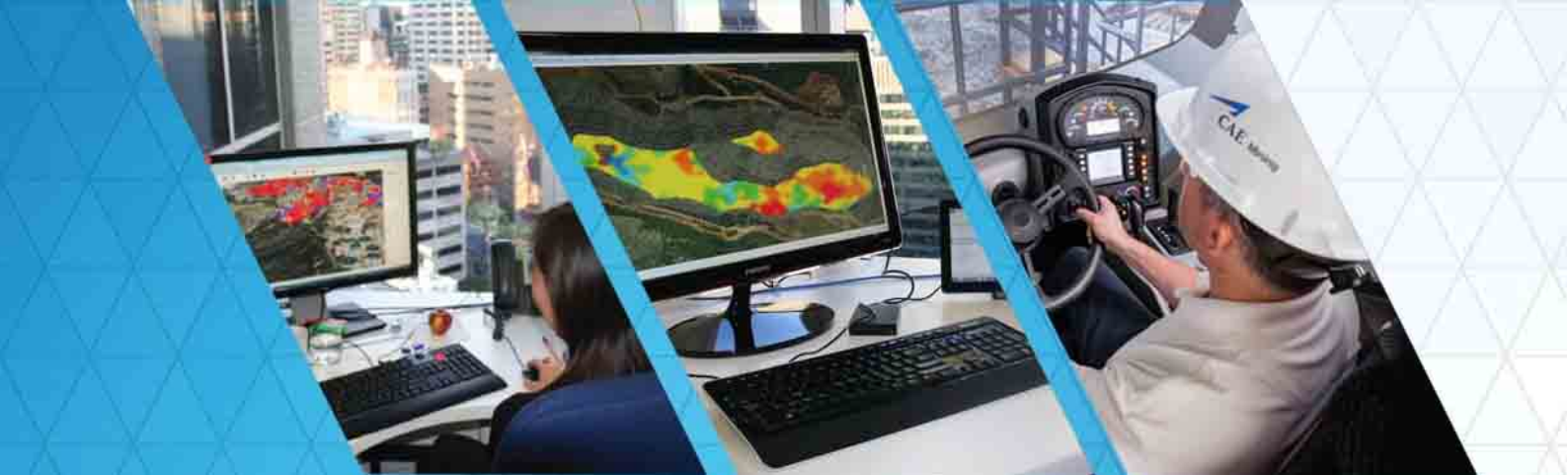


Studio 3



Wireframing User Guide



This documentation is confidential and may not be disclosed to third parties without the prior written permission of CAE Mining Corporate Limited.

EXECUTIVE SUMMARY

Studio 3 provides a fully comprehensive and interactive three dimensional system for creating, modifying, merging, intersecting, displaying and evaluating both solid models and Digital Terrain Models ("DTMs").

Strings may be linked using 'point and click' selection methods and a choice of linking algorithms is included. The optimized method frequently allows even highly complex strings to be linked automatically without point tagging; however, tagging is also provided for greater control over linking.

DTMs may be constructed from a set of points and strings, and linking constrained to lie within user-defined limits.

Studio 3 has the facility to turn wireframe models into cell and sub-cell models in a single pass with the amount of cell sub-division under user control. This unifies the concepts of wireframe and cell models, allowing varying grades within seams and orebodies to be represented in as much detail as required.

In addition, two separate groups of wireframing functions are provided; Boolean and plane functions. Boolean functions allow you to perform wireframe-related tasks based on the interaction of two distinct wireframe objects, and plane operations allow you to modify the contents of a wireframe database by affecting it with either the viewplane or other non-wireframe objects (such as string, for example). New objects are created as required as a result of these operations.

Contents

1	Overview	2
	Purpose of this document	2
	Prerequisites	2
	Acronyms and Abbreviations	2
	More information	2
2	Introduction to Wireframes	3
	Querying Triangle Information	6
3	Displaying Wireframes	9
	Studio 3 View Hierarchy	9
4	Selecting Wireframe Data	12
	Wireframe Selection Overview	12
	Wireframe Selection Options	12
5	Boolean and Plane Functions	15
	Boolean and Plane Functions	15
	Boolean Functions Overview	15
	<i>Studio 3's Boolean Commands</i>	15
	<i>Boolean Operation Tolerances</i>	16
	Boolean Functions in Studio 3	16
	<i>Union</i>	16
	<i>Intersection</i>	17
	<i>Difference</i>	18
	<i>Extract Separate</i>	18
	<i>Strings from Intersections</i>	21
	<i>Solid Hull</i>	22
	<i>Update DTM</i>	23
	Plane Functions Overview	23
	<i>Split</i>	24
	<i>Multiple Split</i>	24
	<i>Split by String</i>	26
	<i>Section</i>	27
	<i>Multiple Section</i>	27
6	Wireframes from Linked Strings	29
	String Linking Settings	29
	<i>String Linking Control</i>	29
	<i>String Linking methods</i>	30
	Understanding "Tag" Strings	31
	Linking Commands	32
	<i>Link Strings</i>	33
	<i>Link Boundary</i>	33
7	Digital Terrain Models	35
	DTM Inner and Outer Limits	35
	Understanding DTM Settings	36
	Creating the Digital Terrain Model	37
	Creating a DTM using the SURTRI Command	37
	<i>Boundaries</i>	37
8	Wireframe Verification	38
	Using the Verify Command	38

9	Manipulating Existing Wireframes	40
	The Current Wireframe Sub-menu	40
	<i>New Triangle</i>	40
	<i>Unlink Triangle</i>	41
	<i>Move Wireframe Point</i>	41
	<i>Insert Wireframe Point</i>	41
	<i>Delete Wireframe Point</i>	41
	<i>Show Wireframe Slice</i>	42
	Decimating Wireframes	42
	<i>Wireframe Decimation Guidelines</i>	42
	<i>The Decimate Wireframe Dialog</i>	42
10	Calculating Wireframe Volumes	45
	Cut and Fill Volumes	45
11	Other Datamine Wireframing Processes	48
	Create a DTM	48
	Add Two Wireframe Files	48
	Calculate Centre of Triangle	48
	Extend Wireframe Along a Trend	49
	Calculate Wireframe Volume	49
	Evaluate Model v Wireframe	49
	<i>Input and Output Models</i>	49
	<i>Balancing Volumes</i>	50
	<i>Densities</i>	50
	<i>Passes through the Model File</i>	50
	Select Data Using Wireframe	50
	Project String to Wireframe	51
	Strings from Wireframe Section	52
12	Creating Wireframes from Points	53
	The Wireframe from Points Dynamic Dialog	54
	<i>Data Resolution and Processing Times</i>	54
	<i>Smoothing</i>	54
	<i>Wireframe Preview</i>	54
	<i>Dialog Structure</i>	55
	Configuring Input and Output Files	55
	Reducing Noise and Managing Holes in your Data	56
	Managing Data Holes	57
	Understanding Noise Reduction	59
	Selecting a Mesh Generation Method	61
	<i>Available Methods and Parameters</i>	61
	Managing Vertices and Triangles	63
	Wireframe from Points – Worked Example	64
	<i>First Run – Coarse Model with Low Resolution Settings</i>	65
	<i>Second Run – High-resolution Model with Hole Removal and Smoothing</i>	66
13	Visualizing Wireframes	68
	Viewing Wireframes in the Design window	68
	Viewing Wireframes in the Visualizer Window	69
	Viewing Wireframes in the VR Window	70
	“Registering” an Image	71
	<i>The Image Registration Toolbar</i>	72
	<i>The Image Preview Pane</i>	73
	<i>The Alignment Table</i>	73
	<i>Guidelines for Best Results</i>	74
	<i>Basic Texture Image Alignment - Procedure</i>	75
	Worked Example 1 – Aligning a Hand-drawn Image to a Surface	75
	<i>Sample Data</i>	76

1 OVERVIEW

Purpose of this document

This document intends to outline the most commonly-used wireframing tools found in **Studio 3**. These commands will be explained in context, with examples wherever appropriate. General procedures are available for future reference. This document is intended as a supplementary support document for all Studio 3 users.

Prerequisites

This document assumes that a basic working knowledge of the Studio 3 object-based approach to modelling exists. It also assumes that the reader has some understanding of digitizing points and strings, and a familiarization with the **Design** window's layout and customization facilities.

For more information on these subjects, refer to *More Information*, below.

Acronyms and Abbreviations

This document uses the following acronyms:

Acronym	Description
WF	Wireframe
DM	Datamine
DTM	Digital Terrain Model
S3	Studio 3
DM	Datamine
UG	Underground

More information

The following resources provide useful information on the subject of Wireframing in Studio 3, and associated tasks:

- Your Studio 3 online Help (**Help | Contents**)
- The Studio 3 Introductory Tutorial (**Help | Tutorials | Introductory Tutorial**)



Wireframes are an essential tool to any CAD engineer.

Essentially, Wireframes are representations of interconnected points in 3D space, where each linkage gives rise to an 'edge' and each closed and distinct loops of edges create 'faces'.

Wireframe data is used to represent an almost infinite variety of virtual objects. Wireframe data can be formed from simple building blocks, known as *primitives* such as spheres or cubes

and can also be generated from other source data such as *point cloud* data (a series of disparate points floating in 3D space) or *splines* - contours created to simulate a surface. Each method of creating wireframes has its own associated variants, for example, splines come in various guises – parametric splines, Bezier curves, non-uniform-rationalized-B-splines ('NURBS'), univariate, polynomial and so on.

The result of the wireframing method is a *mesh* representing a surface, or in the case of *closed* wireframe objects, a volume. When representing volumes, wireframes are commonly referred to as 'solid'. In Studio 3, these objects are also referred to as 'wireframe volumes'.

There are many software packages specializing in one or more of these areas. **Studio 3** represents the global market leader in wireframe modelling for geotechnical modelling purposes.

Studio 3 has an extensive repertoire of commands and functions for creating and manipulating wireframe models. This guide explains many of the principles of wireframing and provides an overview, with examples, of much of this functionality.

Wireframes are used extensively in many aspects of the Datamine Solution Footprint, including open pit design, underground design, scheduling, short and long-term mine planning, resource evaluation and immersive virtual reality visualization, to name but a few.

Solid wireframes can be used to represent a 3D volume, such as:

- A zone of high grade mineralization
- An orebody
- A particular lithology

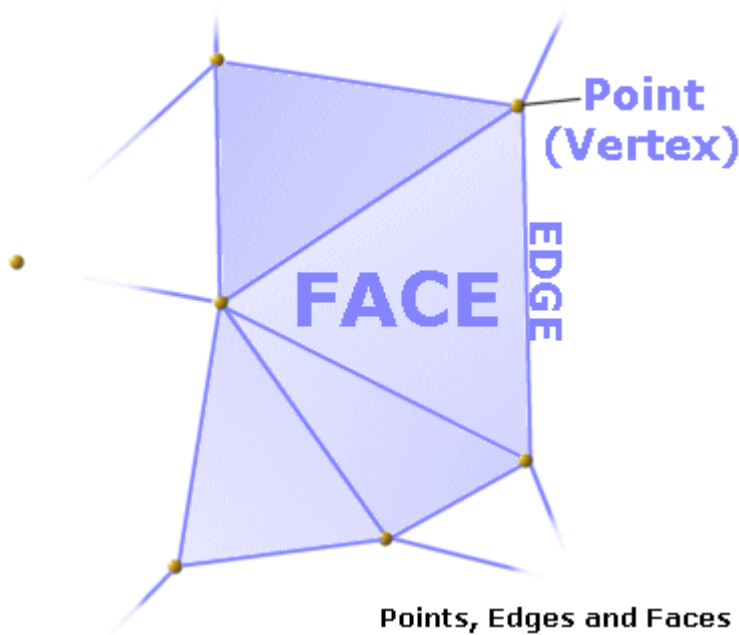
From these wireframes, volumes and tonnages can be calculated, which are useful for determining resources and reserves.

Wireframes can also be used to represent surfaces, such as:

- Topography (Digital Terrain Model)
- Pit geometry

To create a wireframe a series of strings is created and then these strings are linked together to form the 3D surface outline. Datamine wireframe data is stored in two related files, a triangle file and a points file.

The basic structure of a wireframe (regardless of the way this data is displayed on screen) is simple: a series of points (vertices) interconnected by lines (edges) creating triangles (faces). The arrangement and resolution of this structure defines the 'geometry' that is being portrayed – the visible object. The more data you have in your wireframe, the greater its resolution (and file size) and the greater the potential for more organic and freeform surfaces.



Studio 3 wireframe files contain more information than just points, lines and faces. This combination of points and triangles files will always have a minimum number of data columns or fields to define other important properties, such as how points are linked, and which colors are to be used for rendering edges etc.

If the thought of a database frightens you, don't despair – the concept is, in reality, a very simple one. A Datamine file can be thought of as a table of figures, with multiple columns to represent each type of information.

Data can be stored as numbers (numeric) or strings (alphanumeric) and the relationship(s) between one item (row) of data and another can also be defined by cross-referencing, creating groups of related information. This information is used by Studio 3 to draw the basic geometry of the wireframe object (note that as soon as a 'file' is loaded into memory, it is referred to as an 'object' – this nomenclature will be used throughout this manual).

To give you a clearer picture of how points and triangles files are related (you can't define a wireframe without both files present, although in most cases, a points file is loaded automatically providing triangle and point file names are similar), the golden rule is that all spatial coordinate information is defined in a *points* file, and the relationships between these points (i.e. 'how' the wireframe is constructed) is defined in the *triangles* file.

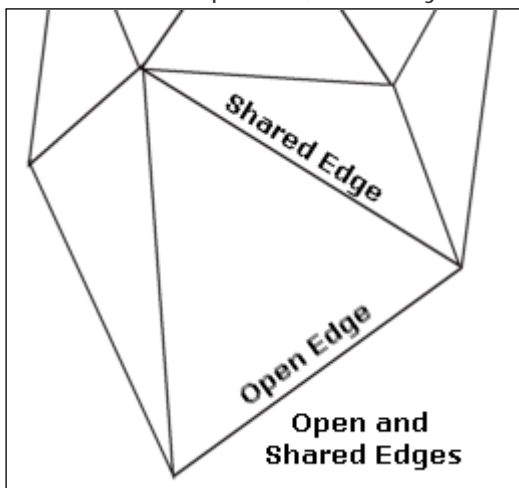
The following table lists all of the data columns that are included in a wireframe data file (and when the file is loaded, the resulting data object). Don't be put off by unfamiliar field names, or concepts that may not be clear at this stage as all data fields will be referenced throughout the remaining manual, and are also explained in detail in your Studio 3 online Help.

FILE	FIELD	DESCRIPTION
POINT FILE	GROUP	Sometimes, data in a wireframe file is grouped into distinct zones. This parameter records a unique index for each group.
	XP	X, Y and Z coordinates of the triangle point defined by the PID value.

	YP ZP	
	PID	The point ID number (as referenced by PID1 etc in the triangle file).
TRIANGLE FILE	TRIANGLE	Triangle number
	PID1	First triangle point ID number
	PID2	Second triangle point ID number
	PID3	Third triangle point ID number
	GROUP	Wireframe group ID number
	SURFACE	Wireframe surface ID number
	LINK	Link number
	NORMAL-X NORMAL-Y NORMAL-Z	Stores the values which define the orientation of the normal of the triangle in the X, Y and Z directions.
	TRE1ADJ TRE2ADJ TRE3ADJ	Stores the 'adjacency data', which define open and shared triangle edges.
	COLOUR	The colour value of the current triangle, according to the Datamine colour index.

A few of the terms in the above table haven't been explained yet, however, the concept that is important is how the point and triangle files are interrelated by their sharing of PID values. Each row in the triangle file (i.e. each triangle) has 3 PID numbers. Each of these represents a point of a triangle, and all are connected.

What is also important, and may not be clear from the table above, is how each triangle is related to other 'triangles'. A triangle, under normal circumstances, will 'share' its edges with up to three other triangles. It is this 'adjacency data', defined in the triangle file with the fields TREnADJ, that shows how each triangle is linked to another. This defines whether the edge of a wireframe is connected to another triangle (shared) or not (open).



Wireframes are classified by the following fields which are present in all triangle files:

GROUP - A GROUP can consist of one or more distinct wireframes that have either the

same, or different surface numbers.

SURFACE - An individual wireframe SURFACE can be selected from within a GROUP of wireframes. This surface can be either a DTM or a solid wireframe.

LINK - Each wireframe surface consists of a number of individual links, each numbered individually.

This classification of wireframes by *GROUP* and *SURFACE* provides a means by which wireframes can be identified for operations such as combining and verifying. It also provides greater control when using the 'erase' commands; specific *GROUPs*, *SURFACEs*, *LINKs* and triangles may be selected and erased.

The wireframe *GROUP*, *SURFACE* and *LINK* numbers are stored in the wireframe points and triangles files as the fields: *GROUP*, *SURFACE* and *LINK* respectively. These values are assigned internally by the system. It is possible to select a wireframe by its *GROUP* or *SURFACE* number. Wireframes from different *GROUPs* can be selected together by using other selection methods such as "filters".

Querying Triangle Information

As well as viewing information about currently loaded wireframes, you can also analyse the properties of individual wireframe triangles. Studio 3 provides a useful command-line instruction that allows you to interactively select a wireframe triangle and see a report on that particular part of the wireframe in the **Output** control bar.



The **Output** control bar is used to record the results of many Studio 3 commands and processes, and it is a useful window to have in view when running such instructions. You can hide or show the **Output** control bar like any other Studio 3 control bar – using the **View | Customisation | Control Bars** sub-menu; select a particular description to toggle its visibility.

All Studio 3 control bars can be docked or floated as required. See your online Help for more information on how to do this.

The command to use for querying wireframe triangles is **query-triangle**. This command can be entered directly into the Studio 3 **Command Line** as shown below (the **Command** toolbar can be shown/hidden, floated/docked in the usual way, so if you can't see it - enable it via the **View | Customisation | Toolbars** menu):



The **query-triangle** command is an example of an interactive **Design** command; it requires further input to complete the task. After entering the command, select a wireframe triangle in the **Design** window and a report will be generated.

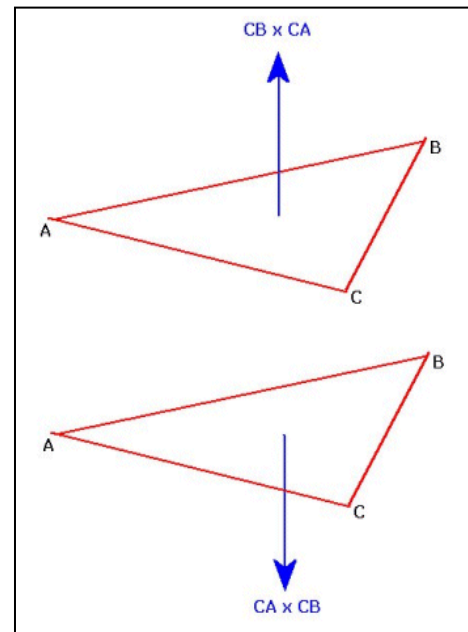
As mentioned at the start of this section, the report is shown in the **Output** control bar and can describe the following information relating to the selected triangle:

- The **X,Y and Z (spatial) coordinates** of each of the three points defining the wireframe triangle.
- The number (**index**) of the selected triangle.
- The unique identifiers for each of the **3 points** that comprise the triangle vertices.

- A **Group** number – triangles can belong to groups, possibly highlighting a particular zone of wireframe. The group number can also be used to analyse the origins of a particular section of wireframe (for example, if two wireframes are joined using a **Union** command, two groups will be present in the resulting compound object – triangles that were created from wireframe 1 become group 1, and wireframe 2 become group 2).
- A **Surface** number – although not a standard property (it is generated by other wireframing processes), a surface number may be present to show the 'side' of the wireframe that a triangle belongs to.
- Values defining the '**normal**' of the wireframe triangle. This description doesn't belong in this section, strictly speaking, as it is not a standard property of a wireframe file. However normal data is added during many Studio 3 processes, such as Boolean processes (discussed later in this guide), and it is useful to have an understanding of what a normal is at this stage.

A 'normal' represents a line projected perpendicularly from the theoretical surface of a wireframe triangle to define its overall 'direction'. If you are interested in a fuller description of the concept of wireframe normals, read on, otherwise skip to the next point.

A *normal* is a vector that is perpendicular to the surface of the triangle. The normal vector is computed as the "cross product" of two vectors that make up the



side of the triangle.

For example, if you have a triangle defined by points A, B, and C, you could create the normal by multiplying $AB \times AC$, $BC \times BA$, or $CB \times CA$. All three ways will result in the same normal. However, the right-hand rule still applies. $AB \times AC$ will result in a normal in the opposite direction from $AC \times AB$. Each position in the mesh should have a normal assigned to it, and a position can only have one normal.

- '**Adjacency data**'. For a wireframe to be constructed, it is not only important to define the size and orientation of triangles, it is also vital that data is supplied relating to how they fit together. Three items of adjacency data are present in every Studio 3 wireframe to denote which particular triangle (if any) is joined to a particular edge of another triangle.
- The **color** of the triangle is also recorded (as an index number).
- Other properties can also be present; some of these may be generated by in-built Studio 3 processes - or processes in other applications for imported data - automatically (*LINK*, *BLOCKID*, *ZONE* etc.) and you may also find user-defined entries in a table, adding using the **Add Attribute** command. For now, an appreciation of the more common fields listed above is a sufficient basis for understanding the information that follows:

Below is an example a report generated by the **query-triangle** command, for a simple loaded wireframe:

```
X           Y           Z
6010.000000  5027.750000    14.7504000
6010.000000  5005.5630000   16.9279300
6035.000000  5014.9380000  -10.1383000

TRIANGLE : 390.0
PID1     : 341.0
PID2     : 342.0
PID3     : 392.0
GROUP    : 1.0
SURFACE  : 1.0
LINK     : 5.0
NORMAL-X : -0.72111398
NORMAL-Y : -0.067671
NORMAL-Z : -0.68950403
TRE1&DJ : -
TRE2&DJ : -
TRE3&DJ : -
TCOLOUR  : 2.0
COLOUR   : 5.0
BLOCKID  : -
ZONE     : 1.0
```



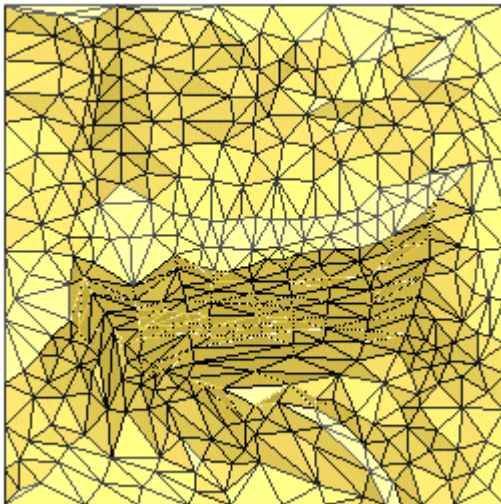
You can also access the **query-triangle** command from the Studio 3 menu system: select **Design | Query | Triangles**.

3 DISPLAYING WIREFRAMES

A points file is the basis for a wireframe. Each point is joined to several others to form a series of triangles that form the surface of a wireframe. Wireframe data can be viewed in the **Design**, **Visualizer**, **VR** and **Plots** windows.

The way the triangles are formed can be controlled; the method used is dependent on the wireframe purpose and user preference. In addition, the resulting object can be displayed in a variety of different formats, using a series of display overlays based on the same underlying wireframe data. You can display a wireframe using the standard rendering mode, or the more advanced 3D rendering modes – meaning a wireframe can be displayed as a ‘mesh’, a solid shaded object, a smooth shaded object, and that’s just the options available to the **Design** window; the **Visualizer** and **VR** windows allow for even more control over the way 3D data can be displayed.

For more information on viewing wireframe data in the various Studio 3 windows, see “Visualizing Wireframes” at the end of this guide.



Wireframes can be shown as *points*, *lines* or an *intersection* (with the viewplane or plot section). As with other Datamine data types (points, strings etc.), you can control how surfaces are treated using legends.

This section explains some of the techniques that can be used to manipulate the display of wireframe data, without affecting the underlying ‘object’.

There is no definitive way to view wireframe data, and it is likely that you will use a variety of techniques when working with data of this type. As each of the Studio 3 windows displays data in a slightly different way, this chapter is broken down in to corresponding sub-sections, starting with the **Design** window.

Studio 3 View Hierarchy

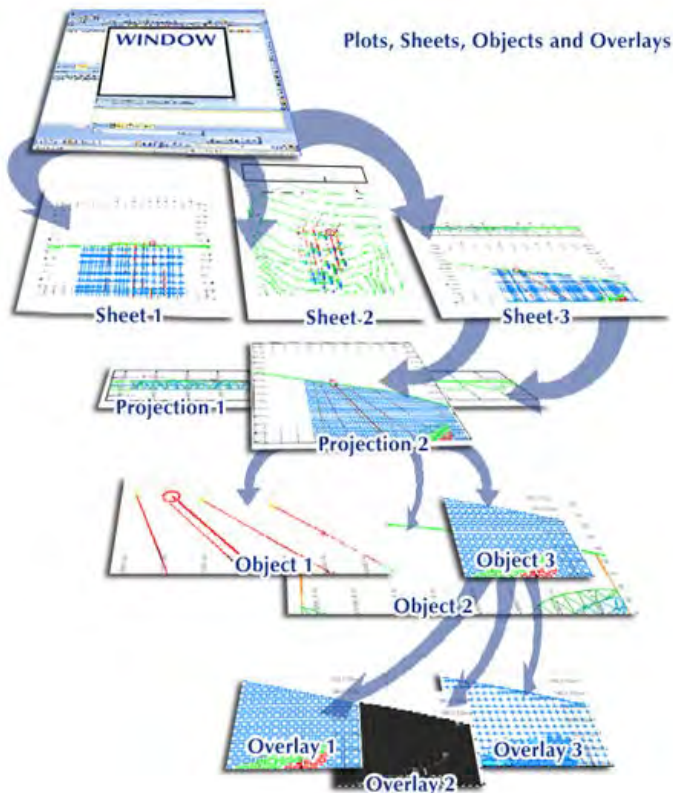
Before investigating Studio 3’s wireframe display functions, you will need to understand the Studio 3 *view hierarchy*. This subject is covered in detail in other areas (for example, the Studio online Help file has a series of topics on data display) but as a quick overview; each of the graphic windows in Studio 3 supports a combination of *sheets*, *projections* and *overlays*.

- A sheet represents a virtual ‘page’ of information – in a way, this is synonymous with the term ‘screen’. For example, the **Design** window will allow you to update the contents, alignment and formatting of the data in memory using a host of commands, but the screen, or ‘viewport’ is always the same – the **Plots** window, in comparison supports multiple viewports, accessed by tabs at the bottom of the screen. In this respect, the **Design** window has a single, dynamic, sheet and the **Plots** window has multiple sheets.
- Within each sheet there is the possibility of one or more *projections*. A projection is, in basic terms, a view of the data within a single sheet. The **Design** window supports a single view-per-screen only, whereas the **Plots** window allows you to display data in a variety of different views on the same plot sheet.

- For each projection of data, that is, each representation of a data object (or objects), an overlay is used to dictate how the data is displayed. Think of an overlay as an acetate overlay, applied over data objects to change the way they appear (although, in reality, an overlay does a lot more than this).

In some data windows where the display can be formatted (**Design, Plots, Logs**), it is possible to show the same object in more than one way simultaneously – in other words, these windows support multiple overlays. Some 3D data screens don't support direct object formatting (such as the **Visualizer** window), in other words, they do not support an overlay concept.

There are multitude of ways to show this concept diagrammatically. It is worthwhile spending a few moments getting to understand the viewing hierarchy concept before continuing, as references to sheets, projections and overlays are made throughout many Datamine Help files, and once you get used to managing these aspects of data display effectively, you can manage your projects more efficiently.



The schematic on the left shows an overview of the viewing hierarchy for the **Plots** window. This is the most complicated display window in Studio 3 in this respect.

In comparison, for example, the **Tables** and **Reports** window have the capacity for several sheets (tabbed report pages), but each sheet can only show one view of data (one projection) and there is no facility for applying overlays (although formatting of data is easy, it is on a once-per-column basis).

Wireframe data can be represented in any of the Studio 3 windows in a variety of formats, from a simple data table showing the contents of a data

table in the **Tables** window to a multiple-sheet, multiple-projection, multiple-object and multiple-overlay scenario in the **Plots** window, however, it is the **Design, Plots, Visualizer** and **VR** windows that are able to display the actual 'shape' of a wireframe file.

The table that follows shows how the Studio 3 view hierarchy is structured:

Window Name	Sheet	Projection	Overlay
Design	1 sheet	1 dynamic projection	Multiple
Plots	Multiple	Multiple	Multiple
Visualizer	1 sheet	1 dynamic projection	None
Tables	Multiple	1 projection	None
Reports	Multiple	1 Projection	None

Logs	Multiple	Multiple	Multiple
VR	1 sheet	1 dynamic projection	None
Files	1 sheet	N/A	None

For more information on viewing wireframe data, please refer to "Visualizing Wireframes" at the end of this guide.

4 SELECTING WIREFRAME DATA

Wireframe Selection Overview

Studio 3's Boolean and plane wireframe operations work with data loaded into memory, where each wireframe exists as a discrete 3D object. By default, selection is object-based but the more traditional selection methods can also be used by changing a project's settings.

Up to now, wireframe data has been referred to as either a *file* (i.e. it exists as a physical file on disk, and is part of the current project but is not loaded into memory) or an *object* (the resulting data in memory after a file has been loaded). Although each wireframe object exists as a separate entry in the **Loaded Data** control bar, each individual object supports the concept of individual 'groups' of data. These groups can reflect the different grade values of an orebody wireframe, the individual components resulting from a Boolean operation (explained later in this guide) or even disparate orebody fragments.

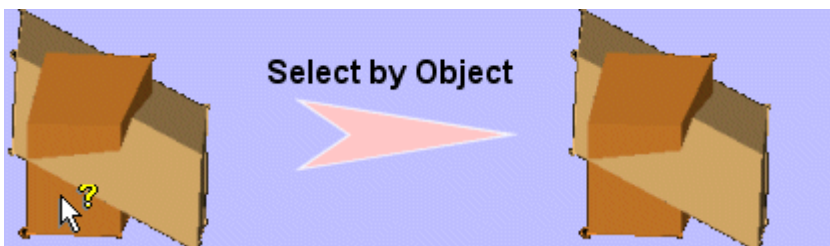
Studio 3 wireframes support multiple surfaces per object (the simplest example would be a cube wireframe with 6 surfaces). This concept is also important when selecting wireframe data.

The **Data Selection** dialog is opened when the pick tool is used during a wireframing operation (e.g. **Union**, **Split**, **Rotate** etc.), but only if no default method for selecting wireframes has been set in the **Project Settings** dialog (**File | Settings**). If a default method has been selected, this method will be used to attempt to select data.

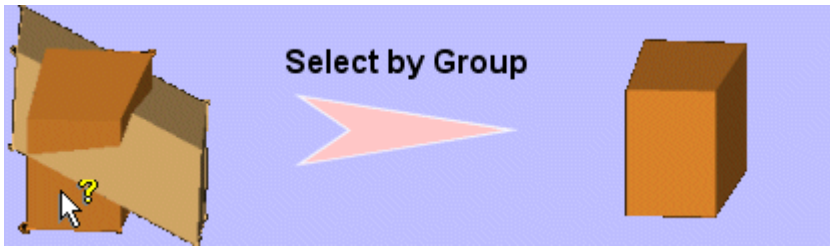
Wireframe Selection Options

As well as selecting data according to the respective object, group or surface, you can also select in relation to a particular attribute, field and filter:

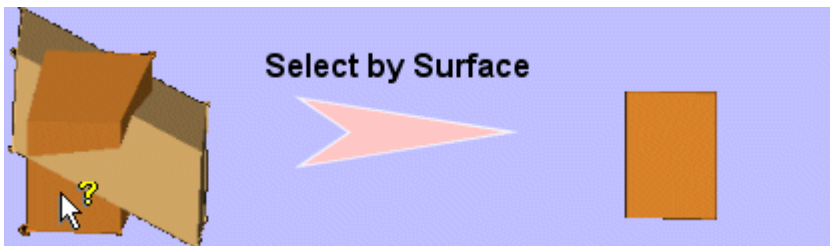
- **Select by Object:** controls the selection of wireframe data by object names. This will cause selection of wireframe data by prompting for wireframe point and triangle filenames. This command selects the whole object and places the object name in the *Object Name* drop down list of the dialog in use at the time.



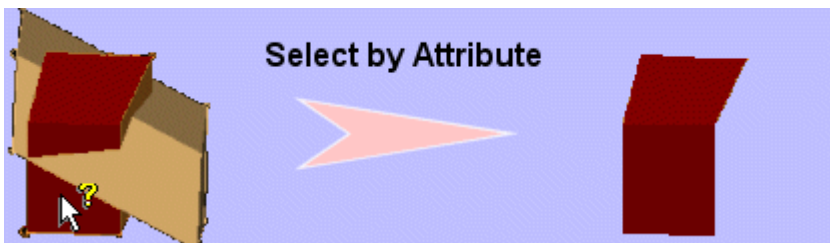
- **Select by Group:** controls the selection of wireframe data by the picked wireframe group. This option selects the part of the wireframe object bearing the group number of the part clicked (e.g. **GROUP=2**) and creates a temporary wireframe object which is reported in the **Object Name** drop down box as the whole object name - split by group (e.g. "**red_cube_tr/red_cube_pt (wireframe) - Split (GROUP=2)**"). Grouped wireframe data often results from the merger of separate wireframe files into a conjoined object (thus preserving the origins of the original data) but can also be added by other means, including manually administering the object's underlying database table.



- **Select by Surface:** controls the selection of wireframe data by picked wireframe surface. Selects the part of the wireframe object bearing the surface number of the part clicked (e.g. SURFACE=3) and creates a temporary wireframe object which is reported in the **Object Name** drop down box as the whole object name - split by surface (e.g. "red_cube_tr/red_cube_pt (wireframe) - Split (SURFACE=3)").



- **Select by Attribute:** controls the selection of wireframe data by user attributes. selects a part of the wireframe which shares all the same attribute values as the part clicked. (e.g. (COLOUR=10) AND (BLOCKID=2)) and creates a temporary wireframe object which is reported in the *Object Name* drop down box as the whole object name - split by attributes (e.g. "red_cube_tr/red_cube_pt (wireframe) - Split (COLOUR=10) AND (BLOCKID=2)").



- **Select by Field:** controls the selection of wireframe data by specifying a field in the selected file. The fields available in the point file are *GROUP*, *PID*, *XP*, *YP* and *ZP*. The fields available in the triangle files are *GROUP*, *SURFACE*, *LINK*, *TRE1ADJ*, *TRE2ADJ*, *TCOLOUR*, *COLOUR*, *NORMAL-X*, *NORMAL-Y*, *NORMAL-Z* and any other user-defined attributes.

This selection method works similarly to *Select By Surface*, or *Select By Group*, but allows the user to define the *Field* name being used (e.g. Select by *COLOUR*). If using default selection mode, the *Field* name can be defined in the **Project Options** dialog (*Wireframe Properties* tab). When the selection mode is made on a case-by-case basis, the standard **Data Selection** dialog allows the field to be selected from the fields present in the currently selected object.

- **Select by Filter:** controls the selection of wireframe data by user defined filters. This option relies on a filter expression, defined in the **Data Expression Builder**, allowing you to use standard comparative filter expressions to determine the data to be selected. The wireframe group and surface numbers are ignored on input, and new group and surface numbers will be generated on output.

An easy way to build a custom filter is to choose the basic selection method which most closely matches your desired custom selection, and choose the **Custom** option. This will load a filter into the *Selection Method* window which can be modified to meet your requirements.

You can choose a default wireframe selection method by:

- accessing the options menu from the **Wireframes | Wireframe Settings** menu entry
- selecting **File | Settings | Wireframing**.

The **Project Settings** dialog (*Wireframing* tab) is key when defining how wireframe data is selected. The correct data selection format is important during Boolean operations, as it is possible to set a default selection method, and attempt to select data that does not exist. For example, if the current default selection method is set to *By Field*, and the current data selected does not contain appropriate field values, no data will be selected. In this situation, the message "The current wireframe selection method is By Field, but no field name has been specified. Access **File | Settings | Wireframing** to change your selection method or specify a field".

In the event that a field *has* been specified (including by *Select by Surface*, etc.), and this field is not present, the following message is shown: "The current wireframe selection method is by field (*fieldname*), but this cannot be found in the selected object. Access **File | Settings | Wireframing** to change your selection method."

In most cases, wireframe data can often be selected prior to a subsequent command to pre-populate a functional dialog, or alternatively, there may be an option to interactively select data during a wireframe manipulation command.

Several wireframe commands require the selection of a wireframe; they all honour the chosen selection method, providing the *Always use default selection method* check box is selected on the *Wireframing* tab of the **Project Settings** dialog. The commands are:

- **Wireframes | Boolean Operations** (e.g. Union, Intersection etc);
- **Wireframes | Verify**
- **Wireframes | Tools | Edit Attributes**
- **Wireframe volumes**
- **Wireframe evaluation**

5 BOOLEAN AND PLANE FUNCTIONS

Boolean and Plane Functions

Boolean and Plane functions are separate categories of wireframe manipulation commands.

- Boolean functions are performed with two wireframe objects in memory. The interaction of these objects (and the command selected) will determine the end result(s).
- Plane functions are performed on a single wireframe object only, although they also involve the interaction of a wireframe and a non-wireframe entity, such as a string object, or the viewing plane, for example.

This section outlines the functions available for both types of command.

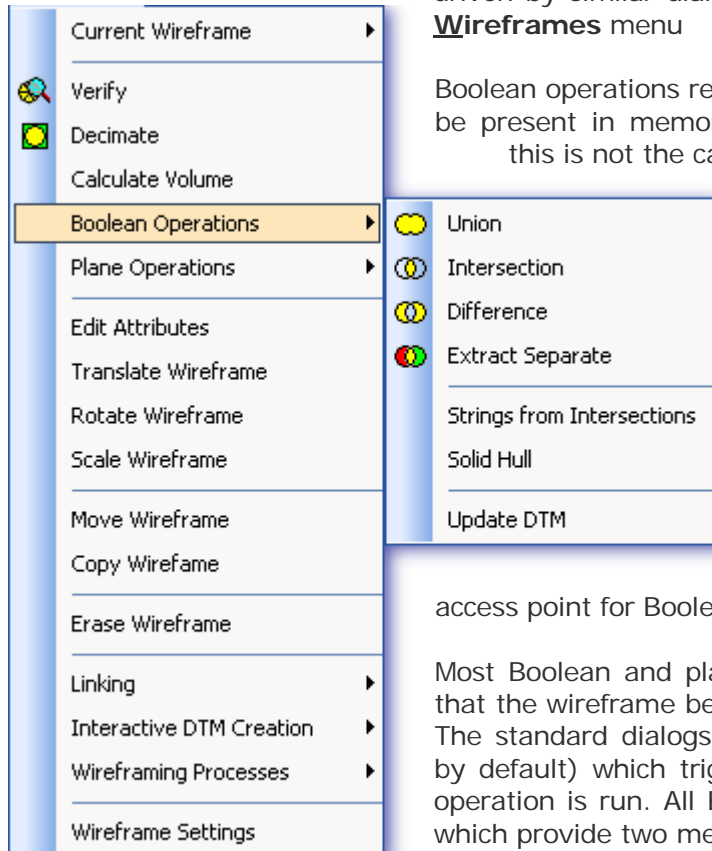
Boolean Functions Overview

The Oxford English Dictionary defines the term 'boolean' as:

Boolean /'bu:li@n/

- **adj.** denoting a system of algebraic notation used to represent logical propositions by means of the binary digits 0 (false) and 1 (true), especially in computing and electronics.
- **n. Computing** a binary variable with these possible values.
 - **ORIGIN** C19: from the name of the Engl. mathematician G. *Boole* + **-an**.

Studio 3 has a comprehensive set of commands for performing boolean and other related operations on wireframes, such as: unions, intersections and splits. All the operations are driven by similar dialogs and can be accessed through the **Wireframes** menu

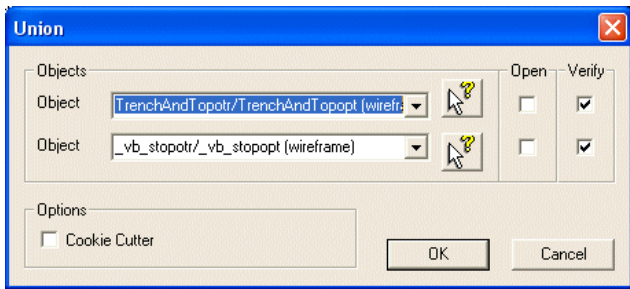


Boolean operations require at least two wireframe objects to be present in memory before a command is accessed. If this is not the case, you will be warned of the situation and processing will cease. All the operations described in this chapter require the data to be loaded into memory first. Wireframe selection is, by default, object-based as each wireframe loaded into memory exists as a separate object.

Studio 3's Boolean Commands

The Wireframes drop down menu in the **Design** window is the access point for Boolean functions.

Most Boolean and plane wireframe operations recommend that the wireframe be verified before running the operation. The standard dialogs contain *Verify* check boxes (selected by default) which trigger wireframe verification before the operation is run. All Boolean operations use similar dialogs which provide two methods for selecting wireframes: from a drop-down box or by picking with the mouse pointer.



The Boolean operations are primarily designed for use with closed (solid) wireframes so they should be used with caution with open wireframes. The dialogs do have an *Open* checkbox which should be checked if the wireframe is open. If this is not checked, the command will attempt to close the wireframe before performing its designated function.

General Procedure for Boolean Operations

The general procedure for performing a Boolean operation is as follows:

1. Ensure two distinct wireframe objects exist in memory.
2. Select **Wireframes | Boolean Operations | #Boolean Command#**
3. In the resulting dialog, use the two drop-down lists to select the objects you wish to merge – note that different objects must be selected.
4. If you wish to verify your wireframe data before the Boolean operation, select the either or both *Verify* check boxes.
5. If either of the wireframe objects are open wireframes (not an enclosed volume) select the appropriate *Open* check box(es). Unless the *Open* check box is selected the command will attempt to close an open wireframe before the Boolean operation is executed. The results with open wireframes may be unpredictable and caution should be exercised.
6. If you do not wish to include faces in the resulting union that have been contributed by object 2, select the *Cookie Cutter* option. In some cases this may result in an open-ended wireframe object; hence the name "Cookie Cutter". (see *Appendix A* for more information).
7. Click **OK** to begin the Boolean operation. Note that if verification has been selected, the process will take longer (the extent of this effect is dependent on the resolution and state of both wireframe objects).

Boolean Operation Tolerances

A Boolean command relies on the presence of two wireframe objects in memory. The calculations that are run may require some extrapolation of results in order to create the necessary resulting object(s), and Studio 3 will perform these commands according to a built in tolerance. This tolerance defines the overall required accuracy of the operation (with regards to the geometry of the original data versus the corresponding resulting data or data subset). This tolerance is set in the **Project Settings** dialog (**File | Settings**).

The *Wireframe* settings tab contains a *Boolean Tolerance* field, used to set the maximum accepted deviation for accepting the results of boolean operations. Smaller values will give rise to lower tolerances and, potentially, more processing time required when performing boolean operations.

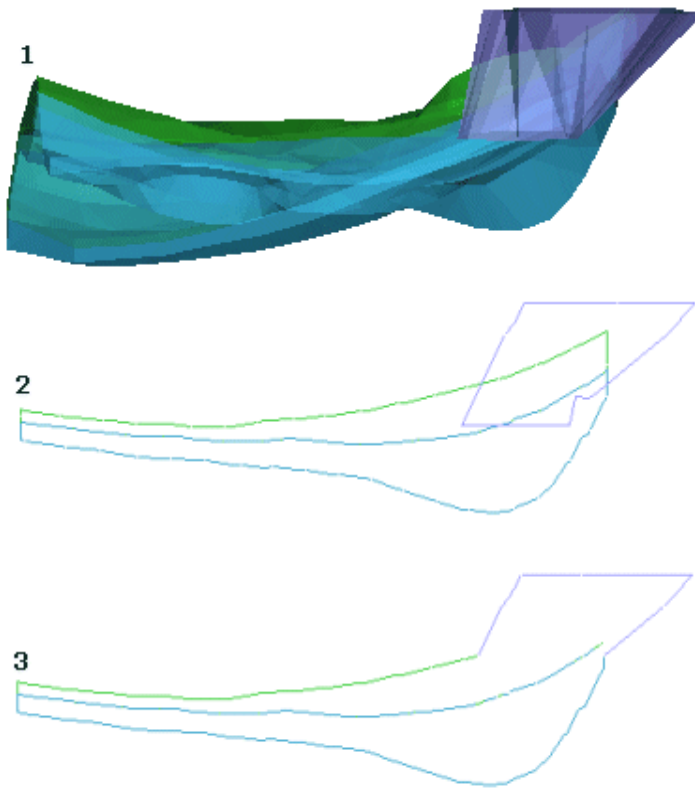
Boolean Functions in Studio 3

The following section describes each of the Boolean functions and will display at least one graphic example of each command.

Union

Wireframes | Boolean Operations | Union

The **Union** operation takes two wireframe objects and creates a single wireframe with the same surface appearance and characteristics as the two component wireframes together. Any common volume is not distinguishable.



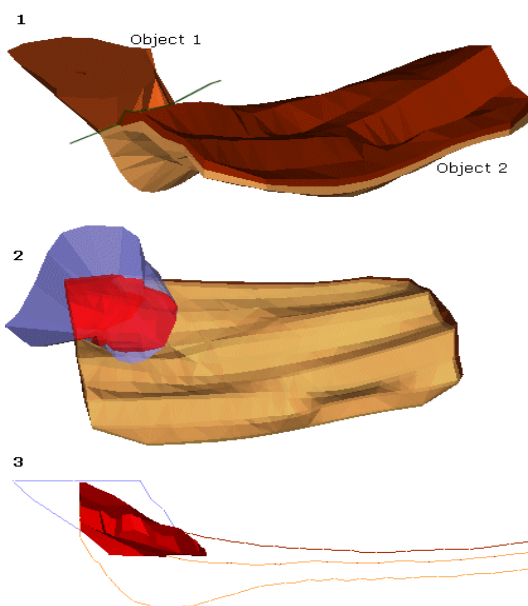
In the example on the left, a low-resolution wireframe has been loaded, consisting of two zones. A newly-discovered shoot has been rendered as a wireframe and loaded into memory as a separate object – a view of both objects is shown in **picture 1**.

A section longitudinally through the two objects in situ has been displayed (**picture 2**).

When a union operation is performed, and the two wireframe objects specified (leaving all default values), the resulting new object created in memory is shown in a similar section view (**picture 3**) – note how all shared (overlapping areas) of the original objects are no longer distinguishable.

The **Union** dialog allows you to select the wireframes to be used and choose whether or not to verify them. Verification is recommended.

Intersection



Wireframes | Boolean Operations | Intersection

This command produces a new object that represents only the common volume of two overlapping wireframe volumes.

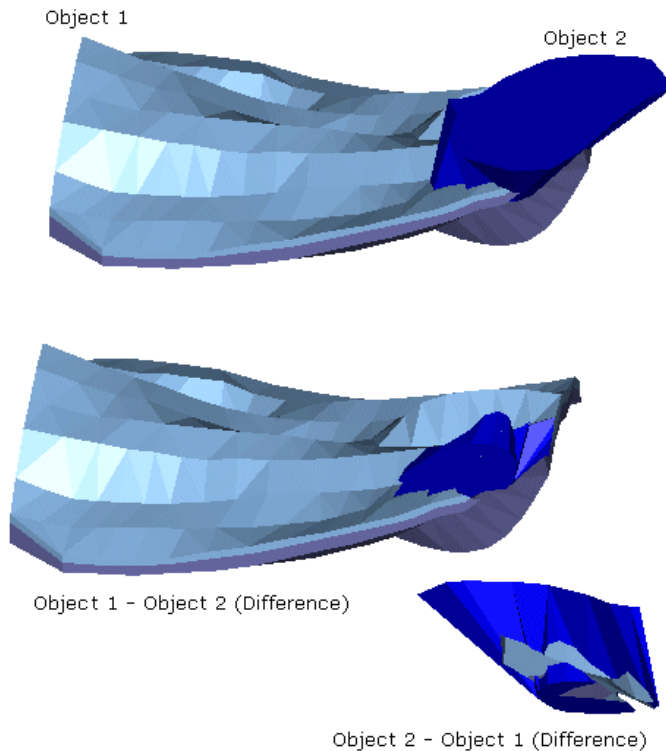
In the example, two wireframe objects are loaded and drawn to the screen - **Picture 1**.

Picture 2 shows a Visualizer view of the objects (color enhanced) to show the two objects as transparent. Note that in this example, the red area is shown to demonstrate the overlapping area of the two objects. You would not normally see this overlap area in this way prior to running the intersection function.

Using the **Intersection** command, a new object is created in memory that represents only the shared (common) volume of the two objects, shown as a shaded red object in **picture 3** (along with an intersection view of the original wireframes).

Difference

Wireframes | Boolean Operations | Difference



The difference command is a useful 'sculpting' technique, where one wireframe is 'carved' by a second. In this operation, there is a difference between object 1 and 2; object 1 will be used as a basis for the resulting new wireframe difference object – and will contain only data that is unique to object 1, any overlapping areas with object 2 will be removed.

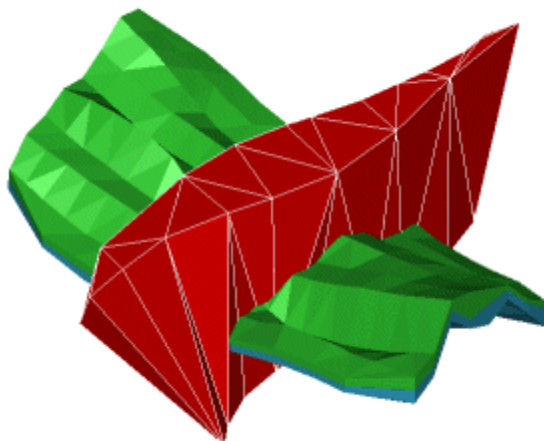
This is highlighted by the example on the left. If the **Difference** command is run, and object 1 is selected in the top-down list, and object 2 in the bottom, the larger of the two resulting images occurs. If the order of objects is reversed, it is object 2 that is carved, resulting in the smaller

difference object.

Extract Separate

Wireframes | Boolean Operations | Extract Separate

In short, this command creates multiple separate wireframes, points or strings for each logically discrete piece of a two-wireframe interaction. Includes "differences", "intersections" etc.



For example, the screen display on the left represents two distinct wireframe objects.

There are both shared and distinct areas in the group; the extract separate operation can create (if the *Single Object Output* option is disabled) a collection of nine separate objects (all wireframes) each representing a distinct 'piece' of the entire collection.

Note that each of these 'jigsaw pieces' is automatically named,

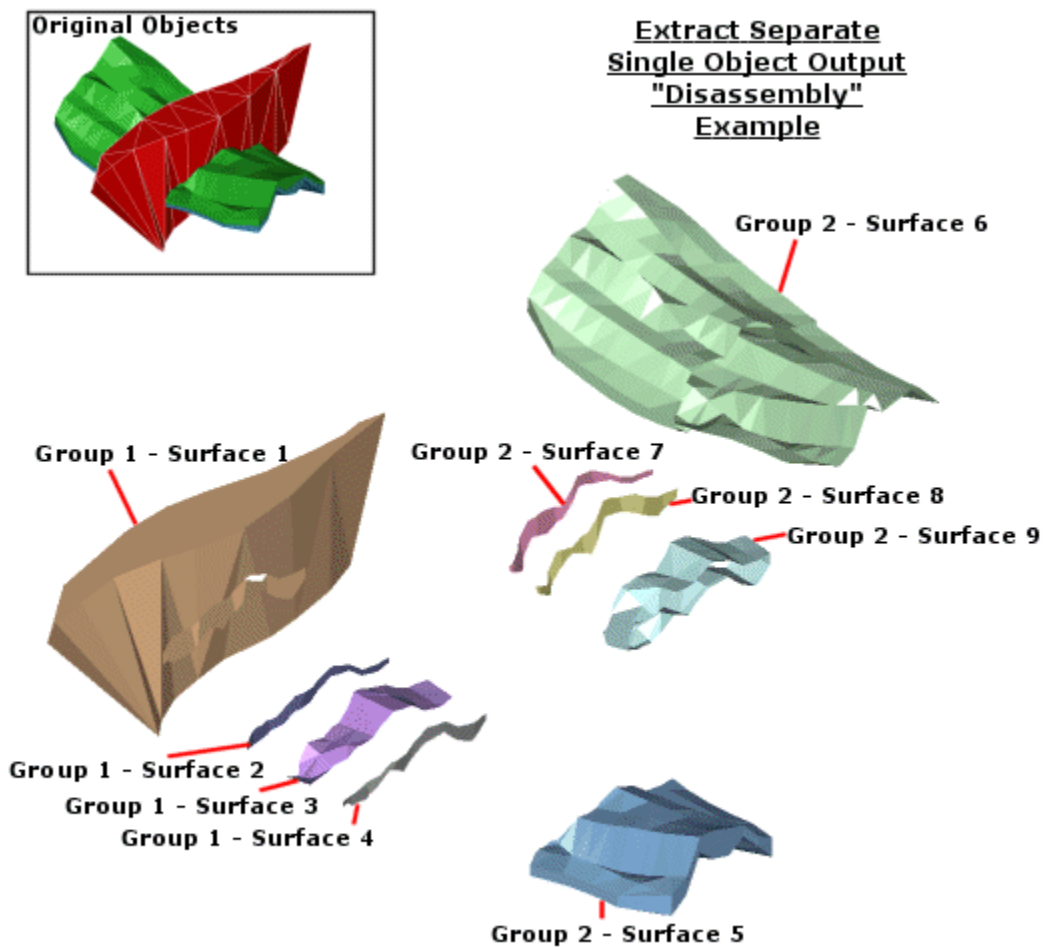
using the syntax of `Extract: #object 1# and #object 2 #piece number#`.

Depending on the complexity (and arrangement) of the objects selected for this Boolean operation, there is a potential for a large amount of objects to be created in memory. For this reason, you will be alerted as to how many objects are to be created before you commit to running the function. At this stage, you can elect not to do this, you will be given an opportunity to produce a less fragmented output.

GROUP (N)	SURFACE (N)	LINK (N)
1	1	1
1	1	1
1	3	
1	1	
2	2	
2	6	
2	2	
2	7	

You can choose to store the results of the `Extract Separate` operation in a single object. This object will store data from both objects within a single database. In this situation, the resulting object's *GROUP* and *SURFACE* fields are important as these determine, internally, which particular triangle 'belongs' to which original object and which independent 'piece' (surface) resulting from the command.

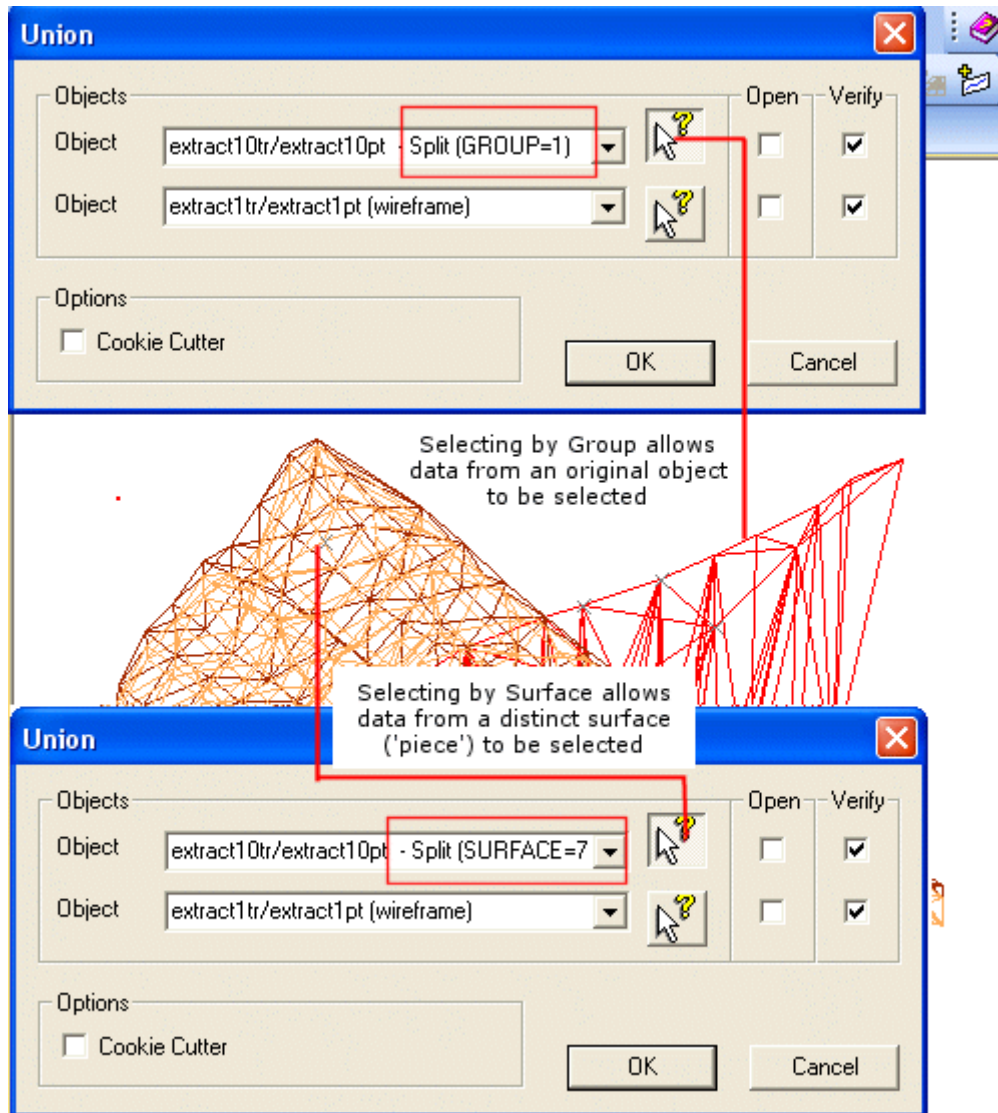
For example, if two objects are selected for an `Extract Separate` operation, and the results could potentially create nine distinct 'pieces', if the *Single Object Output* object was selected initially (or as a result of refusing the first two options given when running a multiple-object-output command, the resulting object would contain the full geometry of both wireframe objects.



If drawn in its entirety to the screen, the results of a single object output from this command would look identical to the display of both original objects (providing the

same display format was used in each case). However, the underlying data of the generated object will contain additional data in the *GROUP* column to represent which of the original wireframe object the triangle was derived from and in the *SURFACE* column to contain a number from 1-9 that represented the extracted, separated surface to which the current data row (i.e. wireframe triangle) belonged.

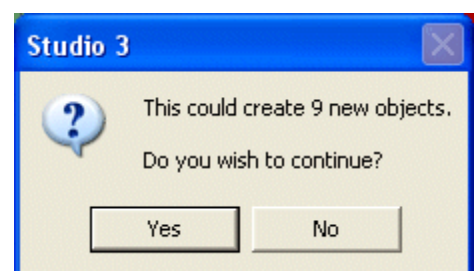
These values are important, particularly with regard to how you specify that wireframe data is to be selected (using the **Wireframe Settings** dialog, as shown in Chapter 4). For the single object output of an **Extract Separate** operation, if you set up your system to select wireframe data 'by group', you can then select data relevant to one of the original objects during subsequent Boolean operations. For example:



“Simple” Groups

So, if you elected not to use the *Single Object Output* option, and having discovered that the number of objects that will be created is prohibitively high, you can choose **No** when asked if you wish to continue. You are then asked if you would prefer to output **simple groups** instead.

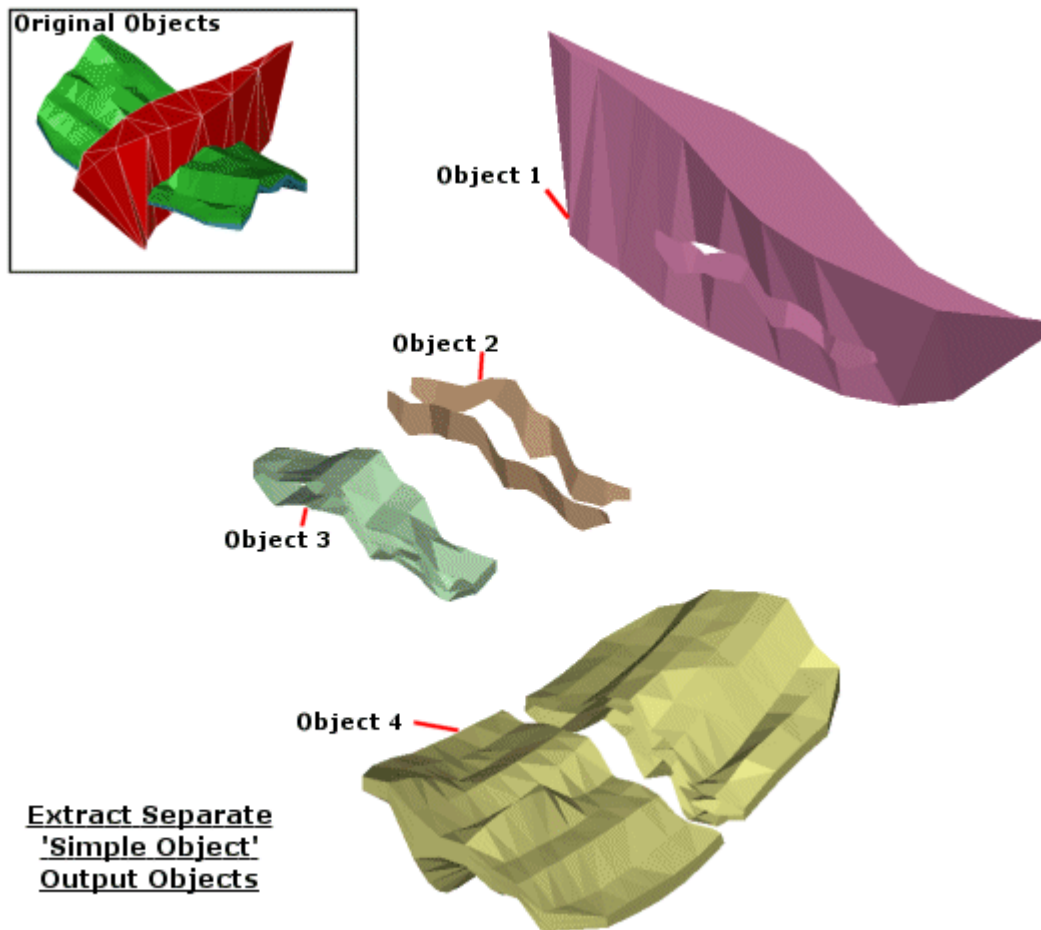
If you choose this option, the command will



generate, at most, four objects representing:

- the part of object1 *inside* object2
- the part of object1 *outside* object2
- the part of object2 *inside* object1
- the part of object2 *outside* object1.

Compare the results of the simple object output with that of the previous 'Disassembly' image:



Note that this option is only available when the *Single Object Output* option is disabled, and the option to create all possible objects is refused.

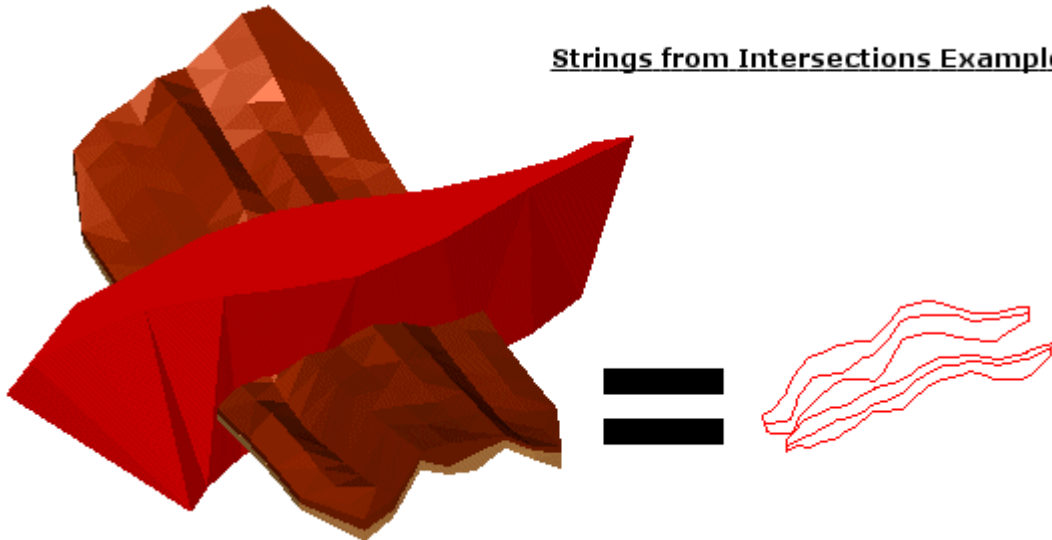
If you decide not to generate simple objects, you are offered a final option to default to the single object output. If this option is refused, no objects result from this operation.

Strings from Intersections

Wireframes | Boolean Operations | Strings from Intersections

This useful command creates new strings from the intersections of wireframe triangles, where two or more wireframes cross each other. As with other Boolean functions, two wireframes must exist in memory.

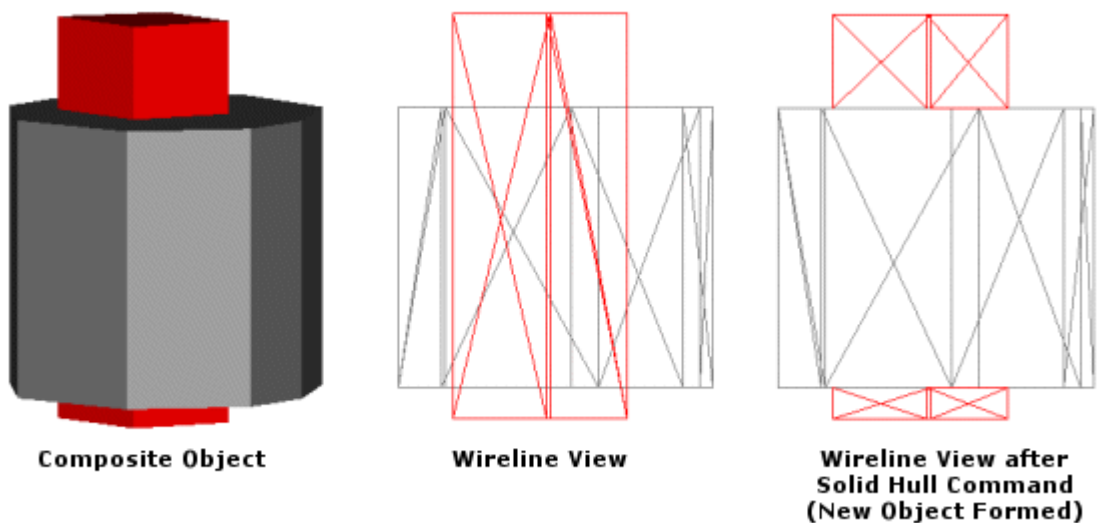
Strings from Intersections Example



Solid Hull

Wireframes | Boolean Operations | Solid Hull

The easiest way of visualizing the effect and results of this command is via a simple example. The wireframe object below is comprised of two different closed surfaces; a grey outer tube and a red intersecting tube. The image on the left shows a solid 3D render of the composite object. The middle image shows a wireline view of the two objects – note how the red sub-object intersects the full depth of the outer (grey)



sub-object. After a solid hull operation is performed, the resulting object, although outwardly not visually different from the original composite, it will be lacking the internal structure present in original composite of the two objects.

The result of the **Solid Hull** command is the equivalent of performing wireframe-union commands on the surfaces in the wireframe two at a time until all are merged into a single solid. The method differs from a **Union** Boolean in that this command resolves the overlap in the selected wireframe. This command is one example of a Boolean function that does not necessarily require two separate wireframe objects to be loaded.

The command should only be used if the individual wireframe surfaces in the input wireframe have been successfully verified, and are closed.

Update DTM

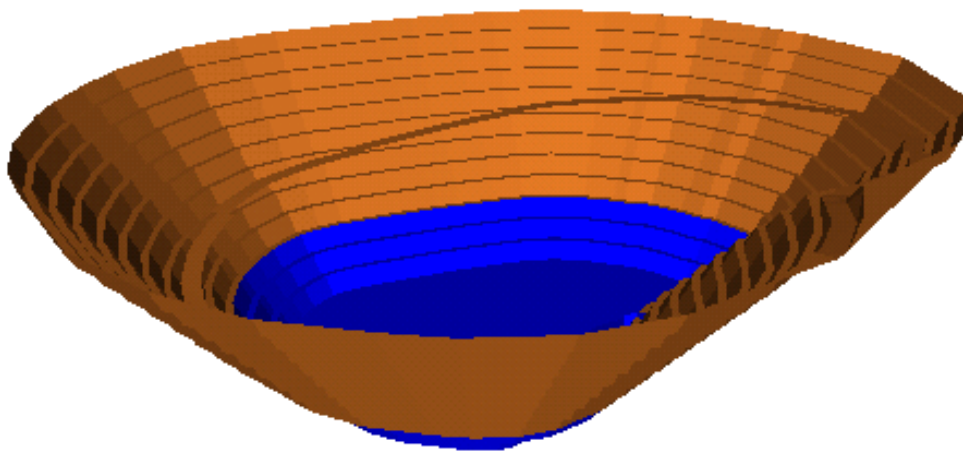
Wireframes | Boolean Operations | Update DTM

This command is used to update one surface wireframe with another. The new surface is generated using the second wireframe selected to update the surface elevation.

In the example shown below the Update DTM (**Boolean Operations | Update DTM**) command has been used to update an open cut pit by adding the new to the existing pit wireframe. The first screen capture shows a slice through the wireframe; different line colors have been used to define the two wireframes:



The following screen captures show the resultant wireframe after the (**Boolean Operations | Update DTM**) has been run:



Plane Functions Overview

All Studio 3 'plane' functions are found in the **Wireframes | Plane Operations** sub-menu.

The term "plane" is used to differentiate this batch of wireframe commands from the previous Boolean operations. Whereas both sets of commands are based on the simple logic of how a wireframe interacts with another entity (or a discrete part of itself), the plane functions are concerned with how a wireframe reacts to another *non-wireframe entity*, such as the viewplane, or a string, for example.

As with Boolean functions, plane operations will create new objects in memory automatically, as many as are required to fulfil the requirements of the operation in question.

The following section describes each of the Plane functions found in the **Wireframes | Plane Operations** menu,

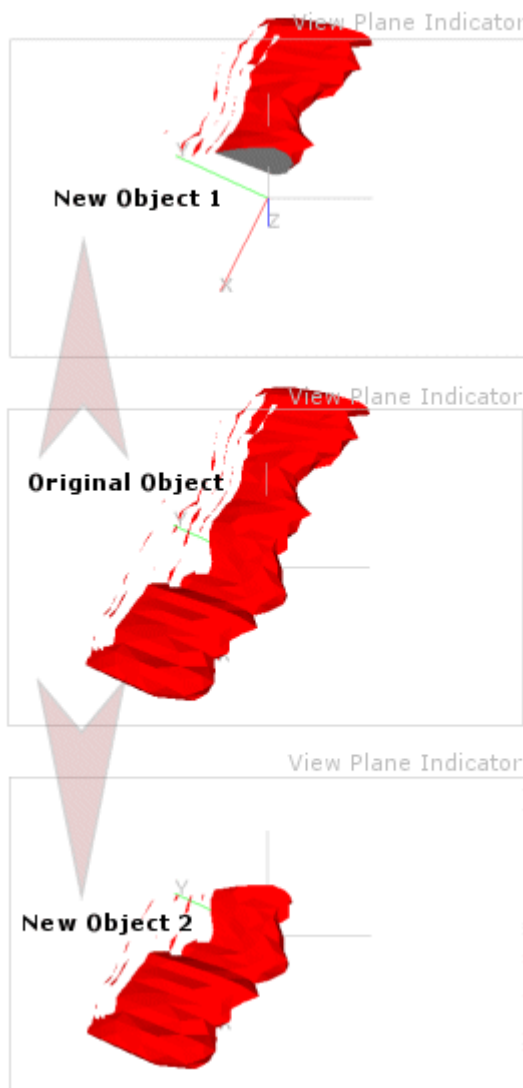
Split

Wireframes | Plane Operations | Split

The **Split** operation is one of Studio 3's simplest plane operations.

A minimum of two new objects are created from this command; one representing the data that falls on either side of either the current viewplane (it is important to have an understanding of the viewplane concept to use this command effectively – see Chapter 3 for more details on the Studio 3 Viewing Hierarchy) or another defined imaginary plane.

This command can either create open surfaces or closed volumes, and a bisection of the current wireframe can be created using any imaginary plane in 3D space. Only one wireframe object can be 'split' and the choice of open or closed result is determined by the *Cap Ends* check box (select it for closed volumes or leave clear for open surfaces).



Once a plane has been defined, either as the current viewplane orientation or by setting up a base orientation (horizontal, north-south etc.) and XYZ reference point, you can split your object by clicking **OK**.

New objects, prefixed by the 'Split' descriptor are then created in memory.

As with any Boolean or plane operation, you can migrate the resulting objects in memory to a physical file by either saving the new objects individually from the **Loaded Data** control bar or by saving your project and specifying how you would like your new data to be stored.

The **Split** operation also supports a full pre-command verification option, and if your wireframe data has not been verified previously, it is recommended that you enable this option before running the command

Multiple Split

Wireframes | Plane Operations | Multiple Split

Similar to the previous command, the **Multiple Split** operation allows you to create several bisected 'slices' of a wireframe body. The additional parameters specified allow you to define the width of the slices (and subsequently, the number of different objects created in memory after running the command).



As with the **Split** operation, you can define the orientation of the 'cutting plane' either to be in line with the current viewing plane, or as any other imaginary 3d plane defined in space – the tools for both split commands are identical other than the additional *Inter-plane distance* option.

By default, the verification option is enabled, and this is often good practice if the wireframe in question hasn't been previously verified. Once you have verified a wireframe, you can store the verified database to a file, and disable the *Verify* option in future, speeding up the overall process. This applies to all Boolean and plane operations.

If you wish to use this command for both **Split** and **Multiple Split** operations, you can do so by defining 0 (zero) as the *Inter-plane Distance* – in this situation, two new objects are created on either side of the defined plane (or viewing plane, if selected). In this mode, both **Split** and **Multiple Split** commands are identical.

Remember that, for larger datasets and small inter-plane distances, the number of objects created can be potentially high, and if this is required, the operation of multiple splitting can take several minutes to complete, particularly if verification is also enabled. Progress bars are displayed for each individual object created so you will be able to see which particular stage the command has reached.

The *Cap Ends* check box, also found in the **Split** dialog, can be used to define whether closed volumes are created (checked) or open surfaces (cleared) as a result of splitting the original wireframe.

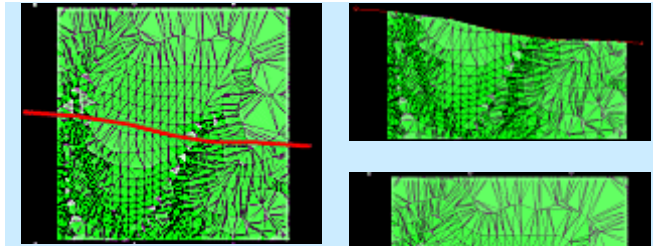
Split by String

Wireframes | Plane Operations | Split by String

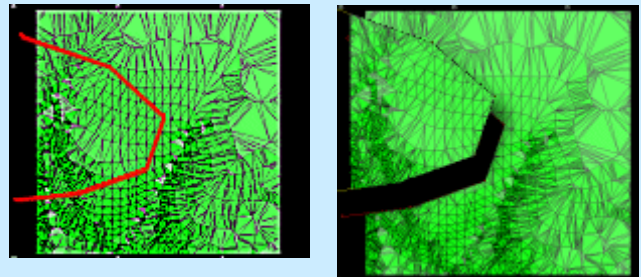
The **Split by String** command also results in new objects being created (each a sub-set of the 'parent' object), but in this command, the object used to define the cutting planes is a string. The dialog shown is identical to the **Split** dialog, and you can even elect to use the view plane or other plane instead of a string, however, if a string is highlighted before the dialog is opened, or if a string object is selected (interactively) before the **OK** button is pressed, the string will be used to define the boundaries of the new wireframe objects.

This command is used to intersect a wireframe, creating discrete objects, using a defined (and previously selected) string. The procedure for using this command is:

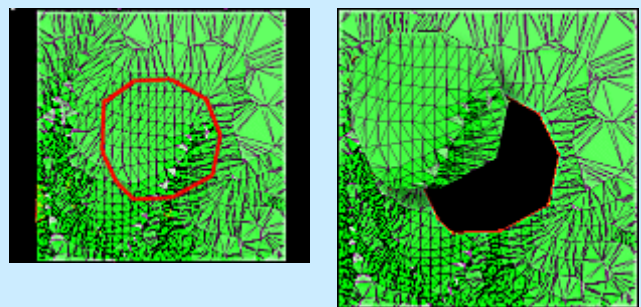
1. Select a string in the **Design** window. This string must be able to create an intersection plane, when project in the selected direction, to subdivide the wireframe data completely (see below for more information).
2. Select **Wireframes | Plane Operations | Split by String**.
3. Select the wireframe object to be intersected from the drop-down list.
4. Define the plane and direction for projecting the selected string to form the 'cutting' plane. By default, the plane will be created by projecting the string directly away from the virtual camera



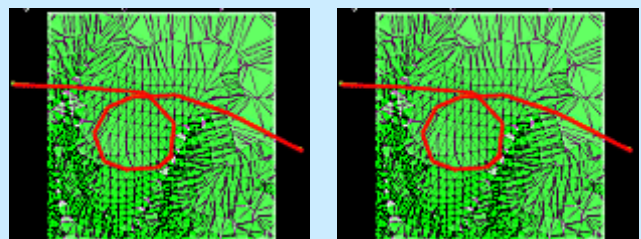
Single open line, unlooped, with terminal points extending beyond the wireframe data body on two separate edges when viewed along the projection plane.



Single open line, unlooped, with terminal points extending beyond the wireframe data body on the same edge when viewed along the projection plane.



Single closed line, with no extents beyond the edge of the wireframe data when viewed from the projection angle. Note that a boolean is still possible even if the closed line 'hangs over' one or more edges of the wireframe



Single looped line, with the looped section positioned along the projection plane. In this situation, two objects will still be created, one empty, and one as the original.

(orthogonally).

- Two separate objects will be created in memory, prefixed with either 'String Split (inside)...' or 'String Split (outside)...'.

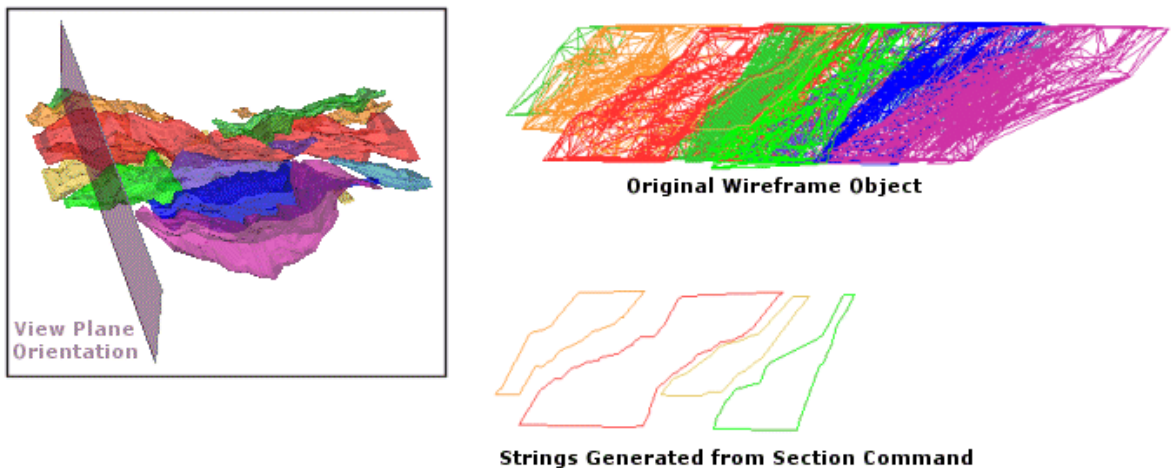
When selecting a string to perform a wireframe 'split', it is important that the string in question is able to subdivide the wireframe data into two separate objects. Looped strings will not be permitted, nor strings that do not extend beyond the wireframe boundaries fully when viewed along the projection plane.

The following table gives examples of what is, and what is not, permitted when using the **Split by String** command:

Section

Wireframes | Plane Operations | Section

The **Section** command is different from the planar functions described so far in that the resultant object is a string, not a wireframe. Instead of splitting an existing wireframe object into sub-objects, a string is generated at the point where a defined intersection plane (either the view plane or user-defined) intersects with the selected wireframe object.



If you wish, you can automatically fill the strings as part of the operation, using the **Fill Strings** check box.

Once strings have been generated, they can be edited/managed as any other string object.

Multiple Section

Wireframes | Plane Operations | Multiple Section

The **Multiple Section** command can be thought of as a 'batch processing' version of the Section command. Multiple sections are created, either as individual string objects, or as a single object containing groups of distinct data.

As with any 'multiple' wireframe command, you need to specify additional information to determine how and where the additional items are created. The initial section is defined in the same way as for the **Section** command, i.e. using the current viewplane, or by defining another plane in 3D space. The distance between each



resulting string (section) will be specified using the *Inter-plane Distance* field. The lower the distance, the greater the potential for resulting sections.

The *Single Object Output* option is selected by default, this check box determines whether a single string object is created, containing multiple string representations of section planes, or if independent string objects (each representing a single plane) are created. Note that the term 'object', as with all references of this type, refers to data held in memory.

The created sections will not be available as a physical file until the project has been saved, and the new strings object saved to either the project file or a standalone Datamine file.

The optional *Section Field* is not available in any other wireframe dialog of this type; if you are generating a single object as a result of the **Multiple Section** operation, this object can have an optional, additional field representing the section index that each string

entity is related to, within the object.

You can use this box to either a) enter a new column definition that will be added to the resulting string object, containing an automatically-incremented section index, relevant to each string entity, b) select an existing field description from the drop-down list (generated from the descriptions found in the original wireframe file) or c) leave the field set to [none] in which case no section index will be output.

6 WIREFRAMES FROM LINKED STRINGS

Wireframes and strings are similar in one respect; they both rely on interconnecting vertices to define their geometry. From this perspective, it is easy to understand why one way of creating wireframes is by linking strings together.

Studio 3 utilizes advanced logic to determine how points are connected to form a 'good' wireframe model. Exactly what constitutes a good model will be discussed in Chapter 8 – Wireframe Verification, however for now, it is sufficient to say that there are many pitfalls for a piece of software to overcome in order to join two or more string entities together in an appropriate manner. The actual manner is decided by you; and will most likely be dictated by the type of wireframe model you want to produce and in which context it is to be subsequently used.

Controlling the way that strings are linked is key to creating a relevant meshed model, and Studio 3 permits as high a degree of complexity as required, from linking together two straight, parallel strings to a complex link between completely different string objects, using "tag" strings as a guideline. There are several tools available to make the process of wireframe generation quick and easy.

String Linking Settings

Before you start linking strings together, it is wise to define how you would like this to be carried out, in other words, defining the *default linking behaviour*.

Linking options are defined in the Studio 3 main **Project Settings** dialog, accessed using the **File | Settings** menu command. Once open, you can access the wireframe linking options by clicking the *Wireframes* tab on the left of the dialog.

String Linking Control

From this tab, you can define how strings are linked using the *String Linking Control* section.

Setting these options will have the following effect on Studio 3 string linking processes:

- **Optimal linking:** If this is selected then several wireframes will be generated internally using different starting points. The final wireframe will be created using the best surface from all those that satisfied the linking criteria.

Toggles between *optimal* (slower) and *sub-optimal* (faster) linking methods. If turned on, when linking strings the wireframe will be created optimally with the best surface being selected from those that satisfy the linking criterion. If turned off the linking will be achieved with tag strings (more on these later in this section) - either user-supplied, or if none exist then the closest pair of points on the two strings after translation to a common centre of gravity will form an implicit tag string. Each pair of tag strings is linked independently. The speed advantage of the sup-optimal option increases with a greater number of string points. If the sub-optimal method fails then linking will default to the optimal method.

- **Link crossover checking:** This automatically checks for a link that intersects another link. Any surface that would contain crossovers will not be created and a warning dialog is displayed.

If the crossover toggle is on then any link that would contain crossovers is automatically rejected.

Crossover checking will increase the processing time, especially where the strings

have a large number of points. For regular string shapes the user may elect to set crossover checking off, but a wireframe intersection check is still recommended.

If sub-optimal linking has been selected and fails then optimal linking will be tried. If optimal linking also fails and you have checked that there are no crossovers in the strings, the best strategy is to:



1. Switch crossover linking off.
2. Complete the link.
3. Use the wireframe intersections command (`wf-intersections`) to locate the crossover

- **Use tag strings:** toggle the use of *tag strings* on or off for string linking. Tag strings are supplied by the user to provide additional control on string linking. This toggle controls whether the tag strings should be ignored when creating the link. The tag strings that may be required for sub-optimal linking to achieve the required shape and avoid crossovers may be unnecessary for optimal linking.
- **Wireframe attributes from strings:** toggle the source of wireframe attributes from strings or user. During string linking attributes can be taken from the attributes associated with the string, or alternatively from the values entered in the attribute panel with the `new-wireframe`, or `new-wireframe-surface` commands.

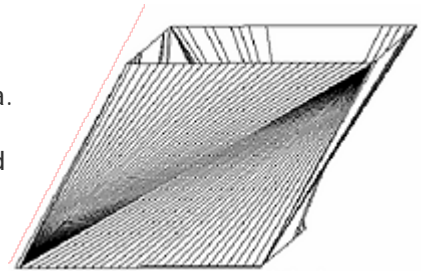
String Linking methods

On the same tab as the *String Linking Control* options, you will find some related settings; *String Linking Methods*. These settings define the actual mechanism by which strings will be linked together, using the settings defined above. There are three options available:

- **Minimum surface area method:**

selects minimum surface area as the string linking criteria.

If toggled on, the resulting triangulation between selected strings will have the minimum wireframe surface area.

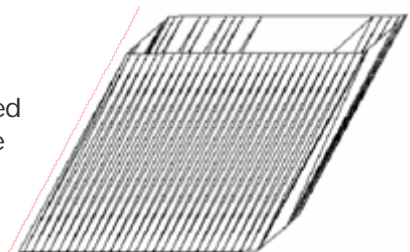


- **Equi-angular shape method:**

selects equi-angular shape as the string linking criteria.

If toggled on, the resulting triangulation between selected strings will generate triangles where the preferred shape is equi-angular.

One measure of the shape of the triangle is the radius of the 'circumcircle' fitted through the three vertices of the triangle; if the three points on the two strings selected to form a triangle minimise the circumcircle radius this will generate a well shaped triangle.



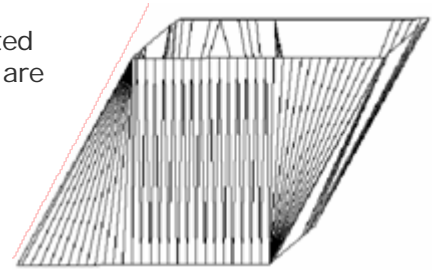
If the three edges were of equal length the triangle would be an equilateral, and if two edges are of equal length then an isosceles triangle is formed with equal angles at the

base of the triangle. If the optimal triangulation minimises the sum of the circumcircle radii of the triangles formed, then the preferred shape will be equi-angular.

- **Proportional length method:** selects proportional length as the string linking criteria.

If toggled on, the resulting triangulation between selected strings will generate triangles where the triangle edges are similar proportional distances along the two strings.

This criteria works best where the shapes of the two strings are similar. If the shape of the strings is dissimilar, then to get best results tag strings should be used to link together those string sections that are similar. The starting edge for the triangulation is determined either by a user defined tag string, or selected by the system using the closest pair of points on the two strings to be linked. The triangles are formed to best maintain their proportional position along the two strings.



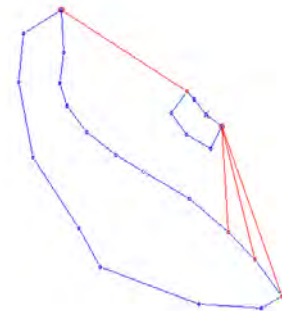
If multiple tag strings are supplied, then the proportional position is defined between adjacent pairs of tag strings. For any pair of strings this method can only generate one link. If this criteria is selected and the toggle linking-method-switch is set on for optimal linking, then the equiangular shape method (link-radii-method switch) will be used.

Although not set as the default string linking criteria, it will always be the *fastest* because no optimisation is involved. Where the shapes are similar this method can often produce the best link, particularly if tag strings are used judiciously. If string shapes are not similar then the other criteria (equi-angular shape, minimum surface area) work more effectively.

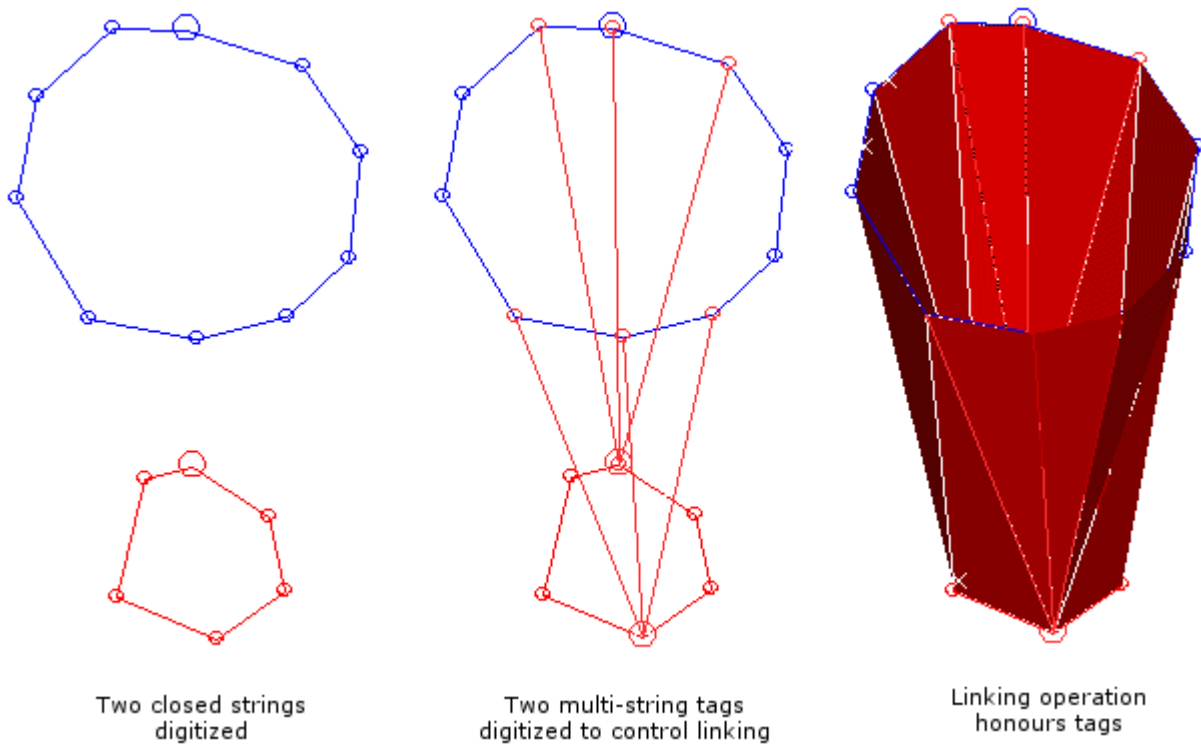
Understanding “Tag” Strings

Tag strings are used to increase control over the linking process. They can define the points to be linked for the **Link Strings** command, for example, which is particularly useful when wireframing complex shapes.

A tag string may contain any number of points but each one *must be on a different string perimeter*. It is also possible to link a single point on one perimeter to a number of different points on the second perimeter as shown on the right.



As a tag string is not a standard Studio 3 string, it cannot be created using the usual **New String** command, instead, the command **Wireframes | Linking | Create Tag String** must be used instead. It is also important that the first and last points of the tag string are ‘snapped’ to the relevant locations on the both of the strings to be linked – this is important. Equally important is that, for tag strings to be used during the linking process, the Use Tags toggle must be enabled. This is found in the same sub-menu and must be active before linking is performed for the associated strings to be used.



When digitizing tag strings, it is a good idea to pick a color that will easily identify the string type. This can be done using the color picker panel during tag creation.

Remember: tag strings will only be used if the *Use Tags* option has been turned on in the **Wireframes | Linking** menu. Tags can be used with *any* of the wireframe linking methods listed in the previous section. It is worth noting that the tag strings that may be required for *sub-optimal* linking to achieve the required shape and avoid crossovers may be unnecessary for *optimal* linking.

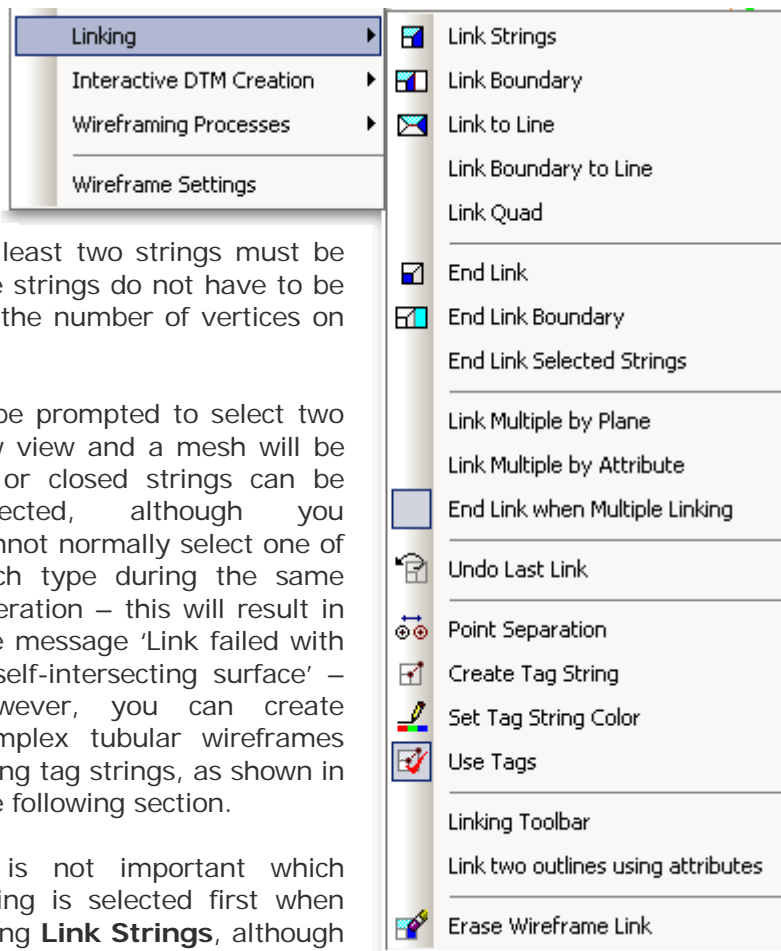
Linking Commands

All Studio 3's linking commands can be found in the **Design** window's **Wireframes | Linking** sub-menu. This menu contains all commands relating to the linking of string data together to form wireframe models.

All of the linking commands are also available as command-line instructions, and this will be indicated wherever an explanation is given of a particular command. A graphical representation will also be shown to demonstrate how linking is performed when a particular command is run. For all of the commands in this section, the *Optimal Linking*, *Link Crossover Checking* and *Wireframe Attributes from Strings* options have been set, and the *Equi-angular Shape* linking method is being used.

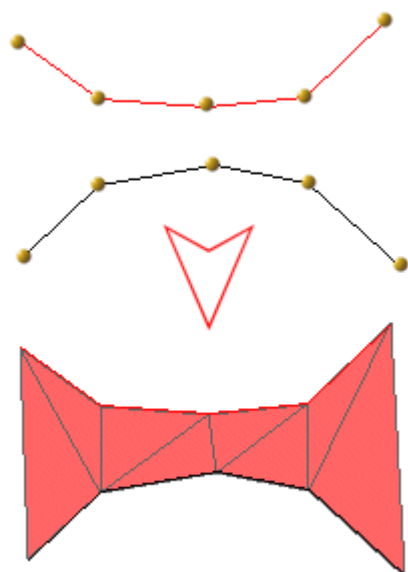
This section outlines some of the more commonly-used linking commands; your online Help provides a full description of all commands available.

Link Strings: this is the simplest wireframe linking command (at least, as regards the apparent result – the actual calculations that occur ‘behind the scenes’ are often quite complex).



For this command to be effective, at least two strings must be loaded into the **Design** window. These strings do not have to be of a similar shape or orientation, and the number of vertices on each can be different.

When this command is run, you will be prompted to select two strings in turn, in the **Design** window view and a mesh will be created between them. Either open or closed strings can be selected, although you cannot normally select one of each type during the same operation – this will result in the message ‘Link failed with a self-intersecting surface’ – however, you can create complex tubular wireframes using tag strings, as shown in the following section.



It is not important which string is selected first when using **Link Strings**, although the actual linking method and options may affect how the strings are joined together.

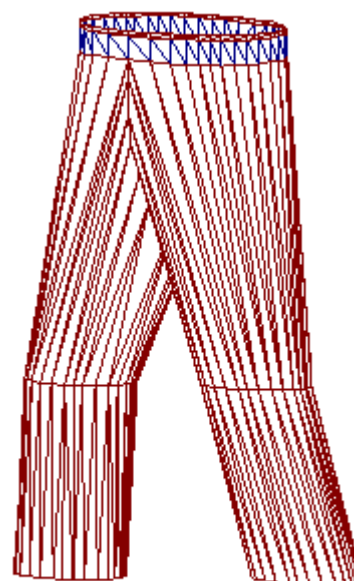
In the image shown on the left, the resulting wireframe view is taken from the **Visualizer** window, although the same effect could be emulated in the **Design** window by

enabling *3D Rendering* in the **Format Display** dialog – see your online Help for more details.

Link Boundary: Links two strings together taking into account (and not crossing) any boundary strings. A minimum of one boundary string is necessary for this command to be used. A boundary string is created using the new-string command, and the string must be snapped to a point on the same string at each end. The boundary *divides* a string, to assist in creating a split wireframe; this command is designed to make it easy to link multiple strings to one other string to create a bifurcated or split wireframe model, as is often required in geological modelling.

The strings to link together are selected in pairs. If a string contains bridge strings the selection must be made on a part of the string that is to be ‘wireframed’. You may continue to select pairs of strings until you select another command.

If present, this command will honour any previously defined tag strings. Tag strings can contain any number of points and be fully three dimensional, but their end points must be connected to points on one of the selected strings. Crossovers of tag strings should be avoided.



An example of using the link-boundary command to overcome a ‘trouser leg’ problem.

Link to line: Links a string to a single line (not a closed string section). This is useful for "closing-off" wireframes.

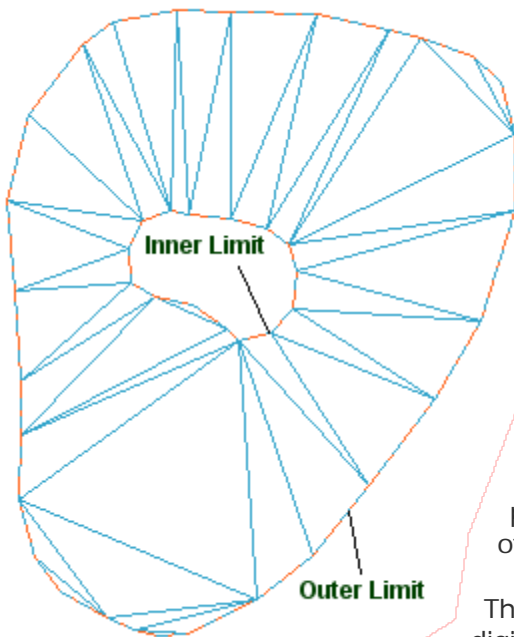
7 DIGITAL TERRAIN MODELS

Simple unfolded surfaces like topography or terrain surfaces can be modeled as a Digital Terrain Model or "DTM". A DTM provides an exact triangulated surface model in which all the data points are honoured. To create a DTM, a set of X, Y, Z data values which describe the surface are required. This might take the form of point surveys or contour strings depending on the source of the data.

Although DTMs are a special type of wireframe model, the file structures of DTMs and other wireframes are identical. Some wireframe commands can also be used for DTMs, e.g. Write Wireframes is used to save all wireframes, including DTMs.

The **Interactive DTM Creation** menu contains the commands for creating surface wireframes (c.f. solid wireframes). This menu can be found in the Design window under **Wireframes | Interactive DTM Creation** and contains an array of functions similar to the image shown on the left.

DTM Inner and Outer Limits



DTM generation may be constrained by closed strings defined as *inner* or *outer* limits. Limits are created by digitizing new strings in the standard manner, and must be displayed before your DTM is created. These can represent a two-dimensional outline of the required DTM or they may contain points that are required to make up part of the DTM. If a string represents a two-dimensional outline then only the X and Y coordinates can be used.

As well as defining the DTM limits, you can also specify the resolution of the triangles to be used when creating the final mesh, by defining both minimum and maximum triangle properties. Studio 3 also has the facility to introduce new points where appropriate, to enhance the creation of the DTM.

The process for creating DTM limits involves the digitizing of one or two strings (depending on whether you wish to specify both an inner and outer limit). Once these are digitized, you will need to select them using the **Select Inner Limit** and **Select Outer Limit** commands (remember to complete the digitizing process first by selecting the small **Cancel** button in the top left corner of the **Design** window before selecting limits).

Once selected, you can can **Deselect One Limit**, and then indicate the limit to be deselected, or you can **Deselect All Limits**.

Of all the commands available, one that should be visited prior to creating any Digital Terrain Model is **DTM Settings**. This area is used to define how a DTM mesh is created.

Understanding DTM Settings

Selecting **Wireframes | Interactive DTM Creation | DTM Settings** in the **Design** window opens your Studio 3 **Project Settings** dialog, open at the *Digital Terrain Models* tab. There are three options available to determine how models of this type are constructed:

- **Use selected limit strings:** this check box controls whether the actual data points in any limit strings are used to form part of the DTM.
- **Duplicate point checking:** if enabled, checks will be made for duplicate points in the DTM data body. If disabled the processing will be a little faster but the operation will be less tolerant of imperfect data.
- **Use World coordinates (otherwise view coordinates will be used):** if enabled, the DTM will be created using the actual coordinates of the data being used. If disabled, the data will be transformed into the current view coordinates before being processed. This enables vertical DTMs to be created, for example.

Other setting for DTM creation can be found in the **Wireframes | Interactive DTM Creation** parent menu, as shown in the image at the start of this section:

- **Set Point Tolerance:** prevents triangles from being created with sides shorter than the value set for this tolerance. Effectively, setting a higher value will produce less (and larger) triangles during DTM creation.

For the **Wireframe Verify** command, points within this tolerance will be considered duplicates for the purposes of verification. Any triangles that are degenerate after this point merging will be discarded.

This command can also be accessed with the quick key 'sto'.

- **Maximum Separation:** sets the maximum side length for any triangle. Setting a lower value for this setting will result in smaller (and more) triangles. This command can also be accessed with the quick key 'mse'.



Creating a DTM in conjunction with specifying point separation values: to triangulate data where the maximum segment length needs to be reduced, you can speed up the DTM creation operation by conditioning strings as required beforehand.

- **New Point Separation:** When creating a DTM, you can force Studio 3 to add additional points to improve the triangulation. This can be useful if some of the string data has long segment lengths which would create large triangles. By setting a separation distance greater than zero string segments are divided up into shorter segments of the set value for triangulation.

Care should be taken to use the same separation for adjoining wireframe surfaces so that triangle edge lengths correspond at the surface join, and then a wireframe-verify can be completed without a mismatch of edge positions.

The separation distance is applied to any new points automatically created. The same separation for adjoining wireframe surfaces should be used so triangle edge lengths correspond at the intersection of the two wireframes.

This command can also be accessed with the quick key 'nps'.

Note the new points are not inserted into the strings but are incorporated into the wireframe. The string data is not changed using this command.

Creating the Digital Terrain Model

When all the required settings have been made, including the selection of limit strings (if required), you can select **Wireframes | Interactive DTM Creation | Make DTM**.

Once a terrain model has been created, you can 'undo' this operation as the DTM includes information relating to how it was originally created.

Creating a DTM using the SURTRI Command

SURTRI is another method available for creating digital terrain models. A Studio 3 process, **SURTRI** generates a triangulated digital terrain model from perimeter, string and/or point data subject to optional boundary and string edge constraints.

The method used is that of a constrained Delaunay tessellation. This process performs all the functions of the **TRIANR** process with advantages of greatly improved speed performance and several additional features.

Points, strings, perimeters and boundaries may be input to the process. Additional points may be added into strings. Crossing strings are automatically resolved. Points duplicated within a tolerance are removed. Maximum separation of points to be joined may be specified. Boundaries may be 2D or 3D internal or external. The centre of gravity of each triangle may be output. An error trace file containing a dump of data may be produced to help locate bad data.

The points used in the tessellation are output into a wireframe points file **&WIREPT** with fields *XP*, *YP*, *ZP* and an additional field *PID* which is the point number allocated by **SURTRI**. The triangle definitions are output into a wireframe triangle file **&WIRETR** with fields *PID1*, *PID2* and *PID3* which identify the points at each vertex. Additional fields *XBAR*, *YBAR* and *ZBAR* defining the centre of gravity of each triangle may optionally be added to the triangle file. These output files may be used as input to other wireframing processes such as **ADDTRI**, **TRIFIL**, **PLOTWS**, **ISOTRI**, **PLOTTR**, **TRICON**, **TRIVOL** and **WIREPE**.

Boundaries

The tessellation may be constrained by a number of bounding perimeters. These perimeters are input via the **&PERIMIN** file. Bounding perimeters may form either internal or external boundaries. External boundaries will restrict tessellation to within the perimeter while internal boundaries will restrict tessellation to outside the perimeter.

A combination of internal and external boundaries may be used at one time subject to the following rules:

- Boundaries must not cross each other.
- If internal and external boundaries are both present then each internal boundary must be fully contained within an external boundary.

For more information on the **SURTRI** process, please consult your online Help. Other wireframe-related processes are discussed in more detail in "Other Wireframing Processes" (Chapter 11).

8 WIREFRAME VERIFICATION

Wireframe verification is an important facility in Studio 3. It is used to analyse the structure of wireframe objects in memory to a level whereby any potential data discrepancies are highlighted and, if possible, fixed before they can cause problems later.

Analysis of wireframe data, and resolution of problems, can significantly speed up subsequent processes, and lead to more consistent results.

Using the Verify Command

The **Wireframes | Verify** command is used to perform a number of validation checks, including:

- Identifying holes within a wireframe surface.
- Identifying intersections of different surfaces with different IDs.
- Identifying crossovers within, or between, surfaces.
- Checking for duplicate points.
- Renumbering wireframe surfaces.

Wireframe verification takes place according to a series of predefined settings. These settings are accessed by opening the **Verify Wireframe** dialog (**Wireframes | Verify**).

Before running a verification process, you will need to select the wireframe object to be verified, using the *Name* drop-down list.

Alternatively, you can use the “Data Picker” tool on the right of the field to interactively select a wireframe (or part of a wireframe, according to your selection settings in the **Project Settings** dialog, *Wireframes* tab). Once data has been selected for verification, you can specify how verification will be performed using the fields in the remainder of the dialog:

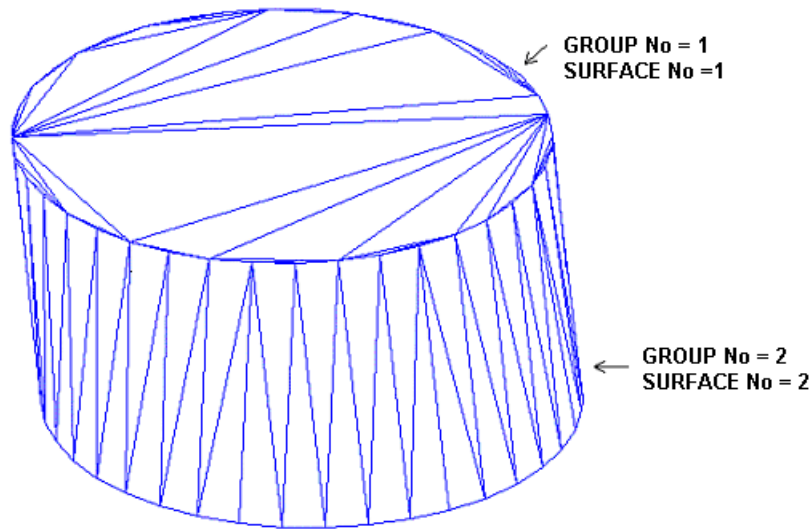
- **Store surface number:** identifies separated surfaces based on face connectivity, assigns a separate index to each surface, then stores that index in the field specified.

This setting enables the renumbering of surfaces during verification. It is useful for re-grouping wireframes. The wireframe below consists of two GROUPs. The links between the perimeters belonging to one GROUP and the end-links belong to another. To select this wireframe as a single entity the surfaces must be renumbered so they are the same.

As the surfaces have the same color, the selection method should be set to *By Attributes*. This will ensure that all the wireframe data required are selected.



By Filters could also be used as a selection method, providing *Store Surface Number* is selected. The surfaces will be renumbered during the wireframe verification. The resultant wireframe will have a single *GROUP* number.



- **Remove duplicate vertices:** removes multiple instances of vertices which occur in the same location, and combines them into a single reference.

The *Tolerance* value is used to determine what a 'duplicate' vertex is; any vertices with a distance between them equal to or less than the value will be classed as being duplicates. In the event that this then causes a face to be degenerate (i.e. not having at least 3 unique coordinates), it will be classed as an empty face and reported or removed depending on the appropriate setting. Tolerance values are saved to the registry, so will be 'remembered' next time this dialog is used.

- **Remove duplicate faces:** removes multiple instances of faces which share the vertex coordinates. This field has an associated Tolerance field.
- **Remove empty faces:** removes any faces which have zero surface area.
- **Check for open edges:** searches for edges which not shared between 2 faces. Where found, a new object is created containing strings made up from the open edges.
- **Check for shared edges:** checks for edges shared by more than 2 faces. If found a new object is created containing strings made up from the shared edges.
- **Check for crossovers:** checks for faces that intersect, but are not adjoining. Where found, a new object is created containing strings made up from the edges formed by the intersections.

9 MANIPULATING EXISTING WIREFRAMES

Once you have wireframe data in memory, you can perform several editing functions to the object as a whole, or you can edit individual elements of a wireframe, down to the editing of individual vertices and edges.

The functions described in this chapter, in most cases, can be run by either selecting a wireframe *before* the function, or by selecting a wireframe object when the resulting command is in operation (this could be a selection from a drop-down list or an interactive data picking tool).

The items described in this section can be found at the top level of the **Design** window's **Wireframe** menu. The commands outlined in the image on the right will be discussed in detail in this section.

The Current Wireframe Sub-menu

The first section to be discussed isn't a command but a sub-menu; **Current Wireframe**. This sub-section of the **Wireframes** menu contains some useful mesh manipulation commands that enable you to create new triangles manually, add extra points to a wireframe, move individual wireframe vertices etc.

Expanding the Current Wireframe menu displays the following options:

- **New Triangle**
- **Unlink Triangle**
- **Move Wireframe Point**
- **Insert Wireframe Point**
- **Delete Wireframe Point**
- **Show Wireframe Slice**

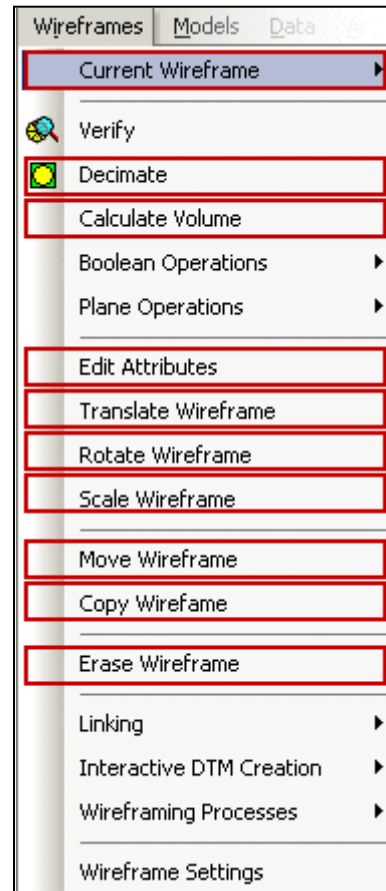
New Triangle

This interactive command requires you to enter 3 points, with the mouse (or other digitizing device) into the **Design** window's display area.

These points represent the vertices of a single wireframe triangle and will be digitised directly onto the current viewplane.

A new triangle does not have to be linked to an existing wireframe object, although this can be achieved by 'snapping' vertex positions to existing mesh points (using the right-mouse button instead of the left when near an existing point).

For each new triangle, a color bar will allow you to define the color of the resulting wireframe 'face'.



Unlink Triangle

The **Unlink Triangle** command can be used to remove the edges connecting adjoining triangles, according to how they were originally linked together. This operation is performed on a triangle-by-triangle basis, interactively using the cursor.

A point on the wireframe is first selected, and all triangles with the same LINK value (i.e. which were formed in the same linking operation) will be removed from the current wireframe surface.

The command is completed by clicking **Cancel**. If no wireframe data is currently selected before running this command

Move Wireframe Point

Another interactive command for editing wireframes, **Move Wireframe Point** enables you to manipulate the positions of individual mesh vertices using the cursor. Edges connected the selected vertex will be honored during the manipulation process.

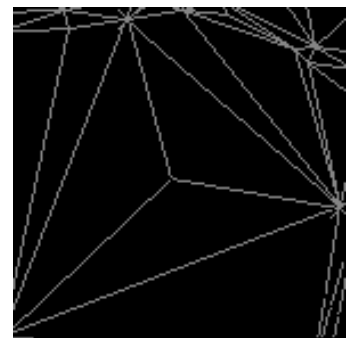
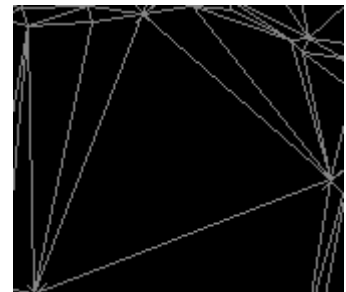


When editing wireframe data with the **Move Wireframe Point** operation, no check is made to secure the integrity of the wireframe surface; it is possible to cause 'crossover' vertices by moving a point outside the confines of the boundary created by neighbouring points linked together.

Insert Wireframe Point

Insert Wireframe Point: insert a new wireframe vertex, and associated triangles to the selected wireframe segment. The procedure for inserting a point is as follows:

1. Select the **Insert Wireframe Point** command.
2. Select a triangle on the wireframe data you wish to insert a point into. The vertices of the selected triangle will be highlighted e.g.:
3. Click again, inside the selected triangle to position the new point dynamically with the mouse e.g.:



Delete Wireframe Point

Delete a wireframe point and recreate modified triangle edges using this command. Note how this differs from the **Unlink Triangle** command; with this option, gaps are avoided as the

surface mesh is automatically reformed after point deletion wherever required. Click **Cancel** to exit point deleting mode.

Show Wireframe Slice

Selecting this option displays a temporary line in the **Design** window that represents the intersection of wireframe data and viewplane.

This data is only viewable in the **Design** window. Note that the intersection line is temporary and is only viewable until the next time the screen is refreshed (e.g. by redrawing).

Decimating Wireframes

Wireframe decimation reduces the amount of data in a wireframe by reducing the number of edge and wireframe faces according to selected and definable criteria. This reduction in data should be sensibly applied so as not to degrade wireframe data beyond the point at which it is meaningful.

By the very nature of the command, decimation results in a loss of data. This means that best practices should be adopted to provide the optimum balance between a less meaningful data set and one that is small enough to be easily managed in day-to-day engineering.

Wireframe Decimation Guidelines

The following guidelines are useful to remember when attempting to reduce wireframe data with the decimation routines in Studio 3:

- The verification process should always be run before a wireframe is decimated, using the **Wireframe Decimate** Dialog.
- When running a pre-decimation verification, the *Remove Duplicate Vertices* option should be selected.
- The *Maximum Error* and *Percentage Reduction* options are key to a successful decimation; in most cases, it is advised to use a *Maximum Error* instead of a *Percentage Reduction*. A sensible value for the former will normally preserve important features, whereas the latter will use a less elegant (although quicker) algorithm in order to achieve the expected data reduction.
- The best *Maximum Error* value will vary greatly depending on the context of the wireframe (e.g. topologies may be able to cope with large errors, but an engineering design may not). It is for this reason that *Maximum Error* is not specified by default, even though it is preferred.
- For wireframes where dimensions are tight, like an underground mine design, a very small tolerance is recommended.
- A weighting of 1.0 will try to preserve the feature, but 0.0 will attach no importance to that feature and it will probably be lost.

The Decimate Wireframe Dialog

The **Decimate Wireframe** dialog is the hub of mesh data reduction activities, and is reached by selecting **Wireframes | Decimate** in the **Design** window.

A full explanation of all the fields in this dialog are provided in your online Help, however it is useful to understand some of the key options provided, as they will allow you to decide on optimum decimation settings more quickly, with a higher degree of 'right first time' successes.

When an object has been selected for decimation, you will need to decide which method your Studio 3 system is going to use in order to reduce data effectively. There are two options available, each one of which has distinct advantages, and potential downsides:

Quadratic Decimation: this option is the quickest option (in some cases, significantly), although there is relatively little verification of data integrity, meaning that the resulting wireframe data will be less accurate (i.e. deviates further from the original in terms of overall geometry) than with a *Measured* calculation.

Measured Decimation: the slower of the two methods, a measured decimation will perform more calculations whilst reducing data to determine which points are to be removed. This process will, overall, result in the integrity of the wireframe surface being preserved to the highest degree (according to the settings made in the remainder of the dialog).

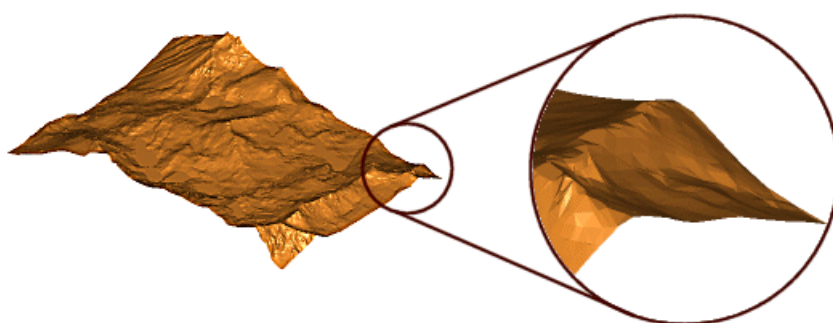
The decimation method and associated settings should be decided according to the context in which your wireframe model is to be used. For example, for a topographical surface, to be used for visualization purposes only (e.g. it is not likely to be used as a parameter in subsequent operations such as determining the properties of an orebody model above or below a DTM, or used as part of calculating cut or fill volumes etc.), you may find that a significant reduction in points is possible on large data sets (50000 points+). As an example, the original topographical wireframe surface shown in the table that follows contains a total of 65014 faces (triangles) in its raw form, straight from survey data. This particular block of topography will not be used for anything other than complementing a VR scene, so a reduction in data will not adversely affect the accuracy of subsequent operations.

The table shows three phases of decimation. First, the surface is shown using all possible faces, that is, the highest possible resolution. Reducing the surface data will allow the surface to be loaded and/or manipulated more rapidly in future.

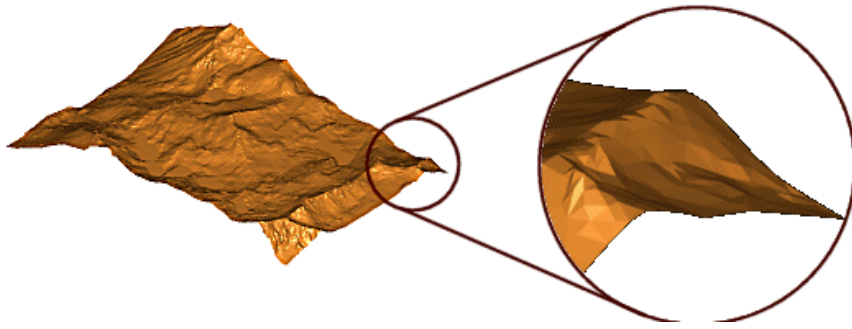
A reduction of 66%, using the quadratic option results in a significant drop in triangle numbers, from 65014 to 22914. The second image shows the same topography at the lower resolution.

A further 66% reduction, resulting in only 7791 faces is displayed in the final image in the series.

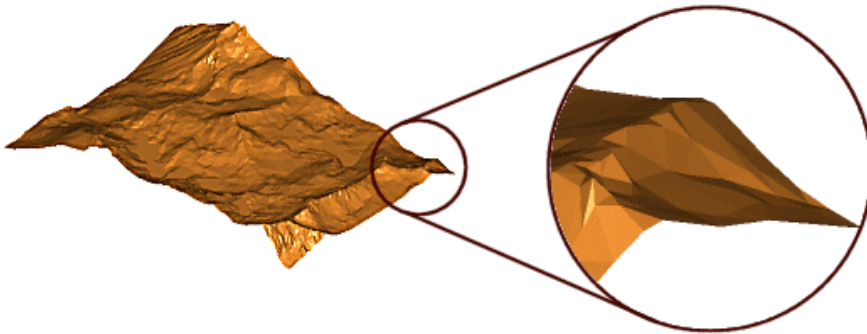
65014 faces



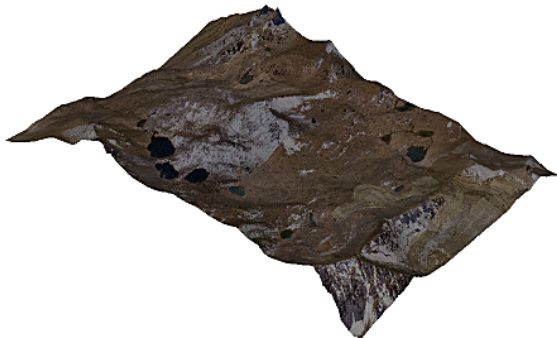
66% Reduction (22914) faces



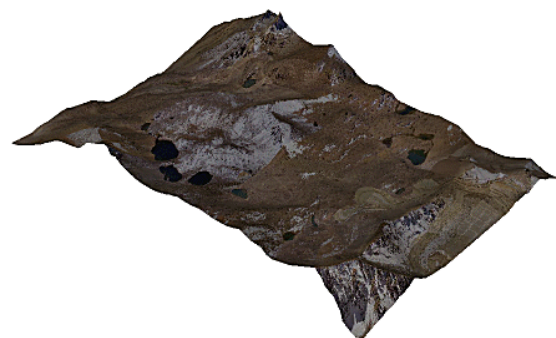
Further 66% Reduction (7791) faces



With a texture applied, in the **VR** view, this reduction in data is almost unnoticeable:



65104 faces with applied texture



7791 faces with applied texture

10 CALCULATING WIREFRAME VOLUMES

It is often useful to accurately record the volume of a closed wireframe; this may be to indicate the overall tonnage for a geological model (or filtered aspect of such a model) or to analyse cut and/or fill volumes for open pit design, for example. The calculation of the volume of a wireframe can be performed using the **Wireframes | Calculate Volume** command. This command is discussed in more detail in the following section.

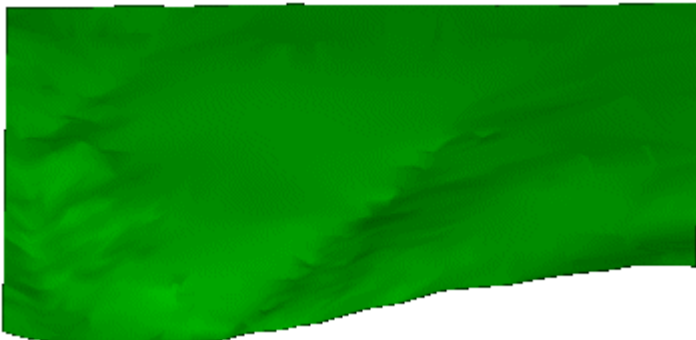
The actual process of measuring a volume is simple; you select a wireframe object in memory and it will be measured, however, sometimes it can be a more complex procedure to actually create the volume that needs to be measured.

Cut and Fill Volumes

Thankfully, Studio 3 provides a wealth of commands, such as Boolean operations (as described in Chapter 5), that allow you to create subsets of data by comparing two volumes.

For example, if a topographical surface model was captured from survey data at the start of a particular phase of mining operations, this model could be compared with, say, a similar survey, taken at a point in the future whereby the terrain has been progressively deformed. By subtracting one from the other (and finding the Boolean 'difference') you could easily measure the amount of positive/negative change.

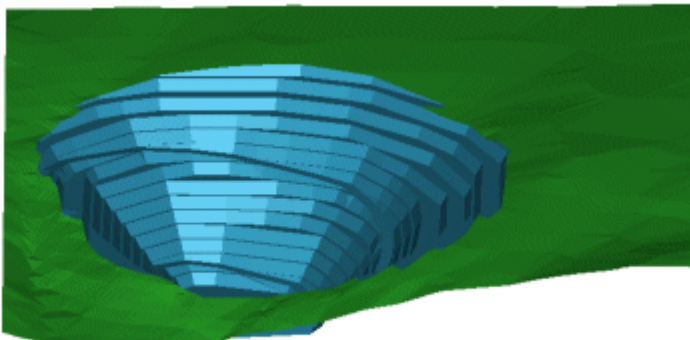
To demonstrate an example graphically, the following image represents an initial surface topography file:



This surface file will be used to close the excavated area that is seen in a surface file that includes the result of several years of open pit working. This example will compare start and end of mine-life scenarios using a simple (low-resolution) data set. The same process can be applied to measure overall tonnages cut between bench phases and/or pushbacks of an anticipated mine schedule, for example, relying on

proposed excavation workings as a basis for calculation.

Continuing the example above, the 'end of operations' image could look like the image shown below right.



Calculating the volume of the excavation is a simple matter of creating another wireframe object where the pit operations have taken place, and calculating its volume using proprietary Studio 3 processes.

There are several ways to create compound wireframe objects, and the method used will depend on the type of wireframe(s) in memory and, in some cases, personal choice. The method outlined here shows a simple way of determining the volume of a wireframe object that is applicable when the volume can be

derived directly from the merging of two objects (as in this case, object 1 will put a 'roof' over the open pit excavation in object 2).



The open pit design shown in this section is purely hypothetical, and was designed using Studio 3's extensive open pit design commands (including the **QUICKPIT** process, which allows open pit designs to be created from the top-up or bottom-down according to a set of initial parameters). For more information on open pit design in Studio 3, please refer to your online open pit design tutorial, accessible from the **Help** menu.

With both objects accessible, you will need to create a compound object, using **Wireframes | Wireframing Processes | Add Two Wireframe Files**. This accesses another Studio 3 process – **ADDTRI** which will require a total of six fields to be defined:

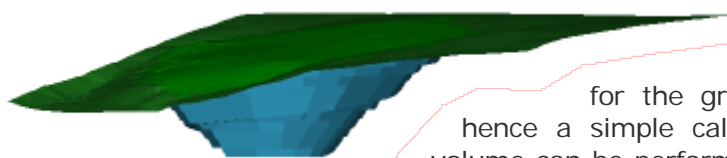
- The wireframe triangle file for object 1 (can be an existing file either in memory and/or on disk)
- The wireframe points file for object 1 (as above)
- The wireframe triangle file for object 2 (also, must be an existing file or object)
- The wireframe points file for object 2 (as above)
- The name of the wireframe triangles file to be created.
- The name of the wireframe points file to be created.

Specify these six mandatory fields and click **OK** to create the new project file.



Note that the created file is not drawn to the screen as yet – this is standard behaviour for all Studio 3 processes – a project file is created, but not loaded into memory. It must be loaded by dragging it from the **Project Files** control bar into the **Design** window, for example.

Once in memory, the new object can be viewed by switching off all views of the two original objects by opening the **Sheets** control bar, and the *Design* window sub folder. Expand this tree until you see the overlays for object 1 and 2 and clear the check boxes next to those descriptions. Redraw the display (type 'rd' with cursor in the **Design** window) and the combined object is drawn on its own (a 3D rendered view of the example objects is shown):



As the topographical surface was created from two open objects, the resulting volume for the green section of the object is zero, hence a simple calculation of the wireframe object's volume can be performed. One method for the calculation of a wireframe volume is to use the **TRIVOL** process (**Wireframes | Wireframing Processes | Calculate Wireframe Volume**). This Studio 3 process requires that you define (as a minimum, other parameters are supported) the input wireframe and points file and, on the *Parameters* tab, a **ZBASE** value, that defines from which Z location ('height') on the wireframe a volume calculation should take place.



In Studio 3 process dialogs, if a field requires a file for input, you can display the **Project Browser** dialog quickly by double-clicking the relevant field.

When all required parameters have been specified, clicking **OK** generates a report in the **Command** window to show the results. Note that one of the parameters supported for the **TRIVOL** process is the ability to specify a perimeter file. This file, a string file, is used to define an inclusion/exclusion zone when calculating the volume of the wireframe file.

```

>>> SUMMARY OF WIREFRAME PROPERTIES <<<
=====
Total volume (above -40.0) = 603965784.87
Volume above -40.0 = 603965784.87
Projected lower area = 0.00
Projected upper area = 2737801.25
Projected vertical area = 0.00
Total surface area = 2979379.00
Minimum elevation = -40.00
Maximum elevation = 255.17
Minimum X co-ord. = 5610.00
Maximum X co-ord. = 6780.00
Minimum Y co-ord. = 4600.00
Maximum Y co-ord. = 5770.00
Minimum surface dip = 0.00
Maximum surface dip = 60.55
Number of triangles = 8745

```

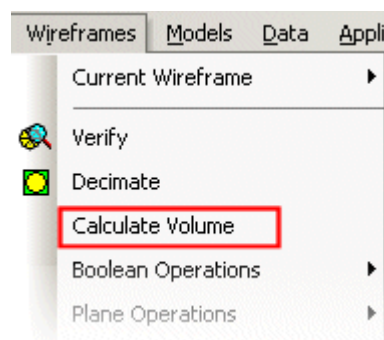
The two initial results represent both the total volume and the volume above the specified **ZBASE**. In the example shown on the left, both **ZBASE** and the lowest limit of the wireframe are the same, hence the identical figures.

As shown, **TRIVOL** also displays other results including the total surface area, the minimum and maximum values for X and Y

axes as well as the lowest and highest dip values of the specified wireframe object.


Calculating a fill volume is a very similar process; the area to be measured results from the merger of 'before' and 'after' objects.

Another way of generating a volume that represents the difference between two objects is to use the 'Difference' Boolean operation as described in Chapter 5. Studio 3 also provides another method for volume calculation, which provides the same results as **TRIVOL**, but with a simpler dialog. The **Calculate Volume** command can be found immediately below the **Wireframes** menu and serves to provide a summary report on the selected wireframe's volume.



Selecting this command displays the **Calculate Volume** dialog. All dialogs in this area of the Design menu system, including Boolean and Plane operations, look similar; they can be divided into two basic sections – the top section of the window is used to specify either a single wireframe object (or two drop-down lists provided to select two objects for the purpose of a Boolean operation), and the lower area of the dialog is used to specify properties for the command in question.

Wireframe objects can be selected from the drop-down list or by using the Data Picker tool that can be found on the right of the object list(s). In most cases, you can also verify wireframe data prior to running the operation.

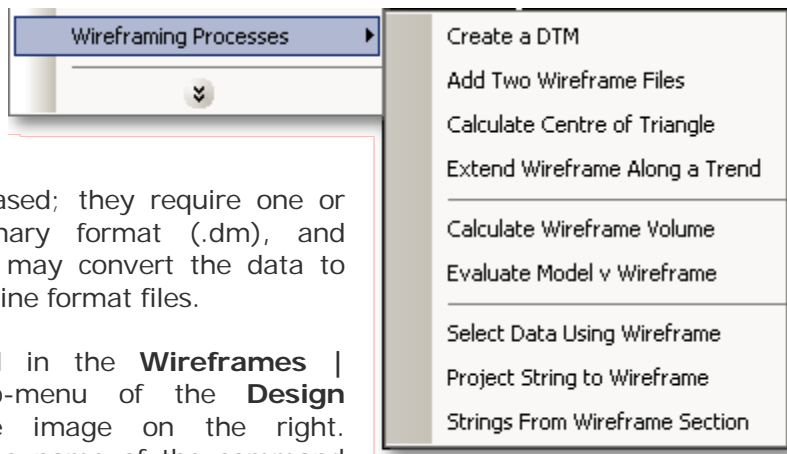
 You can also select a wireframe object prior to running a command – according to your wireframe data selection settings. This will then cause the selected wireframe to be shown in the first (or only) object drop-down list when a wireframe dialog is opened.

Volume calculation can be performed on a single, closed wireframe object, or it can be calculated using a DTM as a 'cut-off point' such that a volume is measured above or below it.

Once run, the command reports results according to the settings specified.

11 OTHER DATAMINE WIREFRAMING PROCESSES

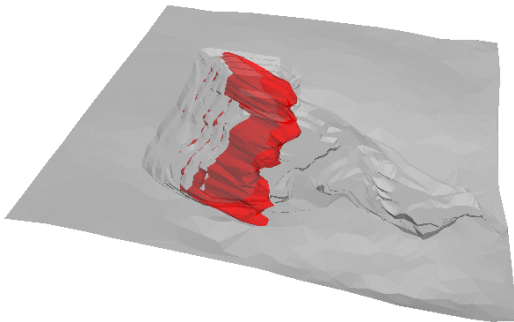
Other than the commands already mentioned, several wireframing commands are available as 'Datamine Processes'.



Studio 3 processes are file-based; they require one or more files in Datamine binary format (.dm), and depending upon the process, may convert the data to create one or more new Datamine format files.

These commands are found in the **Wireframes | Wireframe Processes** sub-menu of the **Design** window, as shown in the image on the right. Alternatively, you can type the name of the command into the Command Line to open the relevant dialog, as shown in the image below:

All process dialogs have a **Help** button that can be used to find out more information on the parameters that are associated with the command in question.



As with all Datamine processes of this type, each function is controlled using a similar "files, fields and parameters" dialog.

Each command is covered in detail in your Studio 3 online Help, but as a quick summary:

Create a DTM

Runs the Studio 3 process **SURTRI** (create-dtm)

Generates a triangulated digital terrain model from perimeter, string and/or point data, subject to optional boundary and string edge constraints.

Add Two Wireframe Files

Runs the Studio 3 process **ADDTRI** (add-two-wireframe-files)

This command joins two pairs of wireframe model files to create a single pair of output files.

The names of the first two wireframe files may be the same as the output file names. This process combines the files correctly, and renumbers the *PID* numbers so that the output wireframe files may be subsequently processed. Normally some sort of **ZONE* field in the input triangle files will exist, so that the structures in the different files may still be identifiable in the combined triangle file.

Calculate Centre of Triangle

Runs the Studio 3 process **COGTRI** (calculate-centre-and-orientation-of-wireframe)

This process calculates the centre point and the dip and dip direction of each triangle in a wireframe. Output from the process consists of a wireframe (a wireframe triangle and a wireframe points file), and a point data file. Both outputs are optional, although at least one must be defined.

Extend Wireframe Along a Trend

Runs the Studio 3 process **WFTREND** (extend-wireframe-along-trend)

This process creates a DTM wireframe surface which can be extended beyond the data limits by a specified distance using a planar trend surface. The data limits are detected automatically, and the trend surface is only used beyond the limits of the actual data. The input data is a set of points which should be sorted.

Calculate Wireframe Volume

Runs the Studio 3 process **TRIVOL** (calculate-wireframe-volume)

This command reports volume and other statistics of a wireframe model.

You can report on a zone-by-zone basis whereby several statistics are output including the minimum and maximum coordinates in a particular direction, the projected vertical area, number of triangles, the volume above a particular zone limit etc.

>>> SUMMARY OF WIREFRAME PROPERTIES <<<			
>>> FOR ZONE		3.00	<<<
=====			
Total volume (above	277.5)	=	22226.89
Volume above	150.0	=	740009.83
Projected lower area		=	0.00
Projected upper area		=	5631.44
Projected vertical area		=	0.00
Total surface area		=	16894.31
Minimum elevation		=	277.46
Maximum elevation		=	544.10
Minimum X co-ord.		=	6247.36
Maximum X co-ord.		=	6514.00
Minimum Y co-ord.		=	5347.36
Maximum Y co-ord.		=	5614.00
Minimum surface dip		=	0.00
Maximum surface dip		=	90.00
Number of triangles		=	36

To run this command, it is necessary to specify at least an input wireframe points and triangle file. A base elevation above which volumes are computed is also required. For a full list of all optional and mandatory parameters for this command, consult your online Help.

Evaluate Model v Wireframe

Runs the Studio 3 process **TRIVAL** (evaluate-cell-model-against-wireframe)

This command evaluates a standard Studio 3 cell/sub-cell geological block model against a triangulated wireframe model, reporting on the contents of closed wireframes, or the reserves, below or above a single-surface (DTM) wireframe.

TRIVAL generates a results file of reserves from an orebody model and an input wireframe; and optionally produces an output orebody model in which a proportion *MINED* field is updated.

Evaluation is normally by partial block, but can optionally be done on a full-block basis in which any block with a *MINED* proportion greater than 0.5 is marked as wholly *MINED*, any proportion less than 0.5 is treated as zero. Results can be classified by XY, XZ, or YZ planes. The results file can be read by **TABRES** to produce various reserve tabulations.

TRIVAL is just one example of an interactive Studio 3 process; it will require further input from you after the process has been initiated.

Input and Output Models

The input model is in standard Datamine model file format, and may contain cells and sub-cells, in any order; the model does not have to be sorted on IJK to be processed by **TRIVAL**. The fraction of the cell or sub-cell mined may be stored in the *MINED* field of the optional output model file, as a value in the range 0 to 1. The output model file will be an exact copy

of the input file, with the *MINED* field added if it did not already exist in the input file. The output model file may be the same as the input model file (in-place updating) if the *MINED* field existed in the input file.

Balancing Volumes

TRIVAL produces for each pass through the model a balancing volume record in the results file; the balancing volume recorded (advanced users: this is *not* the same as that produced by a similar command - MODRES) is the total volume of the model less the volume intersected by the wireframe zone evaluated.

Densities

In computation of tonnages from volumes, any *DENSITY* field within the input model is used. If the *DENSITY* field exists, then non-modelled regions will use the default value of the *DENSITY* field. If the *DENSITY* field does not exist, then an overall density may be supplied by use of the *@DENSITY* parameter. If this was not supplied, then the density used is 1.0.

Passes through the Model File

The process reads in one wireframe zone at a time (as identified by the *ZONE* field if present in the triangle file). For each wireframe zone TRIVAL reads through the entire model and records its evaluation against each separately. If successive zones overlap, or the input model already contains *MINED* values greater than 0, there are two ways in which the evaluation may be recorded in the *RESULTS* file and the output *MINED* field.

```

>>> STARTING EVALUATION PASS 1 <<<
>>> 352 TRIANGLE DATA RECORDS LOADED
>>> START OF PASS THROUGH INPUT MODEL
All 7365 input model records processed
Writing results to file.
>>> 14 Records in File C:\Data\Standard\results.da <<<
>>> 14 Records in File C:\Data\Standard\results.da <<<
>>> END OF PASS THROUGH INPUT MODEL
>>> 14 Records in File C:\Data\Standard\results.da <<<
>>> TRIVAL Complete <<<

```

These two options are controlled by the supplied value of the optional *@INCRMENT* parameter. Either it will be considered that all mining is additive if *@INCRMENT* is set to 0, (i.e. that there are no overlapping zones), in which case any cell already partly mined will have the full volume of the current evaluation added to the amount mined - subject to a maximum proportion mined of 1.0. Alternatively, if *@INCRMENT* is set to 1, it will be considered that mining is incremental, and that successive zones are to be considered as mine expansions, and overlap totally. In this case, a partly

mined cell will be output with a proportion mined which is the maximum of the previous and current proportions. In this case, naturally, if the current proportion mined is less than the previous, there will be no evaluated volume in the results file for the cell.

The input wireframe triangle file must contain an *SID* (Surface IDentification) field. If an 'upper' digital terrain model is supplied (with *SID* values all +1) then the model created will fill cells below the surface. Inversely, if all *SID* values are -1, then cell filling will be done above the surface. For solid wireframe models (in which *SID* values of +1 represent upward facing surfaces and -1 represent lower facing surfaces), cells are filled inside the wireframe. Any number of separate zones may be handled within one run of **TRIVAL**, provided they are all held within the same pair of *&WIREPT*, *&WIRETR* files. The *SID* field is produced automatically by *TRIANGR* (set to +1 or -1 at user option).

Select Data Using Wireframe

Runs the Studio 3 process **SELWF** (evaluate-cell-model-against-wireframe)

This command selects records lying inside/outside wireframe models or above/below triangulated (DTM) surfaces.

Records of a file which have X, Y and Z co-ordinates lying above/below DTM surfaces or inside/outside a wireframe model are copied to an output file. The wireframe model can contain multiple zones, but a point will only be selected once if it falls within more than one zone. The wireframe triangle file does not have to be sorted by zone.

```
Space reserved for strings of up to 1024 points each.
CHECKING TRIANGLE VERTICES

STORING TRIANGLE VERTICES

STORING WIREFRAME TRIANGLES
```

By default, points falling on the wireframe surface are included in any selection. The *EXCLUDE* parameter can be used to exclude them.

The *PLANE* parameter can be used with vertically oriented

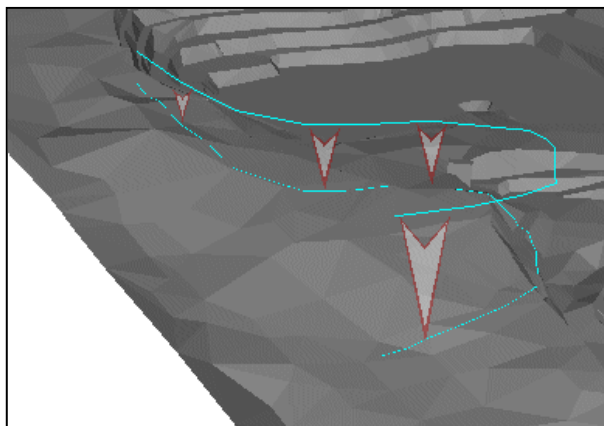
DTM surfaces to allow selection of points to the east, west, north, or south, instead of above or below. This parameter can also be used with steeply dipping tabular solid wireframes to speed up the point selection. In-place operations are not permitted (meaning, the file that is output must be different to the one used for input).

A DTM surface is one which contains no vertical or overhanging portions when viewed from a position perpendicular to the plane specified by *PLANE*. A warning will be given if a DTM surface contains a vertical face, but overhangs will *not* be detected. Holes (missing triangles) in a solid wireframe will generate a warning, and so will duplicate triangles. Double duplicates will cancel out and thus not be detected. The parameter *TOLERANC* is applied in a direction perpendicular to *PLANE*, (and not perpendicular to the surface of the wireframe). If attribute fields are specified, their values are taken from the first record in the wireframe triangle file for each zone, or from the first record in the file if no zone field is specified.

Project String to Wireframe

Runs the Studio 3 process **PERDTM** (project-string-to-wireframe)

Project 2D perimeters onto a DTM surface, creating a 3D string.




This command creates strings or perimeters defining the intersection of a wireframe model with either a single plane or with a family of planes.

You will need to define the perimeter 2D string file, and the points and triangle files relating to the terrain model. An output file name is also required.

Values for additional fields in the input perimeter are taken as constant for the string and therefore carried across to each

point of that string in the output perimeter.

It should be ensured that the perimeter lies entirely within the area defined by the DTM wireframe, otherwise points lying beyond will have undefined ZP values.

 A similar effect can be performed 'in-memory' using the **Design | Project | String to Wireframe** command, although in this situation, the current string object is appended with additional string data.

Strings from Wireframe Section

Runs the Studio 3 process **WIPEPE** (strings-from-wireframe-section)

This command requires, as a minimum, a wireframe points and triangle file (input) to be defined, and the name of an output file. You can also specify a view definition file if multiple strings are required from multiple section planes.

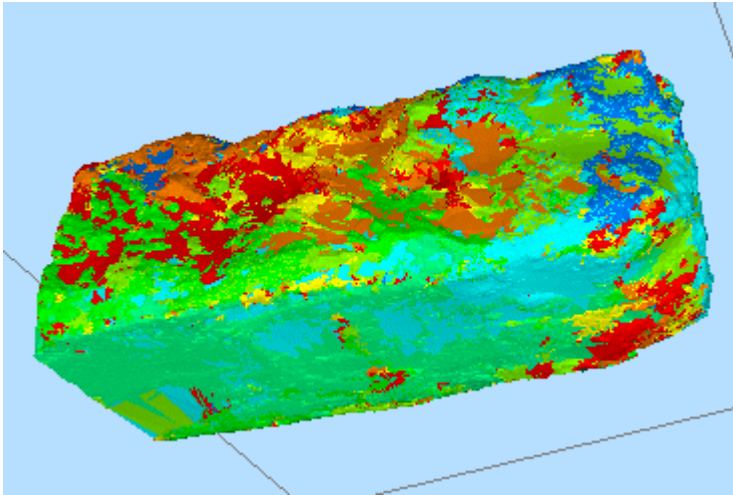
If no section planes are defined in the dialog (through the definition of an input section definition file) you will be required to define a basic direction from one of the following options:

1. A vertical or inclined section defined about a centre point (XC, YC, ZC, DIP and AZI parameters).
2. A vertical section defined by 'end points' for each axis (X1, Y1, X2, Y2, Z1 and Z2 parameters).
3. A plan view section defined by a fixed z elevation and X and Y corner points (X1, Y1, X2, Y2 and ZC parameters).

Once the initial criterion has been defined, you can specify the key coordinates of your intersection plane; using XC, YC and ZC parameters, with *Azimuth* and *Dip* for option 1, the start and end locations of the section in 3 directions for option 2, or the corner points using a single Z elevation value.

For more information on these, or any Studio 3 processes, please refer to your online Help.

12 CREATING WIREFRAMES FROM POINTS



Studio's "Wireframe from Points" command initiates a powerful function, providing access to a variety of tools to allow point data to be surfaced.

You can control the resolution and application of the resulting mesh. Granular control is provided over how the mesh triangles are generated in relation to the underlying source data.

The **Wireframe from Points** command will receive either a Datamine binary file (on disk - does not have to be loaded into memory) or a comma-separated file listing point locations.

On completion, a single wireframe mesh will be created (as a pair of physical .dm files on disk, and a single wireframe object in memory), honoring the positions of all points present in the original data file. Note that if your file contains multiple point "clouds", and independent surfacing of each zone is required, it will be necessary to filter this file and create independent source files before you start. In summary: this command will produce a mesh taking into account all of the points of the input data alongside other function parameters.

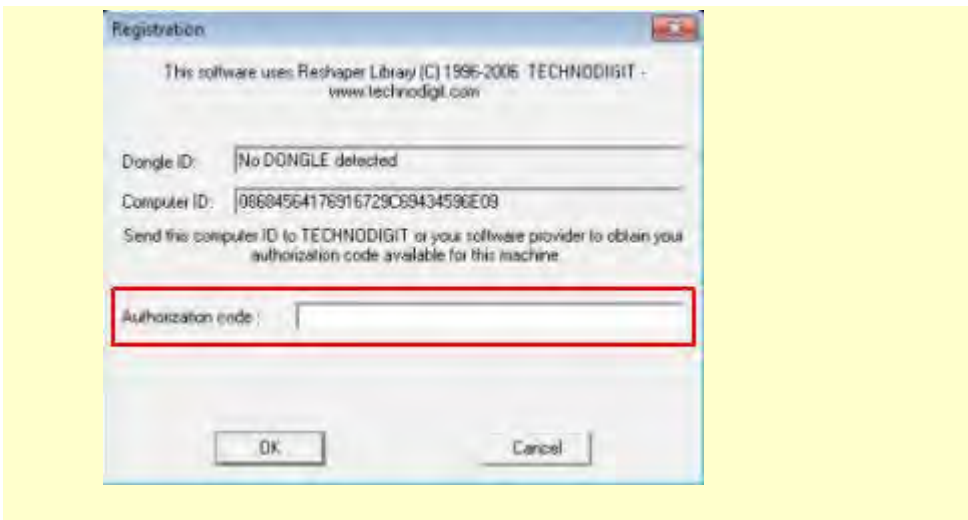
This command can be used to generate both closed wireframe volumes and digital terrain models (DTMs). In the latter case, you will be able to specify how a wireframe mesh is 'laid' onto the imported points, e.g. from which direction.

This command is intended for use in conjunction with point cloud data that is of sufficient resolution to indicate a surface or volume.



A similar effect can be performed 'in-memory' using the **Design | Project | String to Wireframe** command, although in this situation, the current string object is appended with additional string data.

The create-wireframe-from-points command utilizes a 3rd party technology that requires the presence of a separate dongle and license. This is in addition to a "Point Hulling" license in Studio. Your [CAE Mining representative](#) will be able to supply you with the appropriate hardware dongle and activation key to enable you to access this functionality. To enter your activation key, launch the command and enter the relevant code in the Authorization code field shown below:



The Wireframe from Points Dynamic Dialog

The **Wireframe from Points** dialog will only display parameters that are relevant to your chosen meshing method. For example, if you elect to perform a two-pass operation, you will be able to select additional parameters to control how data points are respected during meshing than if, say, a more simple one-pass operation is chosen. In addition, you will be able to specify how 'holes' are detected and filled during the meshing operation.

Similarly, if you are creating a DTM - as opposed to a closed volume - you will be able to determine the direction from which the meshing will be applied (think of laying a cloth over the points in a particular direction).

Data Resolution and Processing Times

The **Wireframe from Points** command can, potentially, generate very large data files. The settings used to control the resolution of resulting wireframe data is entirely dependent on a) the resolution of the input points file and b) the settings that are made in the meshing dialog. In particular, the *Average distance between points* parameter has a potentially significant effect on how much data is produced as a result of the command and the time taken to generate these files.

Smoothing

Smoothing is another useful feature of the **Wireframe from Points** command. If active, you can smooth the resulting mesh surface over one or multiple passes. There is a choice of two different smoothing methods, which are described in detail on the relevant page (see links below).

Again, smoothing a wireframe adds to the processing time, but can be very useful to remove unwanted 'noise' from the original data set (in the case of a topography data file, say, from LIDAR) - however - if used excessively, smoothing can also deviate the resulting mesh away from the original source data, making it theoretically less 'accurate' or less representative of the real-life entity represented by the point cloud.

Wireframe Preview

The key is to find the right balance of parameters for the data file in context. This is made easier by the **Wireframe From Points** dialog's Preview facility, which allows you to generate a view of data based on the current parameters without dismissing the dialog or halting the command. Hence, you can experiment with parameters until the optimal mesh is generated.

Note that the **Wireframe from Points** command generates physical files on disk, even as part of the preview process, which are then loaded automatically into Studio. This means that if you simply wish to generate a series of mesh files without additional processing, there is no need to perform additional data saving commands.

Dialog Structure

The **Wireframe from Points** dialog can display the following screens. Note that all screens are listed, regardless of parameter selection; not all screens will be available for all meshing operation types:

Input File and Output Wireframe: the initial screen of the Wireframe from Points dialog is used to determine the input points file, output directory and file name and the type of wireframe to be created (volume or DTM surface).



Where large points files are being loaded, there is a small delay when moving from the first to the second screen; this occurs because the Studio system is extracting summary information about the file, to be presented on the **Noise Reduction and Hole Management** screen (see below).

Noise Reduction and Hole Management: this screen is used to set up the initial 'scope' of meshing; the resolution of the resulting mesh, how holes are managed and other basic parameters.

Mesh Generation Method and Hole Management: only visible if the Meshing in two steps option was selected on the previous screen, this screen is used to determine how/if new points are created during meshing plus additional controls over how wireframe gaps are handled.

Vertices and Triangles: this screen controls how/if wireframe smoothing is applied during mesh generation.

DTM settings: if a DTM wireframe mesh is being generated, this screen determines how the surface is calculated in relation to the underlying data points.

Configuring Input and Output Files

The first panel in the Create Wireframe from Points dialog is used to define the input points file (either a Datamine binary file or delimited text file) that represents the data you wish to surface when using Wireframe from Points. Once the relevant details have been defined, click Next to access the Noise Reduction and Hole Management dialog.

Set up your *Input* (must be a Datamine points file or CSV text file):

- If processing a points file in Datamine binary format, default data columns should already be present in the source file (as a minimum, the fields XPT, YPT and ZPT must be found).
- If importing an external CSV file, you will be required to map data columns in the external file to XPT, YPT and ZPT fields.

You can select to generate an encapsulated, *Closed* wireframe volume or an open Digital Terrain Model (DTM) surface.

If you are going to generate a closed volume, the **Wireframe from Points** wizard will display the following screens thereafter:

- Noise Reduction and Hole Management
- Mesh Generation Method and Hole Management (if two-pass meshing is selected)
- Vertices and Triangles

If a DTM surface is selected, you will be shown the following screens thereafter:

- DTM Settings
- Vertices and Triangles

More information on this panel can be found in your online help.

Reducing Noise and Managing Holes in your Data

The second panel in the wizard is used to define parameters to resolve the issues of noise and holes in the source data, and contains the following sections:

Point cloud summary details: this section of the screen displays the number of points found in the specified points data file. It also shows a calculated value representing the average distance between each point and its nearest neighbour.

Noise Reduction: see "Understanding Noise Reduction", below, for background information on this set of options:

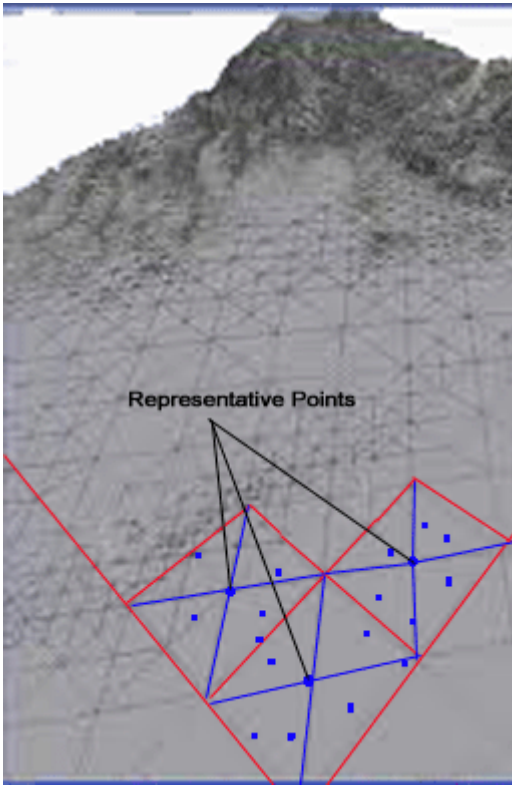
Meshing in two steps: selecting this option will force the **Wireframe from Points** command to generate a mesh, then scan it to make additional adjustments to ensure points are repositioned in relation to a standard deviation of error. The second, statistically-based adjustment is used to generate a surface that follows the trend implied by the source data points. The **Average distance between points** field is used with this selection to determine the resolution of the meshing grid that will be used to construct the resulting wireframe.

Selecting this option enables an additional screen for this wizard, displayed when **Next** is clicked. The **Mesh Generation and Hole Management** screen allows you to control precisely how this second pass of the meshing operation is performed.

Regular Sampling: regular sampling involves the projection of a regular grid onto the modelled volume or surface. In this context, the Average distance between points field represents the size of the mathematical grid interval.

In each square of this grid, the Wireframe from Points command calculates a small piece of surface, and determines the point that is closest to this surface and declares this point "representative" of the grid square. This point is kept inside the mesh and then becomes a **vertex** of the mesh. In general, the surfaces and volumes achieved with this technique will have an approximately constant triangle size, equal to the grid interval.

To achieve a reasonable result with this option it is necessary that a number of points are present inside each element of the grid. Thus, a dense point cloud is preferable and more likely to generate representative results. The more points in a cloud, the more the probability of finding a "representative" point for each grid square, meaning the result mesh triangles will align more closely with the source data.



You should find that the computing speed is *much faster* than in the other options. Note that the resulting mesh will inherit the following properties:

- A higher regularity of triangle shape/size.
 - The triangle 'count' in the resulting mesh will generally be low compared to the original source point data file resolution, however, this depends on the *Average distance between points* setting.
 - Small radii are better defined with small triangles because the shape is better approximated in this case.
 - Flat zones are less "disturbed" with large triangles being used to generate the surface in this region.
 - Holes (see further below) are well recognized, but contours are at risk of being "chopped". Why?
- When the regular grid is projected, there are very few chances that the interval of the grid corresponds exactly to the contours. In the majority of the cases, the software will approximate contours by adding "stairs" to meet the relevant elevation of each triangle.
 - To calculate the small piece of surface in each grid element, a minimum number of points are needed to calculate a truly representative point. Unfortunately, often, the point density is smaller near the borders of an object, resulting in the loss of data in these areas.

No noise reduction; deviation error: selecting this option forces the **Wireframe from Points** command to reposition (not ignore) points that fall outside of the specified deviation distance, where that distance is measured from the theoretical surface implied by underlying data points (which itself represents a 'best fit' surface for the data provided). Where the specified deviation error is low, there is a chance that noise reduction will be insufficient to produce a satisfactory result, whereas conversely, if the deviation error is too high there is a chance that triangles may be repositioned erroneously.

Managing Data Holes

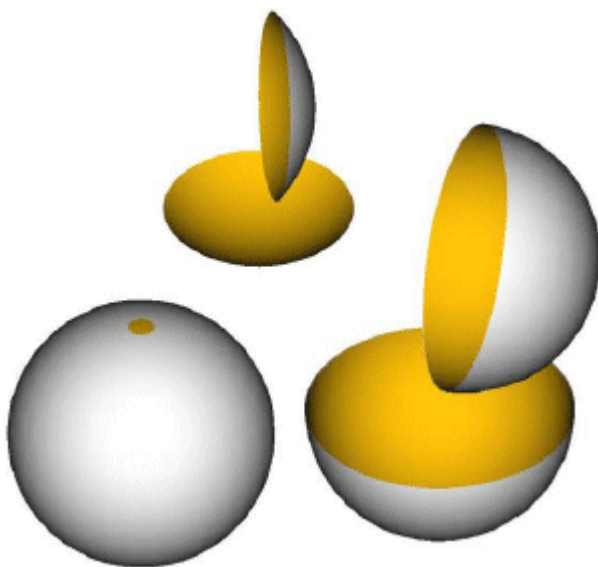
The **Wireframe from Points** facility can automatically detect holes and patch them as part of the meshing process, however, the rules under which this 'patching' takes place needs to be defined up front.

It is also important that the data source file that is being surfaced should contain enough points to accurately represent the surface or volume being modeled. Data can be interpolated as part of meshing (although the means by which this is controlled is only available where a two-pass process is performed - see below), however, interpolated data will only be used to continue an existing trend - if important data points are not captured, they cannot be modeled.

These options control how data 'gaps' are handled when creating a mesh. Some holes may be expected in the output wireframe so this section of the dialog allows you to control what defines a 'hole' and what happens when one is detected.

Once a theoretically correct point set has been calculated from the base points file, the next step is to connect them together to form a mesh. Depending on the nature of the 'shape' of the point data, a number of parameters need to be stipulated in advance. The *Average Distance Between Points* parameter, for example will determine how far apart the resulting mesh vertices will be, and it is important that, where holes exist in the original data, and these holes are meant to be there, they are not inadvertently filled in with unwanted vertices.

The concept of "hole" is important here; imagine, for example, a hollow sphere with a small hole on the surface. The edge of this element is a circle that you would consider a "hole". Imagine that the dimension of this hole increases until it expands to reach the diameter of the sphere. The hole has now been converted to an "external border". This concept of "hole" or "external border" is thus only a question of vocabulary, but from a strictly mathematical point of view, they represent the same thing.



Spheres pierced with a small hole and open hemispheres are geometrically indistinct

Image courtesy of Technodigit 3DReshaper® Tutorial documentation

Hole Detection: if selected, an attempt will be made to detect all of the holes in the specified data. You will also need to specify a **Triangle Size**, which represents the largest size of triangle that can be constructed during meshing - this must be lower than the smallest size of hole that can be detected. For example, if you specify a triangle size of "400" units in area, any hole that exists in the data that is less than 400 units in area will be surfaced, and effectively removed.

Note that the **Triangle Size** is a **maximum** setting - surfacing will still give rise to smaller triangles in order to preserve the original data geometry (and in accordance with other surfacing parameters).

Try to keep only the external border: the software will search the free edge that contains the greatest number of triangles, and it will consider that this free edge is the "external border". If it finds other contours, they are considered as "holes" and they will be filled.

Try to create a watertight mesh: if selected, the meshing processing will attempt to create a closed polyhedron. This will not always be possible (for example, where the triangle size maximum is smaller than a data hole).

General Guidelines for Setting Noise Reduction and Hole Management Parameters

The **No noise reduction; deviation error** is generally more effective where there are a relatively low number of points in the underlying data set (for performance reasons) or where a high degree of accuracy (e.g. close correlation between the ideal theoretical surface and the actual mesh) is required.

In general, noise reduction will be more efficient when the number of points inside the cloud is relatively high (due to a greater probability to choose the most representative point in the right position).

The use of the deviation error during meshing phase 1 will often fill holes, it is thus recommended to use this meshing strategy for the shapes **without** holes or where hole filling is not important and won't detract from the usefulness of the resulting wireframe.



Concerning noise reduction, a larger triangle size will give rise to more noise reduction. On the other hand, an excessively large triangle size may result in the unwanted loss of details.

The larger the triangle size on the edges of the resulting mesh, the more contours will be smoothed correctly, but beyond a certain limit, some holes can be filled.

The selection of **Regular Sampling** and keeping the triangle size by default often gives a good initial preview and starting point from which to refine your meshing parameters. This will depend on the geometry of the point data file, its density and overall dimensions.

The selection of **Meshing in two steps** makes it possible to optimize the result by automatically launching a second meshing phase.

The **Wireframe from Points** command can be configured to manage source data noise and data "holes" in a variety of different ways.

This configuration is necessary as the function will not always be able to interpret data gaps or data capture errors automatically as being either intended or as part of incomplete data gathering. A reasonable example of this would be surfacing of a point cloud taken of the far end of a development drive; this data has one large (and very necessary!) hole at one end - this is the open aspect of the drive 'finger'. To surface this particular hole would be undesirable. However, there may be other data points that are missing from the original source.

Understanding Noise Reduction

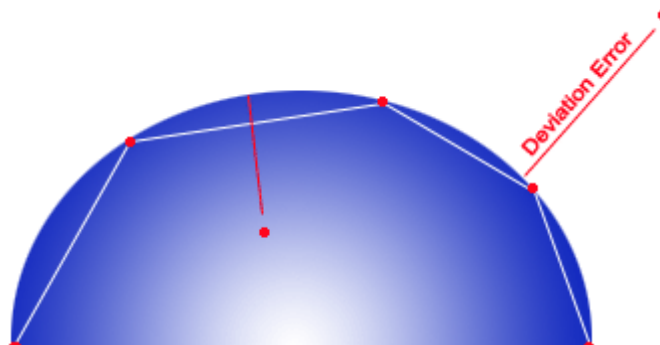
The corollary of insufficient point data is excessive resolution, which can either be present as a result of inappropriate scanning parameters (e.g. data resolution is very high and a lower resolution would generate an acceptable result for, say, volume evaluation) or caused by measurement noise.

Noise represents data points that sit outside the normal distribution of points that represent the surface that was originally scanned. The goal of a meshing operation is to make use of

the "good" point whilst automatically detecting and ignoring the "bad" data. Sometimes, this is a simple calculation but in other cases the determination of noise may require additional input.

Up to two basic processes are used to determine whether data points are good or bad, depending on the selection that is made in the **Noise Reduction and Hole Management** screen:

- **Geometric correction:** a scan is made of the surface to determine a theoretical surface that best fits points within a specific distance of that surface. Where points are found to deviate beyond this distance limit (the 'Deviation Error'), those points will be migrated back to the theoretical surface (i.e. repositioned). The important point here is that the deviation error is measured from the theoretical surface implied by the mesh triangles and not the mesh triangle faces themselves:



- **Quality correction:** under this criterion, points will be ignored if they fall outside a specified distance from the theoretical surface, reducing the resolution of the mesh (at that stage in the process).
- One or both criteria will be considered in mesh generation. The actual process(es) performed is dependent on the selection made in the Noise reduction area of this screen. The following table represents how the **Wireframe from Points** command is used to interpret point data in conjunction with the four radio buttons found in this area:
- **Meshing in two steps:** quality correction (point removal) only will be performed during the first pass of meshing. The second pass will include both geometric (point repositioning) and quality correction.
- **Regular Sampling:** both quality and geometric correction are performed for this one-pass meshing method.
- **No noise reduction; deviation error:** this method will only reposition existing points - no "noisy" data will be removed.
- **Keep all the points:** use this setting with care: field data can be a prohibitively high resolution and littered with extraneous data points. It is important to attempt to intercept and manage noise when generating a mesh. Electing to utilize all of the points in the source file may actually create a less 'accurate' version of a modeled surface than if noise reduction were performed. This option also has the potential to increase the required processing time and/or system resources so should only be selected if there is high confidence in the integrity of (and requirement for) the full point data set.

Selecting a Mesh Generation Method

The **Mesh Generation Method** panel is only displayed if a *two-pass meshing operation* was selected in the **Noise Reduction and Hole Management** screen. During this process, quality correction (point removal) only will be performed during the first pass of meshing. The second pass will include both geometric (point repositioning) and quality correction.

The first pass of meshing will generate a 'coarse' mesh whereas the second pass will interrogate that intermediate mesh and refine the deviation error adjustments. The deviation error refinement compares the point cloud and the intermediate mesh and then improves the meshing accuracy by re-meshing with other points if the accuracy is found to exceed the expected deviation error that is permitted. This pass also refines the holes and contours and if necessary fills holes.

The philosophy of this command is to start from a coarse mesh where all the details are roughly visible. Most of the time, the command "Direct 3D mesh" fits this need. However, in some situations, it may happen that you need to make first a deviation error refinement with the "Take points of the cloud" method. For example, if you have a shape, which is mainly flat and some small stripes or some corrosion in the middle, there are some high chances that you will need to use "Take points of the cloud" method first and then the "interpolate new points" method to get a good continuous shape.

The rough mesh is compared with the reference point cloud and some modifications are done to get a continuous surface in the middle of the measurement data range.

Available Methods and Parameters

There are several parameters available for controlling precisely how the second pass of meshing is performed; the following two mesh generation options are provide different, each with their own set of control parameters:

Interpolate new points: this option is used to refine the intermediate mesh by increasing its resolution, adding new points between existing points to allow the resulting mesh to more closely correlate to the raw point data.

When you have a dense point cloud, the interpolation of new points with a deviation error gives you the best compromise between:

- The **accuracy** of the result. The goal is to create a continuous surface, according to a deviation error, in the middle of the spread of data measurement points.
- The **smoothness** of the result. It is generally not required to smooth the result that you obtain when meshing with the **Interpolate new points** method.

This option allows you to define the following parameters:

Refine with points evenly spaced: in this case you will have to set the **Maximum space** value which is the space between two points on the final mesh.

Refine with deviation error: if selected, a mesh of differing triangle size will be created in an attempt to match the source data as closely as possible. To enable deviation error refinement, you will have to give the *Deviation error* to respect the *Maximum number of triangles* (very useful in managing the resolution of the resulting mesh) and the *Minimum triangle size*.

Selecting this option generates a mesh adapted so that smaller triangles are created in the more detailed areas, where there is a relatively higher incidence of steep position changes between neighboring points.

Note that if the deviation error that has been set already encompasses the position of all points in the source point data file, a second meshing phase will not be performed (there would be no point - the goal of the second pass is to ensure all points honor the deviation error). To put this another way; a re-meshing will be made only if the deviation error is not reached.

Take points of the cloud: selecting this option will force the meshing algorithm to determine the best mesh vertices (that is, those that correlate to the original data but are within a preset deviation error) by considering the position of the original source data and, for each point, the distance between it and the intermediate surface. With this option, new data points will still be added, but they will be added in reference to points found in the original data (e.g. not interpolated between existing intermediary mesh vertices).

Choosing this method is often appropriate where:

you want to generate mesh triangles of roughly the same size. This is often needed by finite element analysis, for example.



When you have a noisy point cloud.

The **Wireframe from Points** command makes the comparison between the point cloud and a coarse mesh and computes the “best” points at regular spacing.

Note that the number of triangles may be very high if you enter a small value for the parameter **Maximum space**.

This option allows you to define the following parameters:

Deviation error and NO noise reduction: this will adjust the intermediate mesh surface by considering every point that is found within the original data set and recalculating the position of points that exceed the *Deviation Error* value. This will result in consideration of more data so is useful for capturing surface/volume details that may be removed with other, more general meshing configurations - however - this needs to be balanced with the risk of repositioning noisy data and increasing processing time (particularly if system resources are restricted).

Deviation error with best points only: in this situation, the second pass of meshing will determine the representative points by a method comparable to the way they are calculated with *Regular Sampling*. However, here, the grid has a variable step; the grid cell size is adapted to the local density of the point; where more points exist, a smaller grid interval will be used to generate the mesh vertices and edges because a certain number of points are necessary to calculate each small piece of surface and to keep one representative point in each cell.

In both cases, the command will refine the mesh in order to respect the **Deviation error** specified.



Note that with any of the options above, the **Abberant points filter** option is relevant; some points can be far away from the intermediate mesh (e.g. due to sampling noise). In this case, these points are rejected from computation because they are considered as aberrant. The distance of this filter is set manually.

For all options in this group, enabling the **Local reorganisation** check box will allow the meshing algorithm to reposition additional

points as required to make a 'best fit', increasing data resolution in areas where a particular geometry has to be followed. If the **Local reorganisation** check box is disabled, you will end up with a [regular mesh](#) of equilateral triangles.

Secondary Hole Management: the second pass of meshing provides another opportunity for refining the meshing around (or across) holes in the point data cloud. The effect of the options in this area is largely generic but, in one case, additional parameters are required for the *Take points of the cloud* selection.

The mutually-exclusive options that are available are:

- **No free border modification:** some points can be outside the borders/surface of the intermediate mesh. If this option is selected, points will be ignored if they lie beyond the free border of the intermediate mesh, and the Deviation error distance is exceeded.
- **Extend free border:** Points that are positioned 'outside' of the intermediate mesh's free borders will be taken into account only if this option is selected. New points may be added to extend the surface out of the original border and accommodate the off-shape point(s). In this operation, the software analyzes the points that are out of the existing mesh then re-meshes by creating triangles whose maximum size is imposed.

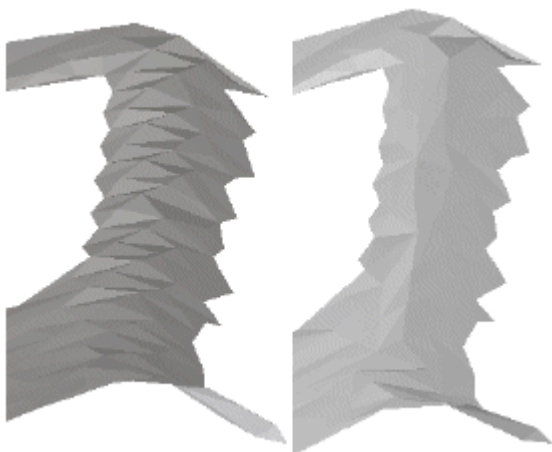
Where the *Interpolate new points* meshing method is chosen, these points will be positioned automatically and the resulting triangle size will honor either the *Maximum space* parameter (*Refine with point evenly spaced* option) or the *Maximum number of triangles* and *Minimum triangle size* parameters (*Refine with deviation error* option).

If using the *Take points of the cloud* option, the additional triangles required to 'breach' the gap between the border and remote data point(s) will be created accordingly to a manually-specified size value.

Refine free borders: if this option is selected, no new points will be created, but the resulting mesh will be extended by repositioning the points of the intermediate mesh.

Managing Vertices and Triangles

This panel controls how wireframe 'triangles' are organized within the resulting mesh. In brief, you have a choice of having locally reorganized triangles or no local reorganization:



If you observe the model on the left (no *Local reorganisation*) you will note that the rounded profile is crossed by triangles that cut the shape to create small chamfers. This is explained by the fact that the triangles are calculated by joining the closest vertices together. The fact that two close points can be on both sides of a fillet or sharp edge is not taken in account.

The model on the right (with *Local Reorganisation* selected), you can see that the mesh has been created in the direction of curvature, limiting the defects in the resulting surface. In both cases, the vertices are identical. The more acceptable result obtained with the reorganized mesh has been achieved without increasing its "weight" in term of triangle numbers.

Wireframe from Points – Worked Example

In the following example, a points file (in Datamine Binary Format) is used to construct the terminal end of a drift.

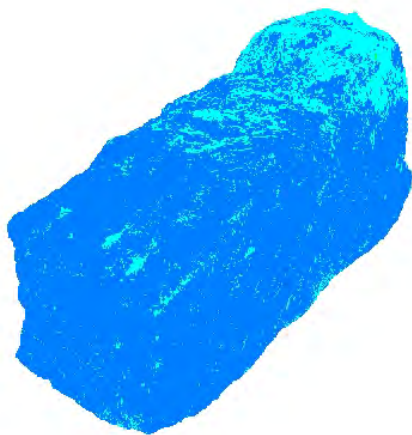
Two runs of the wizard are performed to highlight the effect of some of the available parameters on the resulting wireframe.

Although an underground drive finger is being modelled, the closed volume option will be used and hole management settings are configured to permit the large 'hole' that represents the open end of the data.

The main difference between the two different 'runs' is in the way smoothing and data resolution are applied.

In both cases, an input points file is specified. In its 'raw' form, the points file looks like the following image (VR window display):

In this particular example, the data (laser capture) is very dense – in excess of 5 million data points – there is much more information than will be required in the resulting wireframe (and, in fact, would make the wireframe onerous to use for the purpose of further analysis due to its size if some filtering were not applied as part of the mesh generation).



Before you start, you will need to:

- ensure your Technodigit dongle is inserted into the machine and is registered (using a 3rd party code supplied by your CAE Mining vendor)
- ensure a valid 'Point Hulling' license is available to your system via the License Manager

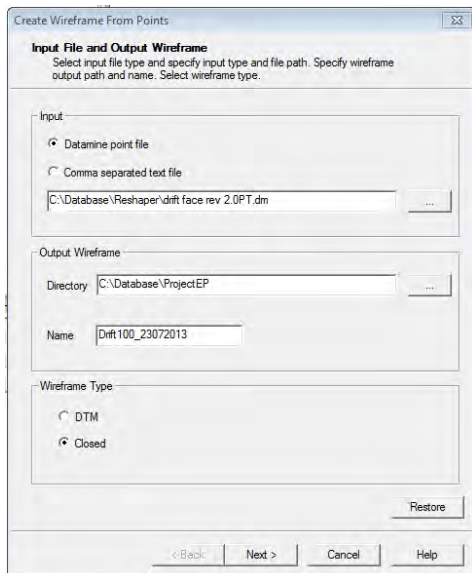
- Studio (MR21 or later) is running

First Run – Coarse Model with Low Resolution Settings

For the first attempt, a rough model is generated for the purpose of viewing basic geometry of the captured data. Holes are not managed and the parameters are mainly those found as defaults.

1. Unload any data that is not required for the meshing operation (frees up memory). Either use the unload-all (ua) command or selectively using the **Loaded Data** control bar.
2. With the Design or VR window displayed (VR will be used in this example), select **Wireframes | Create from Points**.
3. The **Create Wireframe from Points** dialog is displayed, showing the **Input File and Output Wireframe** panel.
4. This example takes a Datamine File (.dm) as the input so the *Datamine Point File* option is selected.
5. The points file is selected and loaded using the browse button on the right.
6. Once the file is selected (it isn't loaded into memory yet), the path to the file is automatically shown and the Output Wireframe *Directory* field is populated – the output wireframe will be created in the current project directory but this can be changed using the associated browse button.
7. A name is entered (the .dm suffix isn't required). In this example, the output file is going to be called "Drift100_23072013". As with all Studio wireframe generation, two files are created to represent the wireframe; points (pt) and triangles (tr).

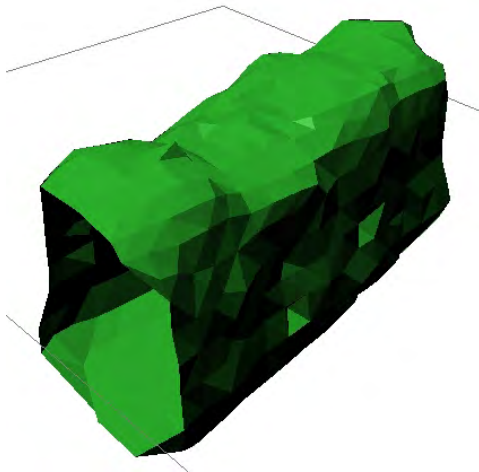
At this stage, the first panel looks like this:



8. Click **Next**. Note that the system now loads the specified file into memory and performs various checks to extract base information, which will be presented on the next screen.

Be patient – this can take time for very large point files. A status update will be shown at the bottom of the panel during this phase.

9. The **Noise Reduction and Hole Management** screen is displayed. This displays summary information about the input file (the total number of points and the average distance between them) and then provides options to determine the method used to mesh the points. It also provides functions to control which areas of the input file are deemed to be 'holes' (used more in the second run of the wizard).
 - a. A simple 'one-pass' operation will be used for this run, so the *Regular Sampling* option is selected.
 - b. The default value for *Average distance between points* is '2.5'. For this run, the default value is left as it is.
 - c. All *Hole management* options are left at their default settings. This means that where an attempt is made to create a mesh triangle over 1000 units in size, the meshing is aborted, leaving a hole.
10. Click **Next** to access the Vertices and Triangles panel. This panel controls *how* the triangulation of the mesh will be performed.
 - a. The **Smooth Wireframe** check box is disabled on this screen to disable any recalculation of vertices or the addition of new points to honour the settings made on the previous screen.
11. Click **Finish** to generate the wireframe output files and load those file into the display window (it is useful to display the result in the VR window at this stage).
12. A progress bar is displayed. Once complete, depending on your project settings, you may be asked to confirm the addition of new files to the project – click **Yes To All** if this occurs.
13. The resulting wireframe is shown and can be rotated/zoomed/panned using the relevant controls:



Note the low resolution of the mesh and the holes showing at various positions. In the next run of the wizard, the resolution will be increased, holes removed and the resulting geometry smoothed.

14. The low-resolution wireframe is unloaded using 'ua' (it is already stored on disk).

Second Run – High-resolution Model with Hole Removal and Smoothing

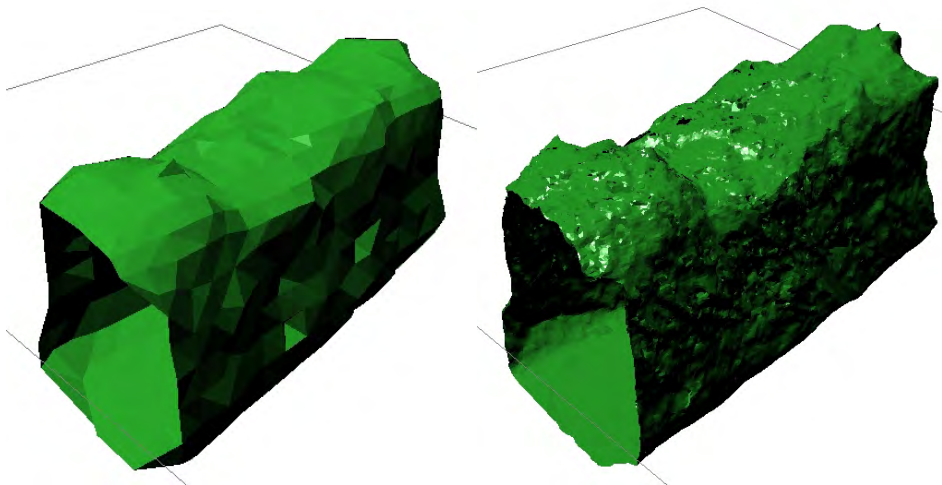
1. Repeat steps 1-7 above but create an output name of "Drift100_23072013_HIRES".

To save time, you can also use the **Restore** button to reinstate the previous settings used – all you need to do then is to rename the output file as above.

2. Click **Next**. The **Noise Reduction and Hole Management** screen is displayed as before.
 - a. As before, *Regular Sampling* will be used.
 - b. This time, the *Average distance between points* box is set to '0.25' – this considerably reduces the triangle size in the resulting mesh. The advantage is a much higher resolution output but it will take longer to process a new mesh (this run, for example, will take approximately 20 times longer to process than the first).

Note that there will be an additional delay

 - c. With regards to hole management, the settings are left unchanged – there is no need to increase the *Triangle Size* limit as, with more triangles generated at closer intervals, this threshold is unlikely to be exceeded so meshing should occur across the entire surface other than the open 'entrance'.
3. **Next**. On the **Vertices and Triangles** screen, the *Smooth Wireframe* option should be selected.
4. Select *Normal Smoothing* and set a smoothing intensity of '2'. This will perform 2 'passes' of smoothing over the resulting raw mesh whilst honouring the above settings. Up to 10 smooth operations are possible.
5. Click **Finish** to generate the new higher-resolution wireframe.
6. Display the wireframe in the VR window. The image below shows a comparison of both runs of the wizard – note the resolution difference and hole status:



13 VISUALIZING WIREFRAMES

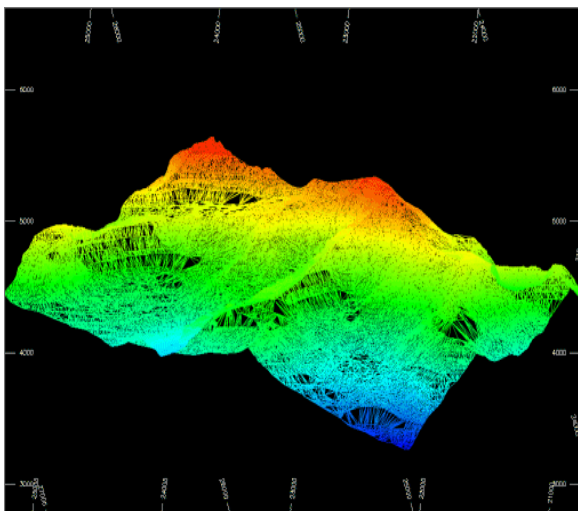
Wireframes can be viewed in a variety of ways, according to the viewing window in use at the time. The **Design** window (shown left, above) supports the use of multiple overlays to allow a single wireframe to be shown in more than one way.

Studio 3's **Visualizer** window (shown centre, above) can also be used to view wireframe (or any geometric) data using a different set of tools aimed at simple 3D rendering of your project.

You can also load your wireframe surfaces into the **VR** window (shown right, above) to show them in fully colored/textured/shaded virtual reality scenes. The advantage of the VR window is that it facilitates highly-accurate alignment of textures and wireframes using the inventive **Image Registration** feature – aligning images to wireframe landmarks is quick and easy with this facility.

All of these 'viewports' are linked – data from one view can be submitted to all other views instantly, allowing you to visualize your data as a CAD drawing, a simple rendering and/or a fully featured virtual environment – all based on the same underlying data.

Viewing Wireframes in the Design window



The **Design** window supports multiple views of the same object. Each view is known as an 'overlay' and the management of these overlays, including their creation and subsequent editing, is performed using the **Format Display** dialog, accessible by selecting **Format | Display**.

Each time any object is loaded into the **Design** window (providing it contains geometric information), a default overlay is created. This overlay can be edited if required, and you can create as many additional overlays as required per object. This process is covered in detail in your Studio 3 Introductory tutorial, but as a reminder of the general procedure:

1. If not already loaded, load your wireframe data into memory (so that it appears in the **Loaded Data** control bar).
2. Select **Format | Display**. You will see the **Format Display** dialog, open at the **Overlays** tab.
3. A list of all overlays, for all loaded objects, appears on the left, and a series of fields, categorized by sub-tabs (which will be shown or hidden depending on the type of data that is associated with the selected overlay).
4. Select the overlay you wish to edit, or Add a new one.
5. Set the relevant properties.
6. Click **Apply** or **OK** to view your changes in the **Design** window.

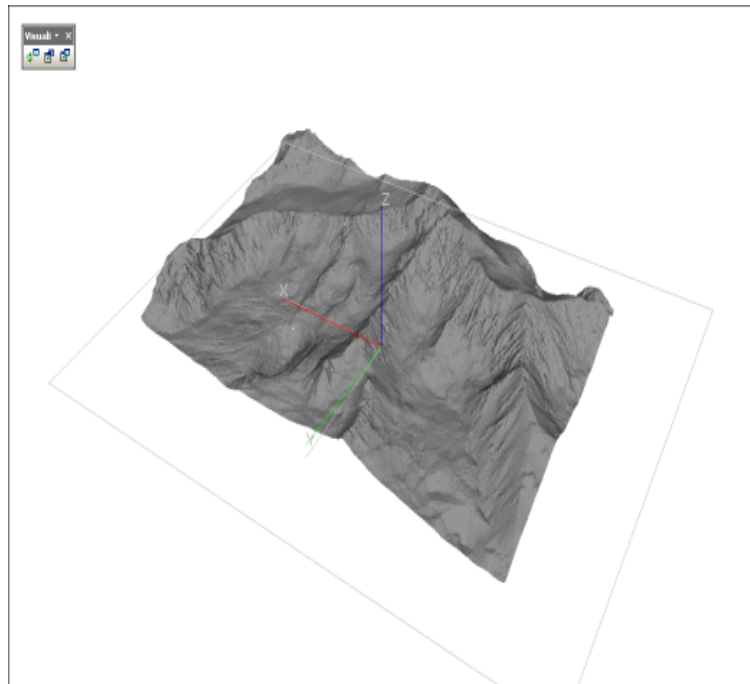
Consult your Studio 3 online Help for more information on all of the available overlay display settings. To access this topic, you can press <F1> whilst the *Overlays* tab is in view.

Viewing Wireframes in the Visualizer Window

The **Visualizer** window provides a quick and easy rendering solution for anyone wishing to show a 3D rendered view of their current data in memory.

By default, you will need to manually update the **Visualizer** view (although this can be set to an automatic option – it is advisable to leave the setting as it is with large data sets otherwise system performance can be adversely affected).

The **Visualizer** toolbar, found in the **Design** and **Visualizer** windows, allows you to update the view and objects in the **Visualizer** window with those currently shown in the **Design** window:



Studio 3 toolbars can be hidden or shown, docked or floated. In this respect, you may not be able to see the Visualizer toolbar on your Design window interface. If this is the case, select **View | Customization | Toolbars | Visualizer** to enable it.

Using the same toolbar, you can also perform a synchronisation the other way round – you can copy the current view direction from the **Visualizer** window and apply it to the current **Design** window data.



Visualizer functionality in Studio 3 originates from a program called 'GVP'. This utility was used to view Datamine models outside of their host system. As such, some older documentation may still refer to **Visualizer** functionality as 'GVP', and this acronym has also been maintained by some longer-term users of Studio software. In this situation, when 'GVP' is mentioned, they are referring to the **Visualizer** window.

The first step in visualizing wireframes is normally to load the wireframe data into the Visualizer window. This is done with the **Update Visualizer Objects** icon (the first icon on the left in the toolbar image above). This will load all current data (wireframe and other data types) into the **Visualizer** window and then display it.

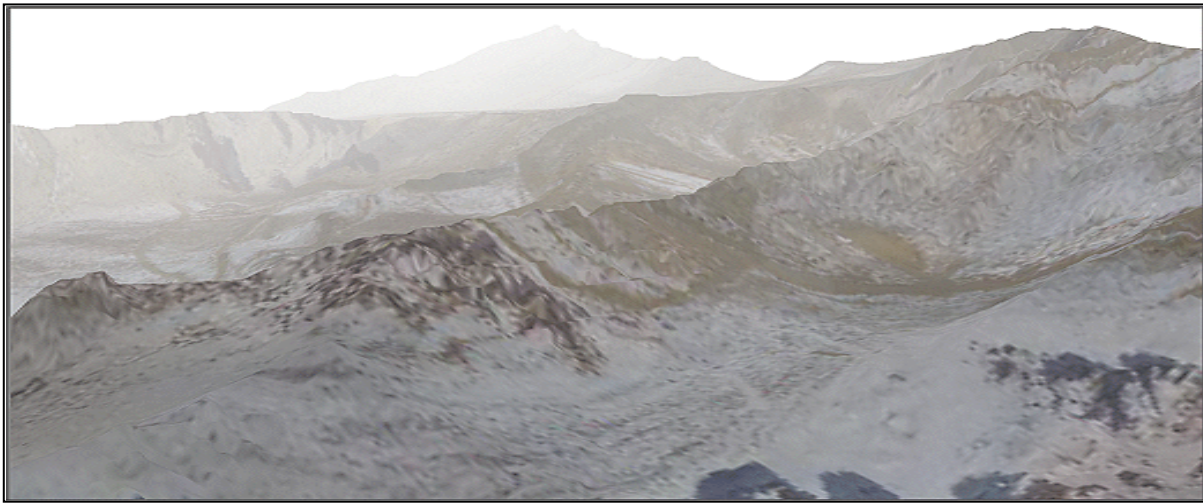
Once an object is displayed, you can rotate it dynamically with the mouse, and can zoom in and out by holding down <CTRL> and moving the mouse up and down.

Other things you can do in the Visualizer:

- Press the left and right arrow keys to automatically spin your visualized objects in the horizontal plane. Subsequent presses of the same key will speed up the rotation in the selected direction.
- Press the up and down arrow keys to spin the object automatically in the vertical plane. You can use this spin action in combination with the lateral spin created with left and right buttons to spin around bias angles.
- Set all visible wireframes to be hidden, visible or transparent. Please note that the **Visualizer** window does not support independent wireframe object display formatting. Settings are applied to all wireframes in memory.

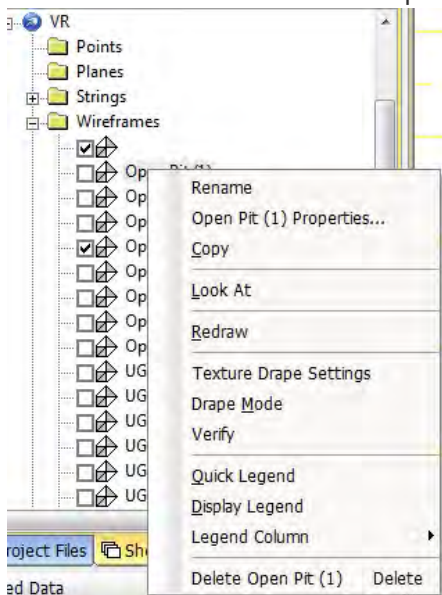
For more information on the **Visualizer** window, please consult your Studio 3 online Help.

Viewing Wireframes in the VR Window



The **VR** window is a fully-featured immersive virtual reality environment.

Data can also be removed from or added to your VR scene at any time after the **VR** window is opened. As with the Design window, you can create multiple instances of the same data object(s) and format them independently – very useful for representing data in multiple ways, e.g. showing an aerial imagery texture map on one topography instance and ordinance survey information on a separate instance.



VR window data is controlled using the **Sheets** control bar

(shown right). Full details on this control bar are given in your Studio 3 Help, but in brief, the **Sheets** control bar contains a **VR** sub-folder. In this folder, all VR data types are categorized by folder (surfaces, strings, points, block models etc.).

When a wireframe is viewed in the **VR** window, you can access it from this area of the **Sheets** control bar by viewing the contents of the *Surfaces* folder.

From here, you can right-click a particular surface object to reveal the **Surface Properties** dialog.

This multifunctional dialog allows you assign all manner of qualities to your surface, including the wrapping of a texture (geo-referenced or otherwise), how (and if) shading is performed amongst others. For textured objects, you can also activate – which enables the texture toolbar and allows you to configure exactly how a texture is placed onto the object's surface.

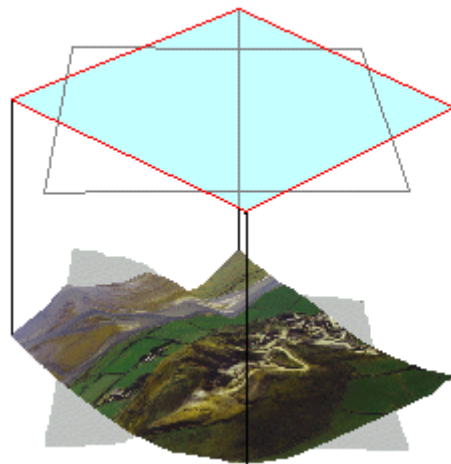
The **VR** window has powerful animation/simulation features, permitting complex fly-bys to be played back to a captive audience. As the VR window is not simply limited to the rendering of wireframes, you can show your topography, mine workings (existing and/or scheduled) alongside your orebody data and geological models to give a clear view of exactly what is being presented.

“Registering” an Image

Image Registration is a powerful method of applying textures to your wireframe in the expected position and orientation. You can apply textures to existing surface by aligning landmarks, and can also automatically create a flat wireframe simply by specifying an image.

Studio's **Image Registration** dialog is used for three main purposes:

- To position a texture on a wireframe using a series of anchor points, allowing for precise alignment to known visual references.
- To load a texture and create a planar reference wireframe on which to place it, for the purpose of reviewing the texture or aligning it with other loaded data. This can help to support a presentation by combining a mixture of an imported texture (e.g. a hand-drawn plot, aerial images, geological mapping images, seismic section data) alongside data resulting from a digitizing or modelling process.
- To automatically create a wireframe based on specified image data and display the corresponding image as a texture.



Alignment of texture is performed by matching a specified position on an image with the point to which it should be aligned with on the 3D data. You can do this by viewing your data from any direction, but for best results, you may wish to consider swapping to a plan view before you start. Note that all view direction commands are available to you even when the Image Registration dialog is displayed, allowing you to rotate your data and perform other functions throughout the process of texture alignment.

There is no enforced limit to the number of alignment points that can be set for a given image, although 3 will be the minimum required to align the 2D texture to an

appropriate alignment in 3D space and project it to the wireframe surface.

The **Image Registration** dialog can be reached in one of three ways when in the VR window:

- In the **Texture Drape Settings** dialog, select **Use Points**.
- Select **Data | Load | Image Wireframe** and select an image in the Open dialog.
- In the **Sheets** control bar, right-click the VR-Wireframes folder and select **Load | Image | Wireframe**.

Full **VR** window documentation, including dedicated tutorials, can be found as part of your Studio 3 online Help and Studio VR Tutorial.

The **Image Registration** dialog is comprised of three main components (see the image below for the location of each control):

1. the Image Registration toolbar (1)
2. the Image Preview pane (2)
3. the Alignment Table (3).



The above dialog will contain an **Apply** button if accessed via the **Texture Drape Settings** dialog - this is only appropriate if you are editing the texture that is already applied to a loaded wireframe. If the function is accessed via the **Sheets** control bar, VR -Wireframe folder context menu, an **OK** button will be displayed. The latter will generate a new, flat reference wireframe to 'host' the loaded image.

The Image Registration Toolbar

The following tools are available for you to accurately position a texture:



1. **New point:** note that at least 3 points are required to align a texture in 3D space. Selecting this option requires a further click in the preview window. A numbered reference point will be created and a new row added to the points table at the bottom of the dialog.
2. **Delete point:** enters deletion mode; any points selected (left-clicked) on the preview window will be removed as will the corresponding row in the table below.
3. **Move point:** enters point editing mode; left-click and drag any point that has previously been digitized in the preview pane. Note that this will not edit the coordinates in the table below that correspond to the edited point; you are changing the reference point in the image, not the point that it represents in 3D space - see the worked example to familiarize yourself with this concept.
4. **Pan view:** in a magnified view, select this to enter panning mode. Left-click and drag the texture preview to the required position.
5. **Zoom view:** select this option and subsequently left-click and drag to dynamically zoom the texture preview and get a better look at potential alignment points on the texture.
6. **Zoom area:** select and left-click to drag a rectangle representing the area of the texture preview you wish to magnify.
7. **Zoom all:** maximizes the texture preview to show the full extents of the image.
8. **Pan to Selected Point:** select this option to centre the view of the texture (at the current magnification) about the selected reference point. Select the reference point in the table first to highlight the relevant row then click the **Pan to Selected Point** command.

The Image Preview Pane

This pane displays a preview of the image to be draped and is also used to digitize the locations of image reference points.

The Alignment Table

Point	X	Y	Z	
1	5890.240	5742.832	119.172	X

The table at the bottom of the **Image Registration** dialog contains the 3D reference coordinates for each digitized image reference point shown in the Image Preview Pane. All values, other than the point number, are fully editable. The XYZ location of a particular reference point can be defined using one of the following methods:

- typing coordinate values into the table and the clicking **Apply**
- clicking on a row in the table and then in the VR window, right-clicking on the corresponding reference point e.g. a point on a wireframe, string or point

The table may also be used for fine-tuning of reference point coordinates in a particular direction.

You can also use the delete function on the far right of each row in the table to remove a reference point. This will also remove it from the **Image Preview** pane.

Guidelines for Best Results

For best results when texturing wireframes, you should consider the following general guidelines:

- Align your wireframe in a plan view if texturing an 'open' surface. This helps to show a better correlation between the image preview and the resulting textured mesh.
- Create a horizontal section through your data if texturing an open surface. To understand why, consider the following image:



- The red points in the image preview represent the points determined in the **Image Registration** dialog as 'anchor points'. The horizontal red line represents a horizontal section created prior to texture application (using the Section functions). The red arrow heads represent the points in space to which the texture should be aligned. The texture will then be dropped in a vertical direction onto the resulting wireframe without distortion or skewing.

If digitized directly onto the surface, in some cases the undulation of the surface will not invoke any significant distortion and the resulting image may be acceptable / accurate. However, where there is a big difference in, say, elevation values between the digitized 3D points and the points on the same image, it is possible that the projection may not accurately reflect the topography or volume that is being textured.

As such, it is recommended that a suitable section is created to act as a digitizing canvas when determining alignment points in 3D space.

Basic Texture Image Alignment - Procedure

Texture image alignment involves calculation of the best possible fit for the image given the specified alignment points.

Alignment of 2D textures and 3D images using reference points requires that the 'solution' for the best-fit texture can be found. In other words, it is possible to dictate an impossible alignment. You will be informed of this situation should it occur, allowing you to refine your alignment points or digitized 3D references accordingly.

The basic procedure for aligning a texture image on a wireframe surface is:

1. Load or import a wireframe file.
2. Apply a suitable texture image using the **Wireframe Properties** dialog. This can be a georeferenced image, if available.
3. Create a section in plan view on which to digitize your alignment points (although not strictly required, a flat section plane will provide best results as opposed to digitizing points onto a 3D surface).
4. In the **Sheets** control bar, VR-Wireframes folder, right-click the wireframe, select **Texture Drape Settings**.
5. Click **Use Points** to display the Image Registration dialog.
6. Digitize a minimum of 3 reference points on the preview image.
7. Select the relevant **Alignment Table** row (representing an alignment point) and then in the VR window, right-click on a corresponding point in your data e.g. the wireframe surface or surveyed points.
8. Repeat steps 6 and 7 for each point in the table.
9. Click **Apply**.
10. Use the scale, pan and edit commands to refine your texture positioning.

Worked Example 1 – Aligning a Hand-drawn Image to a Surface

This example uses the installed demonstration image data located on your local system, or downloaded from the CAE Mining website. Details regarding the location of data are provided below.

In the example, you are going to load an image representing a hand-drawn plot, and use the coordinates found on that document to create a correctly-aligned wireframe in 3D space. This could then be used as a basis for digitizing, for example, or comparing against other loaded data from a related data set.

This example assumes that your application is running and licensed on the host system and that you can see the Sheets | VR folder (Studio and Studio 5D Planner) or the Workspace in "Group by Type" mode (InTouch or EPS InTouch).

This example is common to all applications which support the Image Registration feature.

Sample Data

If running Studio or Studio 5D Planner applications, sample data is already installed on your local system at the following location (assuming default settings were accepted during the install process):

- **C:\Database\DMTutorials\Data\VBOP**

Detailed instructions on the location of specific data files will be provided in the example steps.

If running InTouch or EPS InTouch, sample data can be downloaded from the following location:

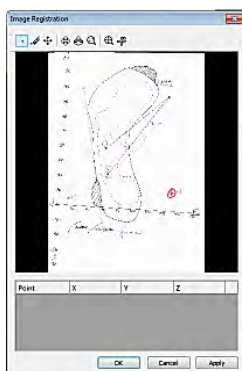
- **http://www.datamine.co.uk/InTouch_Example_Data/InTouch_Example_Data.zip**

Download the archive file and unpack it to the following location (creating folders as required):

- **C:\Database\InTouch_Test_Data**

The locations for Studio, Studio 5D Planner, InTouch and EPS InTouch will be referred to below as your "Sample Data Folder".

1. With your application running and a VR data display in view, open your Sample Data Folder.
2. Right-click the wireframe top-level folder in the **Sheets** or **Workspace** view and select **Load Image Wireframe...**
3. Navigate to your Sample Data Folder and open the *Pics* folder.
4. Double-click the *IMG_PLOT.jpg* file.
5. The **Image Registration** dialog is shown with a preview of the loaded image:

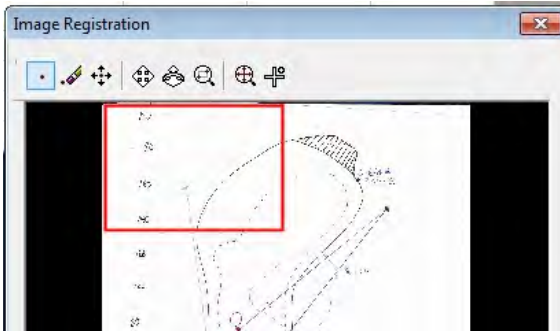


6. Next, you need to define 3 points on the image that you wish to align with the 3D wireframe surface.

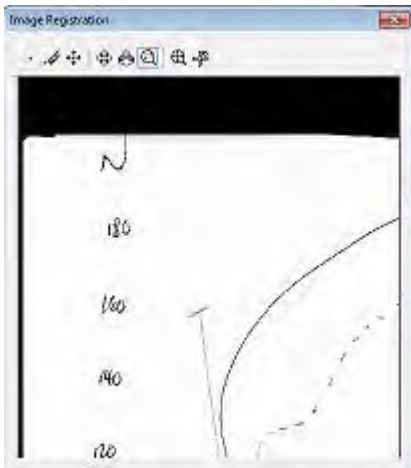
To make it easier to select the first point, select the **Zoom Area** button at the top of the dialog:



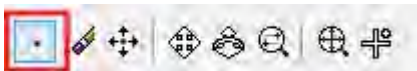
7. Left-click and drag a rectangle represented by the area shown below:



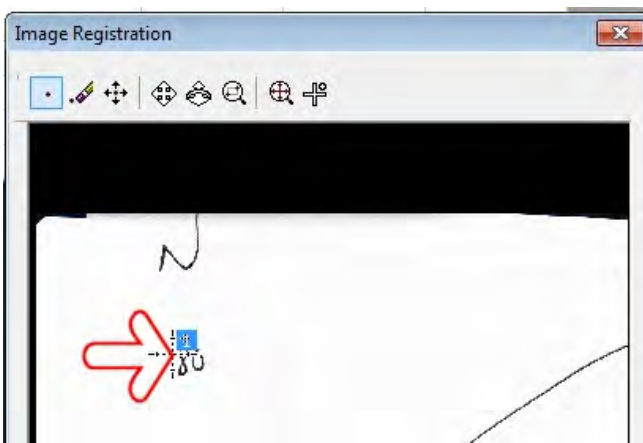
8. The texture preview will zoom to the selected area:



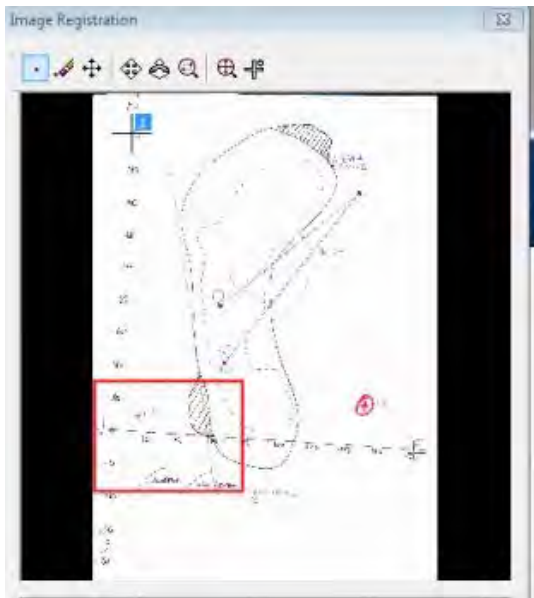
9. Select the **Add Point** button at the top of the dialog:



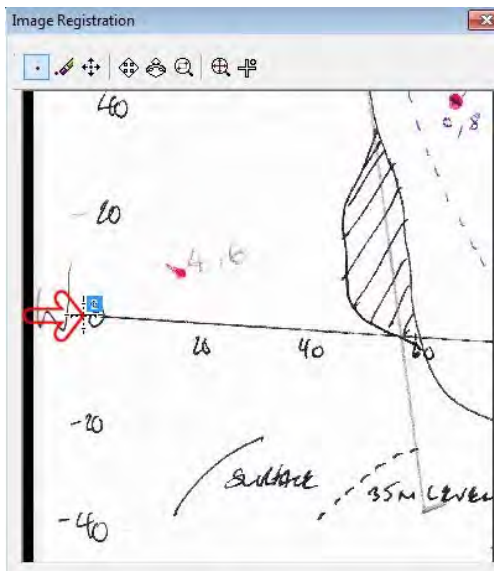
10. Left-click to add an alignment point directly above the "1" on the "180" description, i.e.:



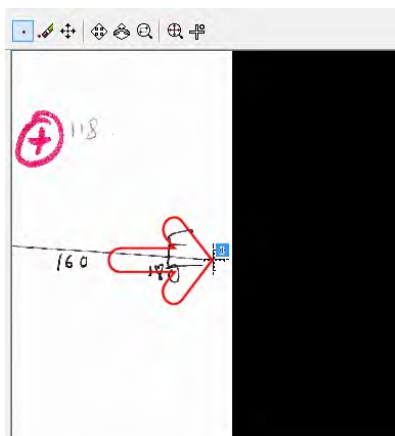
The red arrow is shown for indication purposes only.



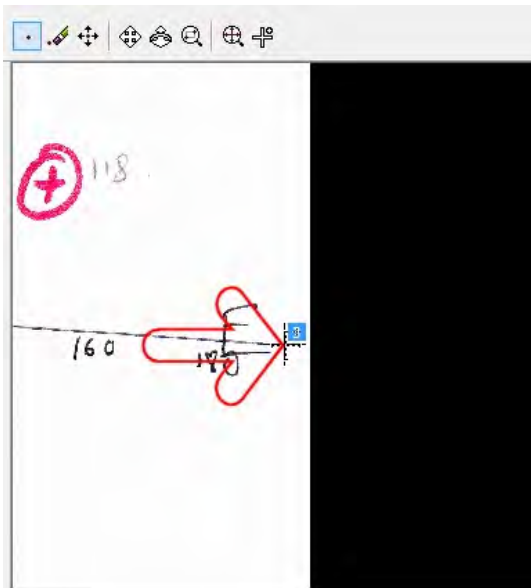
11. The next point will be added at the intersection of the graph axes. Zoom the preview to show the area shown below:



12. Select **Add Point** mode and click on the very beginning of the horizontal axis to position the second point, e.g.:



- Finally, maximize the view and use the zoom commands to position a third point at the end of the horizontal access (this represents the "190" value position, even though it isn't shown in text, e.g.:



- The table below now contains 3 rows (one for each digitized point). The values for X, Y and Z are all currently zero. This is because none of the digitized points have been 'assigned' to a point in 3D space yet. As the coordinates are known (they are on the chart), you will need to enter them into the table below, manually.

For the purpose of this example, you can assume that the elevation that is relevant to the plot is 250 meters. In other words; all points have a Z value of 250.

Configure the table at the bottom of the screen so it appears as shown:

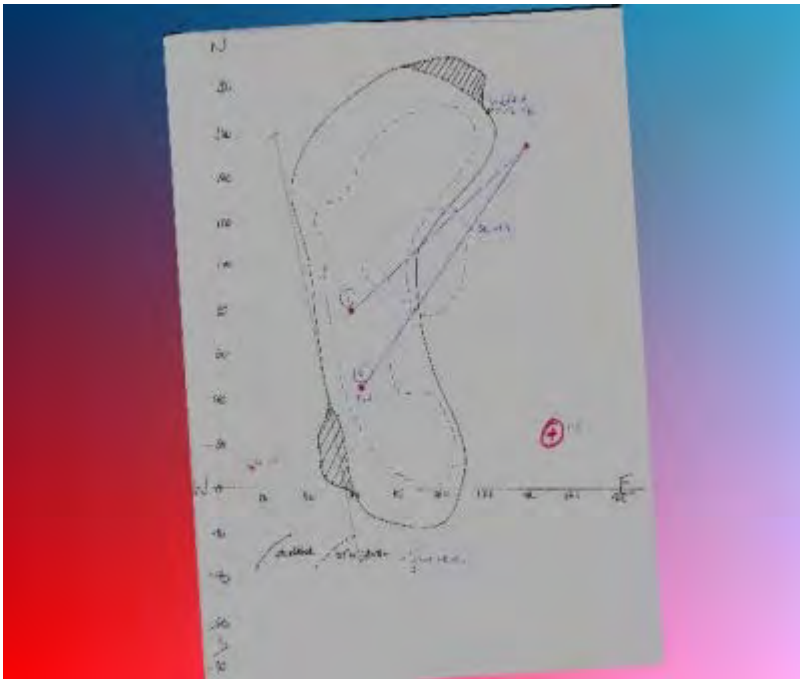
Point	X	Y	Z	
1	0.000	180.000	250.000	✕
2	0.000	0.000	250.000	✕
3	190.000	0.000	250.000	✕

- Click **OK** and your image will be applied to an automatically-generated wireframe. If you can't see anything in your data display window, right-click the [IMG_PLOT.jpg] entry that has now appeared in your Wireframes folder and select **Look At**.

- Finally, click the **Plan View** icon on the **View** toolbar:



17. This will orient your wireframe to the view window. Note how the chart axes, although scanned incorrectly, now appear in the correct orientation:



Worked Example 2 – Aligning an Image with 3D Topography

In this example, an existing 3D topography data is loaded and applied using the Image Registration function to align the image with known reference points. To ensure the points are applied to the same elevation, a section will be created above the topography for alignment purposes.

1. With your application running and a VR data display in view, open your Sample Data Folder using Windows Explorer.
2. Open the *Datamine* folder.
3. Left-click and drag the file *_vb_itsurfacettr.dm* into the data display window.

If the open pit wireframe is not visible in plan view at this point:

- a. Go to your Sheets | VR control panel (Studio, Studio 5D Planner) or your Workspace in 'Group by Type' mode (EPS InTouch).
 - b. Right-click the *_vb_ITPhoto-Texture.jpg* entry and select Look At.
 - c. In the View toolbar, select Plan View.
4. The aim is to display the open pit design in plan view, maximized to the screen, as shown:



5. In the *Wireframes* folder, right-click the *_vb_itsurfacetr/_vb_itsurfacept (wireframe)* item and select ***_vb_itsurfacetr/_vb_itsurfacept (wireframe)Properties***.
6. Select the ellipsis button on the right of the *Texture* field.
7. Locate your Sample Data folder.
8. In this folder, navigate to the *Pics* sub-folder and double-click the file *_vb_ITPhoto_Texture_rotated.jpg*.
9. Click **OK** and the texture is applied to the wireframe - note the incorrect alignment:



10. Now you can align the image. Right-click the loaded wireframe in the **Sheets** control bar or **Workspace** and select **Texture Drape Settings**.
11. In the **Texture Drape Settings** dialog, click **Use Points**.
12. The **Image Registration** dialog is displayed, showing a preview of the loaded image:



13. You will now define 3 points on the image that will be aligned with 3 landmark positions on the 3D wireframe surface.

14. Select the **Zoom Area** icon:



15. Left-click to drag a rectangle represented by the area shown below:



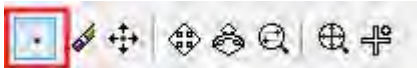
16. The view should now be similar to the following:



It does not have to be completely accurate, provided that most of the supply road is visible.

You can use the mouse wheel at any time to zoom the image preview in and out as required.

17. Select the **Add Point** button at the top of the dialog:



18. Left-click the junction of the main supply road and pit access lane, as shown below. Get as close to the specified point as possible - a reference point crosshair and "1" indicator will be displayed:



Note that a new row has been added to the table below - this will be explained in more detail below.

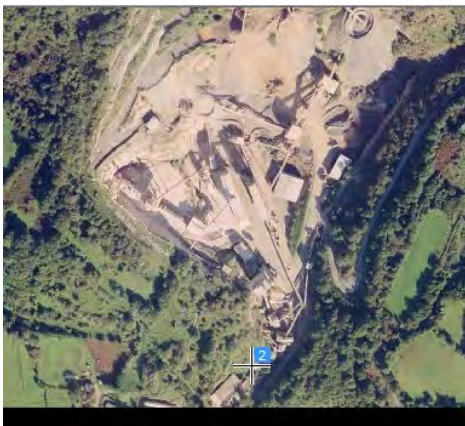
19. Use the **Zoom Fit** icon to maximize the texture to the screen:



20. Next, magnify the area detailed below, using the **Zoom Area** function as previously demonstrated:



21. Use the **Add Point** function to left-click at the point shown below:



22. Use the **Zoom Fit** icon to maximize the texture to the screen:



23. Zoom to show the full image again, and zoom into the final area:



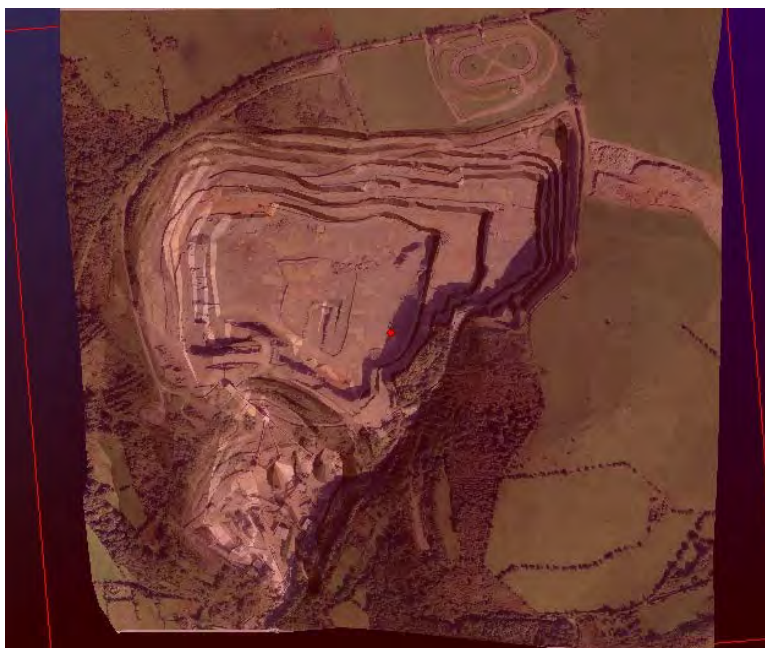
24. Add a third and final point at the position specified below:



25. The editable table at the bottom of the screen currently shows zero values for X, Y and Z for each point. This is the result of reference points being assigned to the image but not currently matched to a point in 3D space.
26. The coordinates for the digitized points are known. The table shown below should be edited to show the figures provided:

Point	X	Y	Z
1	5855.857	5550.030	157.371 ✕
2	5952.493	4645.979	157.371 ✕
3	6516.163	5583.011	157.371 ✕

27. Click **OK** to close the **Image Registration** dialog.
28. Click **OK** to close the **Texture Drape Settings** dialog.
29. The textured wireframe should now be visible in plan view, with a correctly aligned texture:





8585 Cote-de-Liesse

Saint-Laurent, Quebec

H4T 1G8

Canada

Tel: +1 514 341 2000 ext2404

www.cae.com/mining