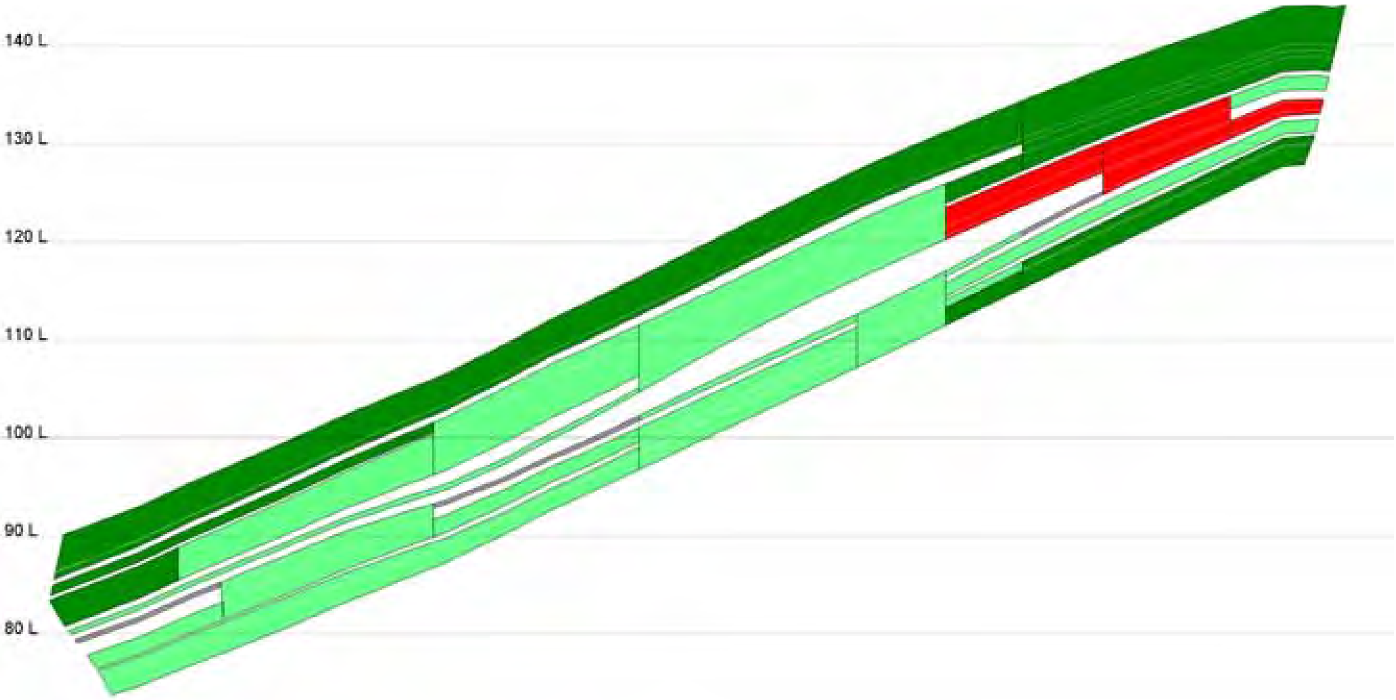
MAPTEK™ Vulcan™

ACHIEVE THE IMPOSSIBLE!

Scripting is recommended as a 'hands off' way to run Vulcan, freeing geologists and engineers to do more of the 'thinking work'.



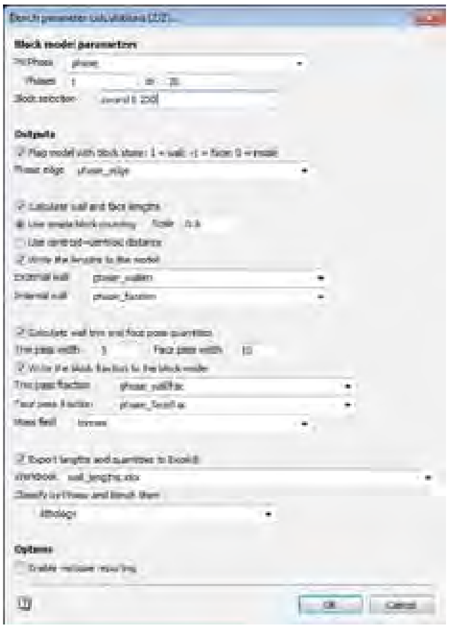
Every project has some repetitive tasks, performed daily, hourly or many times in any Maptek Vulcan™ session. So why script? Scripting can be used to re-execute frequent commands, streamlining workflow.

Scripting allows you to customise or build processes. A complex multi-step task can be made a continual process that can be run without supervision, saving minutes or even hours of your time.

A real benefit of Lava scripting is the ability to access data structures like triangulations, databases, objects or grids and manipulate them directly. Historically, data would have been exported to ASCII, manipulated and then re-imported.

Geologists and engineers are paid to solve problems. Creating an audit trail allows review and standardisation of processes for multiple users across multiple sites. Locking down parameters establishes consistency. Recalculating reserves at the end of each month is a mechanical task. Once you have worked out the process you can lock it in a script and run it confidently every month.

There are several options to choose from. The macro function in the file menu is a simple way to automate existing Vulcan menu functions. This is an easy method for creating scripts, and to learn how the scripts should be structured.



Batch parameter calculations (2.2)...

Block model parameters

Phase: phase
Phases: 1 to 15
Block selection: -zword 0 200

Outputs

☑ Flag model with block state: 1 = wall, -1 = floor, 0 = inside
Phase edge: phase_edge

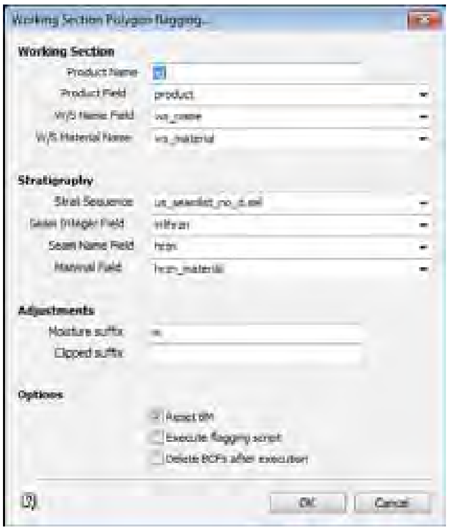
☑ Calculate wall and face lengths
● Use simple block counting Scale: 0.6
□ Use centroid-centroid distance
☑ Write the lengths to the model
External wall: phase_wallen
Internal wall: phase_facelen

☑ Calculate wall trim and face pass quantities
Trim pass width: 3 Face pass width: 10
☑ Write the block fraction to the block model
Trim pass fraction: phase_wallfrac
Face pass fraction: phase_facefrac
Mass field: tonnes

☑ Export lengths and quantities to Excel
Workbook: wall_lengths.xlsx
Classify by Phase and Bench then
lithology

Options

□ Enable verbose reporting



Working Section Polygon flagging...

Working Section

Product Name: [illegible]

Product Field: product

W/S Name Field: ws_name

W/S Material Name: ws_material

Stratigraphy

Strat Sequence: us_seamlist_no_d.sel

Seam Integer Field: intfrom

Seam Name Field: hrzn

Material Field: hrzn_material

Adjustments

Moisture suffix: m

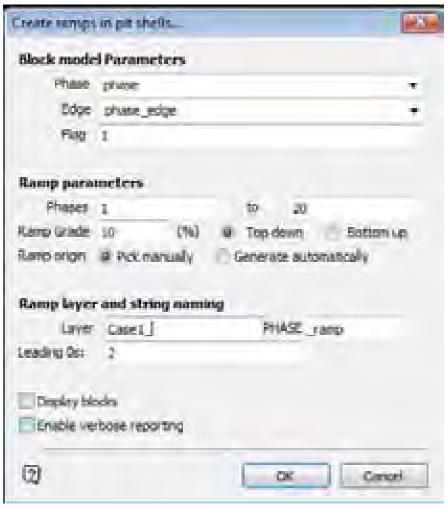
Clipped suffix:

Options

☑ Accept BM

☐ Execute flagging script

☐ Delete BCFs after execution



Create ramps in pit shells...

Block model Parameters

Phase: phase

Edge: phase_edge

Flag: 1

Ramp parameters

Phases: 1 to 20

Ramp Grade: 10 (%) ● Top down ○ Bottom up

Ramp origin: ● Pick manually ○ Generate automatically

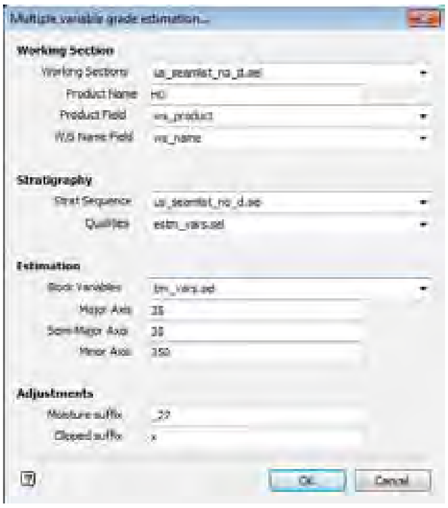
Ramp layer and string naming

Layer: Case1_ PHASE _ramp

Leading 0s: 2

☐ Display blocks

☐ Enable verbose reporting



Multiple variable grade estimation...

Working Section

Working Sections: us_seamlist_no_d.sel

Product Name: HQ

Product Field: ws_product

W/S Name Field: ws_name

Stratigraphy

Strat Sequence: us_seamlist_no_d.sel

Qualities: estm_vars.sel

Estimation

Block Variables: bm_vars.sel

Major Axis: 35

Semi-Major Axis: 35

Minor Axis: 350

Adjustments

Moisture suffix: _22

Clipped suffix: x

Macros record menu selections that a user makes. Once you understand how macros work, you can start passing variables to them. They are a really good place to start.

A range of external programs can be run through the command shell. This is perfect for processes that may take a couple of hours. Start them up and return later for the answer. You can build very complex processes, passing arguments between scripts and programs, doing loops and testing variables. Most of the time this is run at the GUI or through the command shell. This requires more programming skill than just using the macro function.

Lava or Perl scripting appears under the Vulcan file menu, providing direct access to all of the Vulcan data structures. It lets you control the graphics window and create your own panels to get user input.

Lava scripts can be executed from the file menu or run externally. Lava can be used to bring together all of the existing functions in Vulcan, and to build custom processes on top of that. The advanced programming skills are not that difficult to learn.

Most scripting will be a combination of all three options. The macro function allows you to run all of the existing menu functions. The T-Shell scripting can pull together all of the external programs, and you can also run Lava scripts from T-Shell script. Similarly you can use the power of Perl or Lava to run external programs.

Developing a run of mine working section block model for a multi-seam coal deposit is one example where scripts are useful.

A T-Shell script takes command line arguments to execute a series of Perl scripts that read parameters from a spec file, build multiple GDCalc scripts and run the command line version of GDCalc. The script checks and executes all of the files.

The output includes in-situ and ROM grids for each seam and potential working section combination, Envisage polygons describing the extent of each working section, and a logfile for checking successful execution of GDCalc. Results can be visually checked at this stage.

The second step is to flag those working sections into the block model. This script builds solid triangulations for each working section polygon, a BCF to flag the block model within the respective solid for each working section, a T-Shell script to execute the block model flagging, and then executes the flagging.

The third script creates block estimation files and mapfiles from the ROM working section grids and a T-Shell script to run the estimations, and then runs the estimations.

In summary, scripting can create custom functionality, and perform tasks not directly available within current Vulcan tools.

Thanks to Andrew Vidale
Principal, AVCS

Presented at AUSTRALIA 2012