# ET Surface

ET Surface is a set of tools for ArcGIS that enable the users to create surfaces and perform surface analysis without the need of 3D or Spatial Analyst extensions. The only requirement is an ArcGIS license (ArcView, ArcEditor or ArcInfo).

ET TerrainViewer - a stand alone application included in ET Surface allows 3D visualization of surfaces (PolygonZs, TINs and Rasters), features stored in shapefiles. It also allows draping aerial photography over a surface.

Versions 4.0 and above include ToolBox implementation of most of the functions. This allows running the functions from within Arc Toolbox environment, including them in models or using them in Python or VB scripts.

In version 5.0 most of the raster functions were completely rewritten with optimization for speed and memory use.

**Main Features:**

- **Interpolate Surface**
  - ESRI TIN and PolygonZ TIN from points, polylines or polygons.
  - Modify ESRI TIN and PolygonZ TIN by adding data from points, polylines or polygons.
  - Break lines (Hard and Soft) are supported for ESRI TIN.
  - Contours to Raster
  - IDW
  - Density
- **TIN Surface Analysis** - Slope, Aspect, Visibility, Volume, Cut/Fill
- **Raster Surface Analysis** - Slope, Aspect, Hillshade, Viewshed, Volume, Cut/Fill, Interpolate Contours
- **Raster Distance Analysis** - Euclidean Distance, Direction and Allocation, Weighted Voronoi (Thiessen) allocation, Cost Distance and Allocation
- **Raster Processing** - Clip, Erase, Smooth, Clean Boundaries, Resample
- **Raster Calculator** - enables the user to perform complex mathematical calculation on rasters.
- **Zonal Statistics**
- **Profile Extractor**
  - Create profile of multiple surfaces (ESRI TIN, Raster, PolygonZ TIN)
  - Draw profiles for:
    - User drawn cross-section line
    - Selected polyline or polygon graphic
    - Selected polyline or polygon feature
  - Interactive Profile Window
    - Zoom
    - Pan
    - Identify
    - Label data points
  - Animate profile for cross-sections moving along route
  - Draw profile directly on the data frame
  - Draw profile on the layout preserving the scale of the data frame
  - Export profile to a text file, an image, point or polyline feature class
- **Line of Sight (LOS) tool**
  - Draw on View and in Profile Window
  - Set offset for the Observer and Target
  - Apply Earth curvature corrections
  - Apply light refraction corrections
  - Apply radio wave corrections
  - Animate rotation of the Observer

- Digitize 3D features or graphics with elevation extracted from ESRI TIN, Raster, PolygonZ TIN

# ET Surface Installation

Installation instructions

Note that you have to be logged  as an Administrator on the machine you are installing ET Surface

**Important note: If you have ET Surface installed and plan to uninstall ArcGIS you MUST first uninstall ET Surface**

- Close ArcMap
- If you have a previous version of ET Surface installed, uninstall it first.
- Make sure that you have the sub-version of ET Surface appropriate for your ArcGIS version
- Unzip ETSurfaceXX.zip - two files will be extracted from the archive:
  - setup.exe
  - ETSurfaceXX_YY_Setup.msi
- Run setup.exe - a simple installation wizard will guide you through the process.
- A new program group with 3 items will be created
  - ET Surface User Guide
  - Readme
  - Terrain Viewer

If you start ArcMap the ET Surface toolbar should be already loaded.

- For pre ArcGIS 10 -Go to View ==> Toolbars and select the toolbar.
- For ArcGIS 10 - Go to Customize ==>Toolbars and select the toolbar.

**Note:**

ET Surface runs in DEMO mode until registered.

- The Demo mode has the following limitations
  - Build TIN function will use the first 500 points of the input feature class only.
  - Profiling, Line of Sight, and Digitize Z features functions will use the first 50 data points only
  - The functions that use surface datasets will work on surfaces with:
    - ESRI TIN - less than 1000 triangles
    - TIN PolygonZ - less than 1000 triangles
    - The raster functions do not have restriction on size, but the resulting rasters (if with more than 10,000 cells) will have a stamp with NoData cells.
  - The functions for which the main data source are feature classes, will not work on feature classes with more than 100 features.
- The registered version does not have the above restrictions.
- See How to Register ET Surfacefor registration information

# ET Surface - How to load in Arc Toolbox

The ET Surface tools can be loaded as any standard geoprocessing tool. The Setup program of ET Surface will register the DLL with the system and the tools in the DLL will be registered with the correct component categories. The Setup will also copy in the installation folder of the software the ET Surface Toolbox

Load all ET Surface tools:

1. Right-click the Arc Toolbox folder inside the Arc Toolbox window and click Add Toolbox.
2. Navigate to the folder where ET Surface is installed and select ETSurface.tbx file
3. Click Open.

Load a tool into your own toolbox or toolset:

1. Right-click the toolbox or toolset where you want to add system tools, point to Add, and click Tool.
2. In the dialog find and expand ET Surface Geoprocessing toolbox and toolsets in it. Check the tools you would like to add to your toolbox or toolset. If you check the toolbox, all tools within the toolbox will be added. If you check a toolset within the toolbox, all tools within the toolset will be added.
3. Click OK.

Since the usage of the ET Surface tools is exactly the same as the standard tools provided with ArcGIS, we highly recommend you to have a look at "Geoprocessing in ArcGIS" in the desktop help

# ET Surface - How To Register

A. Single Use license

The registration process involves three steps

1. Visit ET Surface page on ShareIt.com and purchase the software. You will receive a reference number for your order.
2. On the ET Surface Toolbar or ET Surface Main Dialog go to Help ==> Request License Key button. Fill the small form - all the fields are required.
    - User Name
    - Company
    - ShareIt reference number ( see Step 1)

    After filling the form there are two options to chose from:

    - Create Key Request File will write all the information to a file (*.et3). Send this file to register@ian-ko.com and in 24 hours you will receive the Key File that will unlock the full version
    - Send Key Request via e-mail. This option will open you default e-mail program with all necessary information. You just have to click the SEND button

    Important note:

    Do not change anything in the request file or the body of the generated message. It will cause the registration process to fail.

3. When you receive the Key File , save the attachment (*.ets file) to you hard disk. Click on Register button (Help ==> Register). In the form, click on Load Key File button. Select the received file. The program will be registered.

Important notes:

- You need to be logged as an administrator on the PC
- Do not change anything in the Key File. It will cause the registration process to fail.

B. Concurrent license

ET LicenseManager should be installed on a PC on your network

1. Contact your system administrator and get the following information:
    - The Name or IP address of the PC where the ET LicenseManager is installed
    - The TCP port on which the ET License Manager communicates
2. Go to Help ==> Connect To License Server
3. In the dialog fill
    - License Server - fill the network name or the IP Address of the license server
    - TCP Port - fill the port number
4. Click on the Test License Server button
5. If a connection to the license server is established, click OK to save the settings. You are ready to work.
6. If the test fails - contact your system administrator.

# Types of Surfaces

The functions and tools of ET Surface can be used in on 2 types of surfaces - Rasters and TINs

**Rasters** represent the surface using a matrix of uniformly sized square areas. Each square, which may be referred to as a cell or pixel stores a single value that represents what the surface represent. The most common case of the value of a cell is elevation of a terrain, but it can be many other things (temperature, rainfall, etc.). Each cell of a georeferenced raster represents a physical cell in the space. The size of the cells define the resolution of the raster - smaller cells - better resolution - larger size of the raster.

The **TIN** model represents a surface as a set of contiguous, non-overlapping triangles. Within each triangle the surface is represented by a plane. The triangles are made from a set of points called mass points.

See TIN Notes for more information about TINs

ET Surface can generate only a TIN surface, but most of the functionality can be used with both Raster and TIN surfaces.

Raster formats that can be used:

- ESRI Grid
- ERDAS Imagine images (.img)
- Tagged Image File Format (TIFF) image (.tif)
- Personal Geodatabase raster
- File Geodatabase raster

TIN Formats that can be used:

- ESRI TIN - proprietary format. Surfaces created by 3D Analyst or Arc/Info TIN module or ET Surface version 4.0.
- PolygonZ TIN - A TIN Surface stored in a PolygonZ dataset (Shapefile, Personal or File Geodatabase). Surfaces created by ET GeoWizards and ET Surface.

A list of the functions of ET Surface and surfaces on which they can be used

# ET Surface and Projections

This topic will discuss some issues concerning the projections of the surface datasets and the analysis performed on these datasets.

Coordinate System Types ( from ArcGIS desktop help)

- A Geographic Coordinate System (GCS) uses a three-dimensional spherical surface to define locations on the earth. A GCS is often incorrectly called a datum, but a datum is only one part of a GCS. A GCS includes an angular unit of measure, a prime meridian, and a datum (based on a spheroid).
  A point is referenced by its longitude and latitude values. Longitude and latitude are angles measured from the earth's center to a point on the earth's surface. The angles often are measured in degrees (or in grads).
- A projected coordinate system is defined on a flat, two-dimensional surface. Unlike a geographic coordinate system, a projected coordinate system has constant lengths, angles, and areas across the two dimensions. A projected coordinate system is always based on a geographic coordinate system that is based on a sphere or spheroid.
  In a projected coordinate system, locations are identified by x, y coordinates on a grid, with the origin at the center of the grid.

Coordinate systems and 3D analysis.

The Geographic Coordinate System provides a way to store common coordinates for locations anywhere in the world.. Due to this fact it is used in many areas (Location based services, navigation, etc.).

If however we want to measure distances and areas on data in a GCS we are facing an obvious problem - the units of measure of a GCS are actually angles - a distance of 2.5 Decimal Degrees does not mean much, an area of 1.5 "Square Decimal Degrees" (if such term existed) means even less.

One can argue that using GCS we can calculate distances and areas on the Spheroid and the results will be in meaningful distance/area units (meters, feet, etc..) and more accurate than the results derived from projected data. This might be true, but only on large scale (continental) data where the projected data will be more distorted by the single projection used to represent it in Cartesian coordinates.

If we take into consideration the geographic extent of the surface data that is normally used for 3D analysis, we can conclude that an appropriately selected projection for the location of the data will give us better results.

Based on the discussion above, the functions of ET Surface work

- On surface data in any Projected Coordinate System.
- Surfaces (Raster, ESRI TIN, PolygonZ TIN) that are in a Geographic Coordinate System, need to be projected to a suitable projection first.
- In order to get correct results for Slope, Volume and 3D Area the Z units should be the same as the units of the spatial reference of the data.

# ET Surface - Raster functions performance and limitations

The performance of the raster functions of ET Surface depends on the specification of the computer (Processor & RAM). There are however several other factors that influence the performance depending on the function. All function check at the beginning whether whether the memory available will be enough to complete the function. Some general considerations:

- Do not use smaller cell size than required for the task. The cell size defines how much memory will be needed and influences significantly on the performance of the functions. It is a good idea to calculate the approximate number of cells of the output raster - based on the extent of the output - Number Cells = Width x Height / $CellSize^2$
- If you are going to execute a function that requires a lot of resources
  - Close the unnecessary applications to free some memory.
  - Restart ArcMap with a new project.
  - Load only the data necessary for the function
  - Execute the function.

The table below tries to indicate the factors that influence the performance and the limitations in number of cells for each function. Common factors influencing the performance and keys used in the table.:

- Distribution of the input points - normally the more even the distribution - the better the performance - DP
- Number of input points - NP
- Number of cells in the input raster - NCI
- Cell Size the smaller the cell size, the larger the output - CS
- Difference between the minimum and maximum weight for the cost functions - DW
- Contour Interval - CI
- Smoothing -SM
- Extent of the output - EO

All the limits below tested on standard Pentium PC with 2 Gigabytes of RAM

Indication for the limits in the table below - a raster that covers 100kilometers by 100kilometers with cell size of 20 meters will have 25,000,000 cells.

In ET Surface 5.0 most raster functions were rewritten with optimization for speed and memory use. The table shows the indicative limit for raster sizes in previous versions and ET Surface 5.0

| **Function** | Performance depends on | Indicative limit - number of cells | |
|---|---|---|---|
| | | Previous | ET Surface 5.0 |
| **Surface Interpolation** | | | |
| Contour To Raster | CS | 30,000,000 | 100,000,000 |
| IDW | DP, CS | 45,000,000 | 100,000,000 |
| **Raster Surface Analysis** | | | |
| Raster Slope | NCI | No Limit | No Limit |
| Raster Aspect | NCI | No Limit | No Limit |
| Raster Hillshade | NCI | No Limit | No Limit |
| Interpolate Contours from Raster | NCI, CI, SM | No Limit | No Limit |

| | | | |
|---|---|---|---|
| Viewshed | NP, NCI | 45,000,000 | 100,000,000 |
| Cut/Fill Analysis | NCI | No Limit | No Limit |
| Raster Volume | NCI | No Limit | No Limit |
| **Raster Distance Analysis** | | | |
| Euclidean Distance | DP, CS | 45,000,000 | No Limit |
| Euclidean Direction | DP, CS | 45,000,000 | No Limit |
| Euclidean (Voronoi) Allocation | DP, CS | 45,000,000 | No Limit |
| Weighted Voronoi Allocation | DP, CS | 45,000,000 | 100,000,000 |
| Cost Allocation (Cost from source) | DP, CS, DW | 45,000,000 | No Limit |
| Cost Distance (Cost from source) | DP, CS, DW | 45,000,000 | No Limit |
| Cost Allocation (Cost from raster) | DP, CS, DW | 30,000,000 | No Limit |
| Cost Distance (Cost from raster) | DP, CS, DW | 30,000,000 | No Limit |
| **Raster Processing** | | | |
| Clip Raster with Envelope | NCI | No Limit | No Limit |
| Clip Raster with Polygons | NCI | No Limit | No Limit |
| Erase Raster with Polygons | NCI | No Limit | No Limit |
| Smooth Raster | NCI | No Limit | No Limit |
| Clean Boundaries | NCI | No Limit | No Limit |
| Create Constant Raster | CS, EO | No Limit | No Limit |
| Create Random Raster | CS, EO | No Limit | No Limit |
| Change Raster Data Type | NCI | No Limit | No Limit |

# ET Surface Toolbar


ET Surface toolbar  is a container of all the functionality of the software

- ET Surface Menu has the global functions of the software grouped in several sub-menus.
- The Profiling tools  work on multiple surfaces. Only the surfaces visible in ArcMap table of contents (TOC) will be used by the profiling tools
- The Select Surface Layer combo box

  is used to set current surface layer for use with the Line Of Sight (LOS)  tool and the tools for digitizing  3D features.
- Manage graphics tool  will help you to select graphics by name or type, delete selected graphics or group them.
- The Help Menu can be used to access the User Guide, register the software, view the Log file

# ET Surface Hydrological Functions

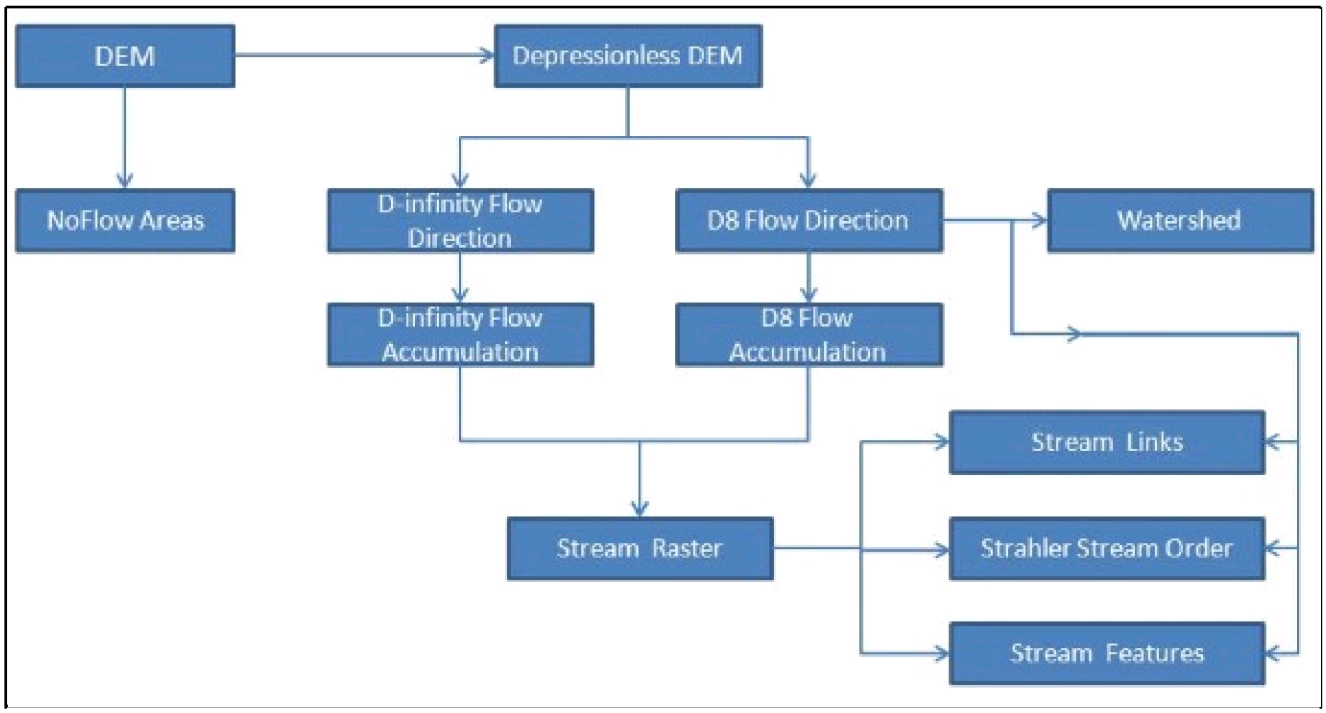
ET Surface 6.0 introduces a set of functions for hydrological analysis.

They are used to model the flow of water over the terrain represented by a raster Digital Elevation Model (DEM). The availability of Digital Elevation data has increased significantly over the last years. The advances in technology have allowed also for increased accuracy and resolution.

**The Hydrology functions in ET Surface include the following:**

- **Analyse the DEM Raster for areas for which the flow can not be determined (NoFlow areas) - sinks and flat areas**
- **Remove NoFlow areas and create "Depressionless" DEM.**
- **Model the flow of water over the DEM and create Flow Direction and Flow Accumulation rasters**
- **Delineate Stream Network from Flow Accumulation rasters, assign Strahler Stream Order and Stream Link IDs to stream sections.**
- **Create Stream polyline features and Node point features of the Stream network.**
- **Delineate Watersheds**

The starting point for any hydrological analysis is a Digital Elevation Model. Usually there are a number of steps which have to be followed for performing a specific task. The following flowchart shows the sequence of steps for using the provided functionality:



**DEM Pre-Processing**

The most common digital data of the Earth's surface is raster digital elevation models (DEMs). And because the flow of water is determined by the terrain it has become more common to use DEMs for the analysis of hydrological processes.

All the functions in the Hydrology group are based on a raster Digital Elevation Model.

In order to use a DEM for hydrologic analysis it is important that the flow of water can be defined for each cell of the DEM raster and then following the flow direction it is possible to reach the DEM edge.

Depressions (also known as pits or sinks) are areas in the DEM which do not allow the flow of water to an outlet at the edge of the DEM. They may be representative of the natural terrain, but could also be result from the technical procedures in the DEM production.

In order to use a DEM for hydrological modeling, it is important to eliminate depressions.

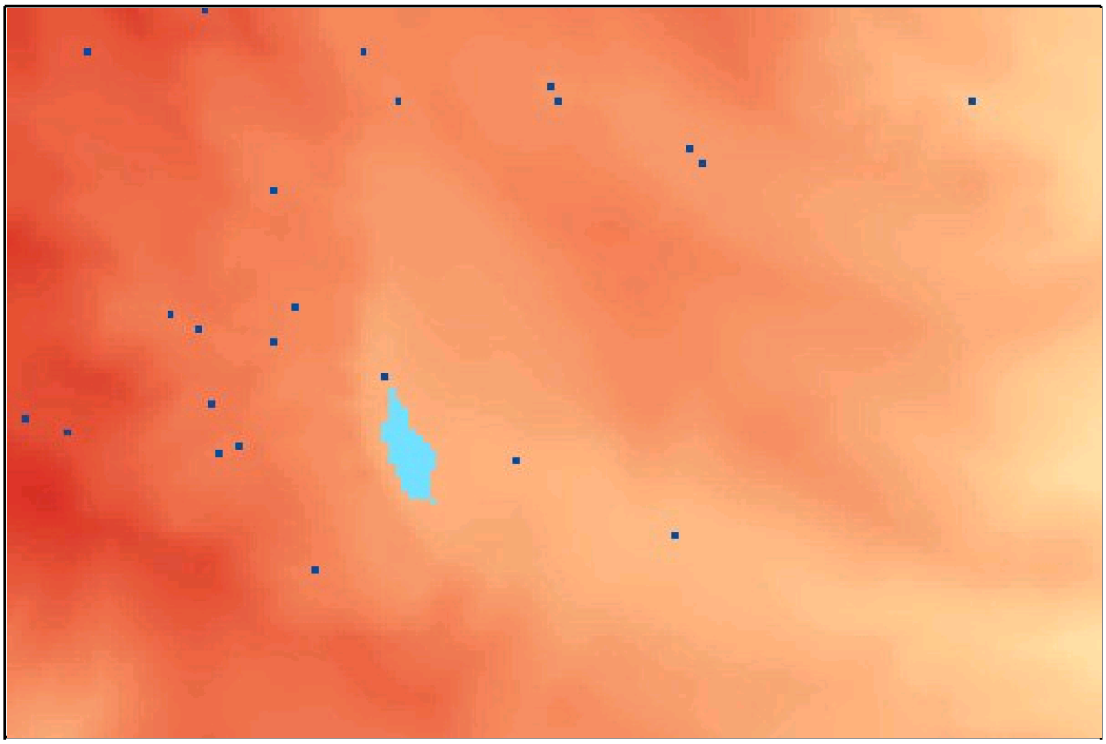
ET Surface includes a function for finding the problem areas (NoFlow Areas) and a function for removing them (Fill Depressions).

NoFlow Areas analyses the DEM by inspecting each cell and its 3x3 neighborhood and marks the NoFlow Areas.

An output value of 1 indicates that the cell belongs to a flat area - there is no lower neighboring cell and the lowest neighbor has the same elevation.

An output value of 2 indicates a sink - meaning that all neighboring cells are with higher elevation.

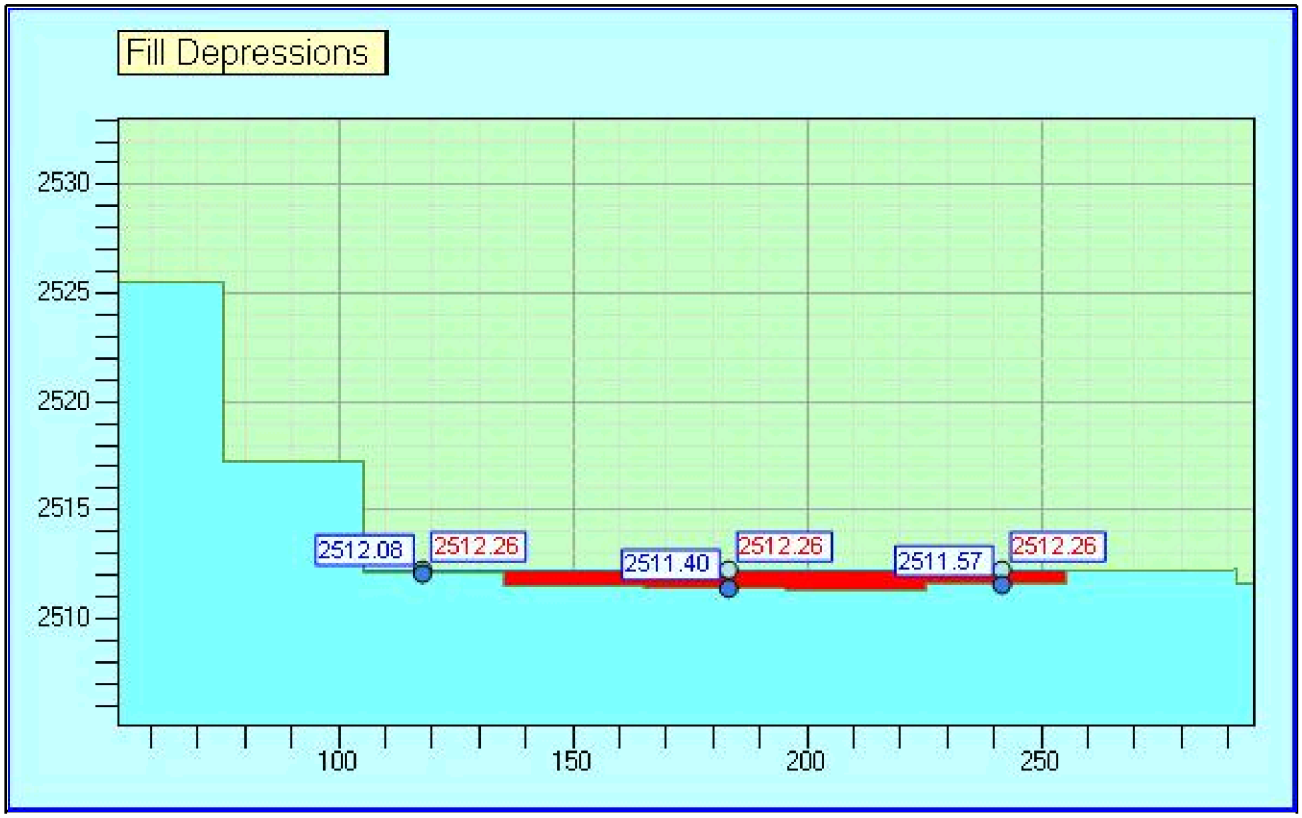
An example of the NoFlow Areas output - Flat areas with light blue and Sinks with dark blue.



Fill Depressions removes the NoFlow areas from the DEM using the algorithm proposed by Planchon and Darboux (2001). It removes sinks and flat areas by increasing the elevation of NoFlow cells in such a way that all inner cells of the DEM have a defined flow. The resulting DEM can be called "depressionless"

In flat areas first the cells with defined flow at the boundary are established and then the elevation of neighboring cells is increased with a minimal slope.

Filled depression view in the Profile Extractor - the red area shows cells where a depression was filled.


Fill Depressions
2530
2525
2520
2515
2510
2512.08
2512.26
2511.40
2512.26
2511.57
2512.26
100
150
200
250

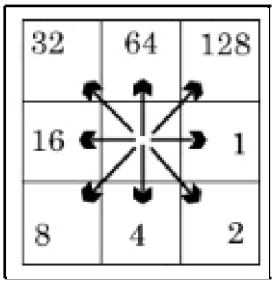
**Flow Direction and Accumulation**

There are a number of algorithms for determination of flow over a DEM. At present ET Surface has implementations for two of them:

- Deterministic 8 (D8) proposed by O'Callaghan and Mark (1984).
- Deterministic Infinity (D-infinity) proposed by Tarboton (1997).

**The D8 method**

The D8 method is the most widely used and has been implemented in many software packages. In this model the flow direction for each cell is determined according to the steepest descent to one of its 8 neighboring cells. The direction values are most often coded as 1,2,4,8,16,32,64 and 128 according to the figure below:




Thus if the steepest descent of a cell is to the left, its Flow direction will be coded as 1, if it is to upper right, it will be coded as 32. The steepest descent is calculated by dividing the elevation difference by the distance between the cell centers.

Once the Flow Direction is defined, the next step is to determine the Flow Accumulation. The value in the Flow Accumulation raster represents the number of upstream cells from which the water flowing downstream will pass through the current cell. The calculation is done from the Flow Direction raster starting from the top - the cells to which there is no contribution and passing their accumulated value downslope.

In ET Surface the Flow Accumulation values represent number of cells. The actual contributing area for the cells can be determined by multiplying the accumulation value by the area represented by a cell.

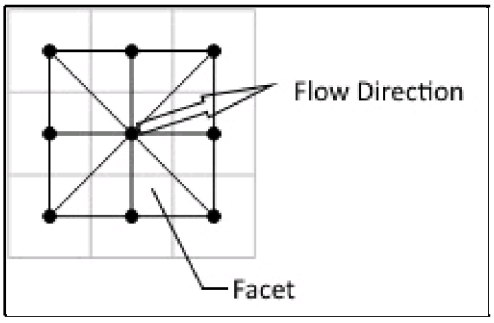
Thus for a raster with cell size 1 x 1 meter the values are directly square meters. For a cell size of 10 x 10 meters, the accumulated value must be multiplied by 100 to obtain the contributing area.

Cells with a high value in the Flow Accumulation raster indicate high concentration of water and can be used for identification of streams.

**The D-infinity method**

The D-infinity method was suggested by Tarboton (1997). In it Flow Direction is defined as the angle of the steepest descent determined by analysis of 8 triangular facets formed by the 3x3 cell neighborhood.

The triangular facets are formed by the centers of the center cell and its 8 neighbors. The flow direction for a particular facet is the direction of the steepest downward slope on the facet. The flow direction for the cell is the one with the highest magnitude of all the eight facets. The possible values are in decimal degrees from 0 to 360 starting from North in clockwise direction.

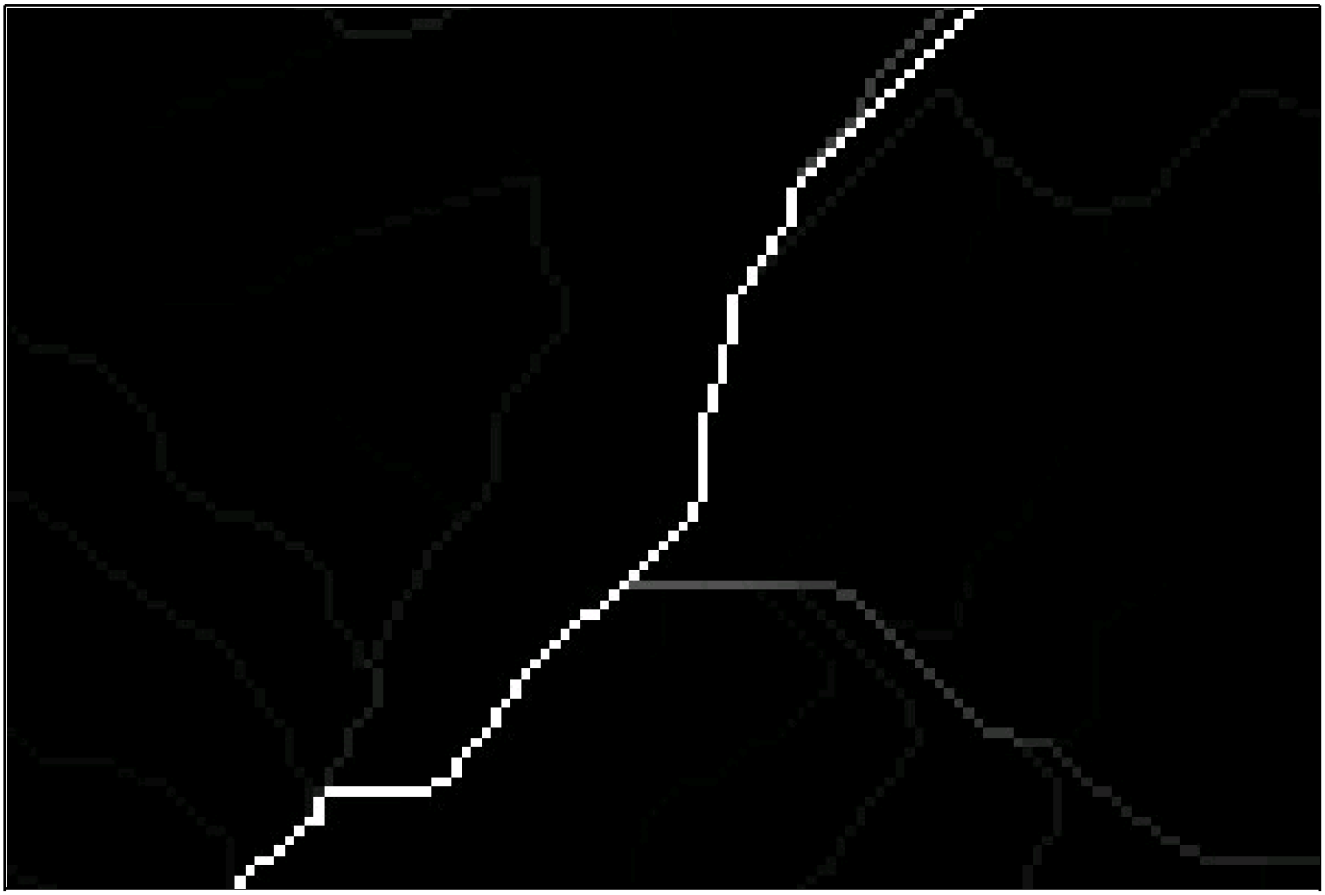



For example a value of 90 indicates flow to the East and a value of 225 - flow to South West.

The D-infinity method allows flow divergence - the flow from a cell will either go to one or two of the neighboring cells.  This is a better representation of water flow on divergent slopes.

Thus a flow accumulation value of 100 in a cell can be distributed between two neighboring cells transferring respectively 65 to the one and 35 to the other.

An example of Flow Accumulation D-infinity result (above) and Flow Accumulation D8 (below).
Note the dispersion of flow in the D-infinity result compared with the concentrated flow in D8.



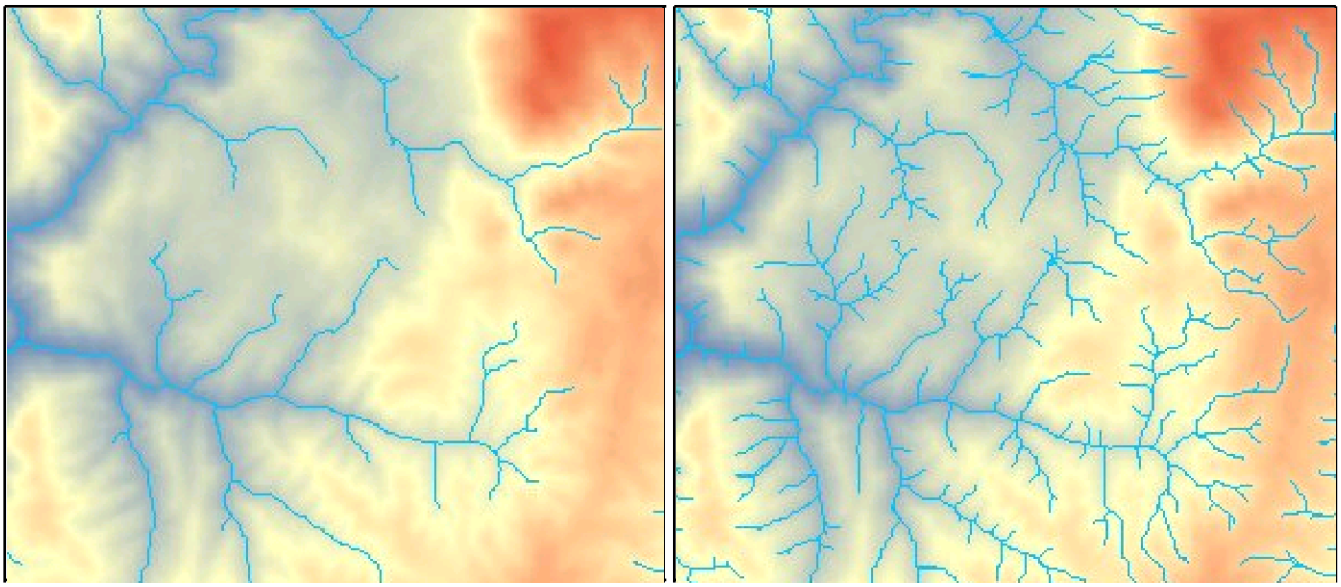
**Delineation of Stream network**

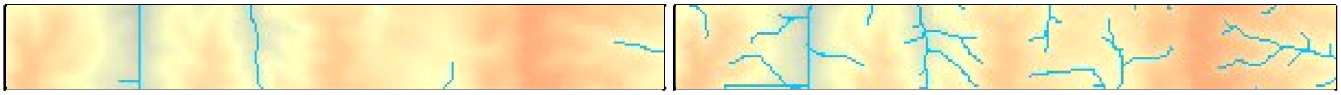
The Flow Accumulation raster is used for Stream delineation. As we saw previously, the Flow Accumulation raster values represent the number of upslope cells flowing through each cell. By applying a Threshold value to the Flow Accumulation raster we can determine the cells which form a Stream network. All cells with Flow Accumulation value above the threshold are considered to be part of a stream.

The threshold is called Stream Initiation Threshold (also Channel Initiation Threshold). It represents the minimum contributing area required to initiate and maintain a Stream. The determination of the threshold value will depend on several factors including the type of the terrain, climate, soils and the DEM resolution.

Reviewing existing data and maps for the area and experimentation with different values are helpful for the determination of the threshold value. Lower values will result in denser Stream Network.

The following images show the difference in Stream density at Threshold values of 1000 and 100





When using Flow Accumulation raster produced with the D8 method, the increase of values downslope is guaranteed, since each cell can only flow to one neighbor.

This is not the case with Flow Accumulation created by the D-infinity method. There the accumulated value for a cell can be distributed to two neighboring cells, thus causing reduction of the accumulation. This can lead to disconnected streams after the Stream Initiation Threshold is applied.

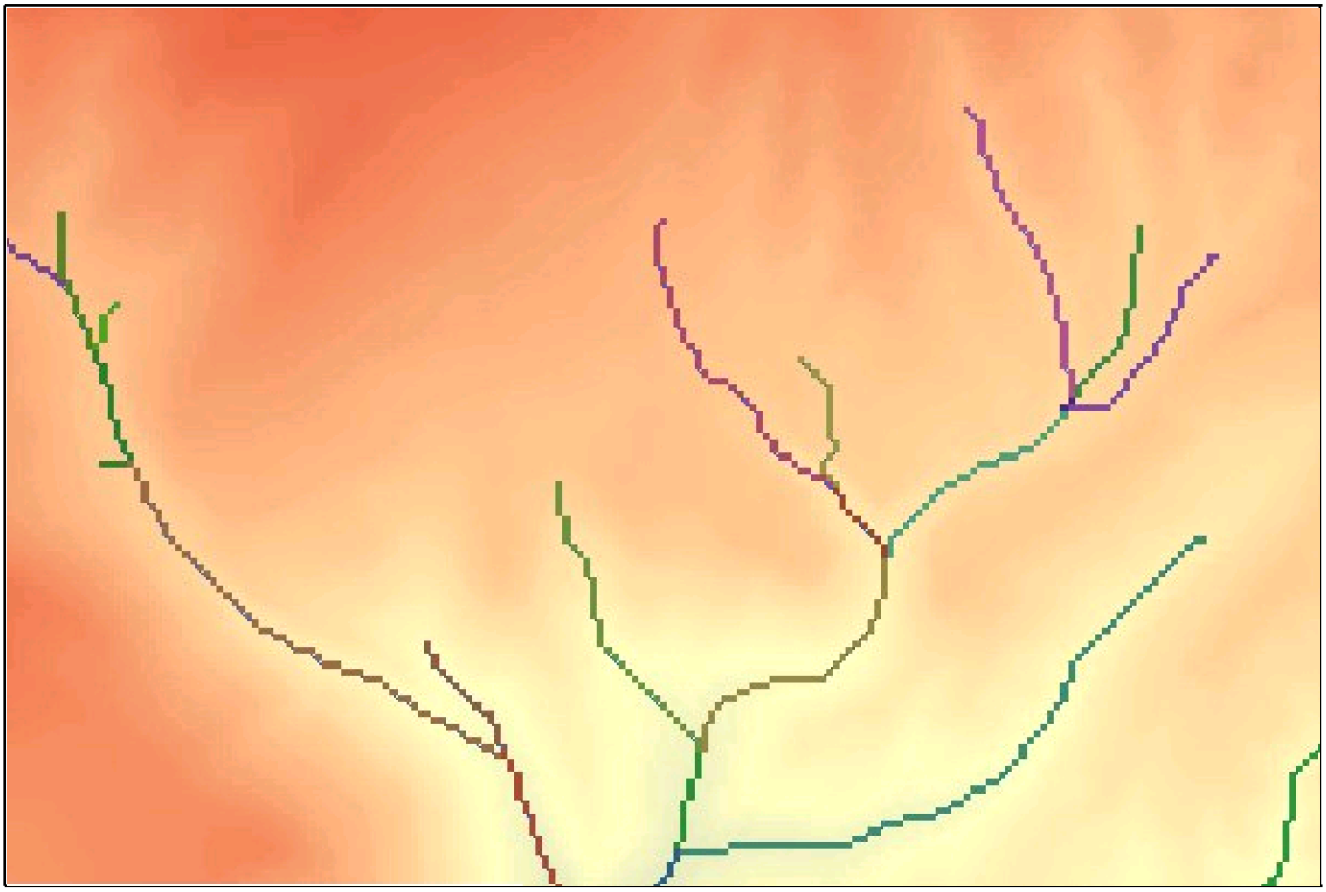
To circumvent this the D-infinity Flow Accumulation function in ET Surface provides the option to use the D8 accumulation method once the Threshold is reached. In this case the user has to specify the Threshold and provide a D8 Flow Direction raster as input.

**Stream functions**

The Stream Link function assigns unique values to sections of the Stream Network. It uses as input the Streams raster and the D8 Flow Direction raster.

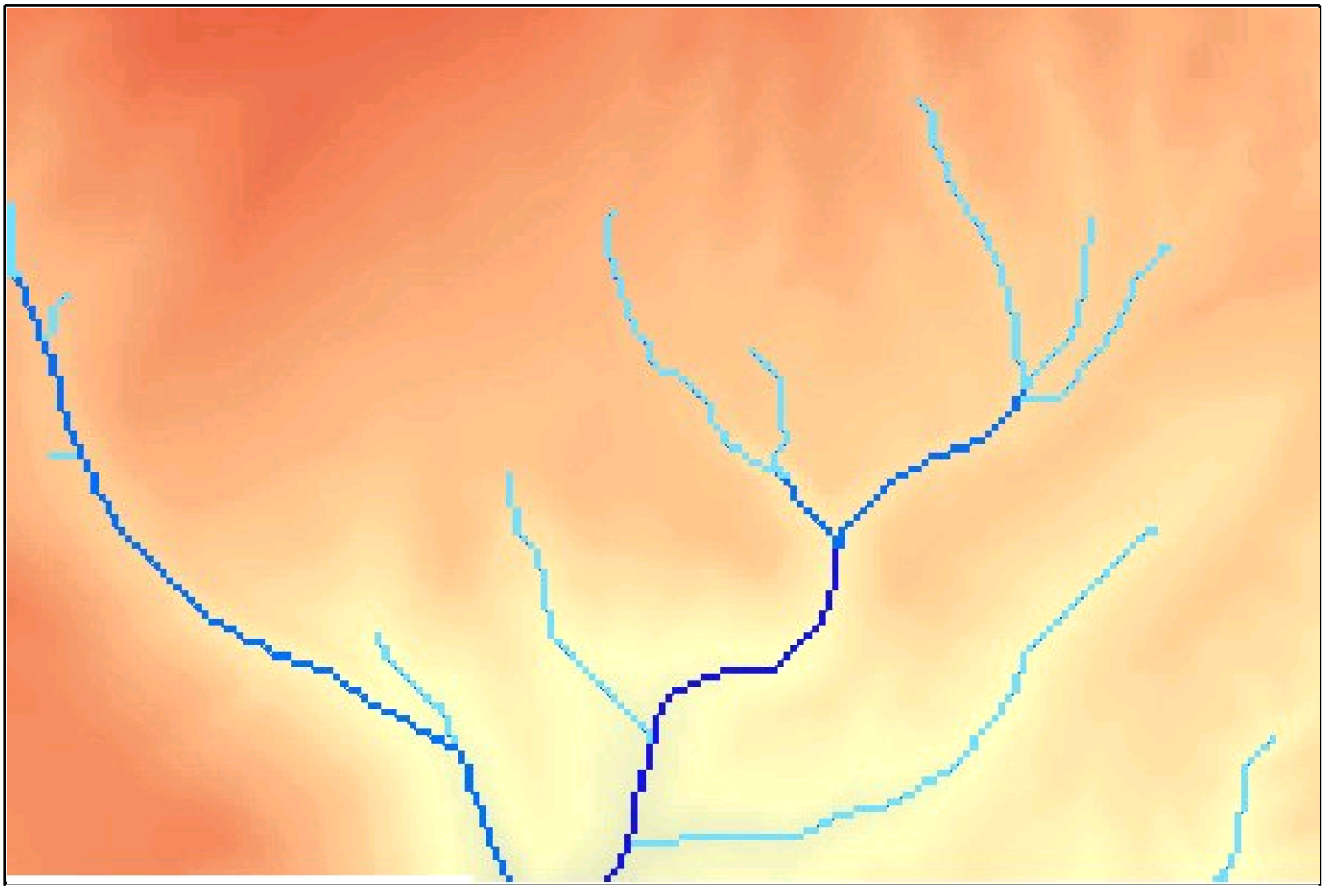
A Link is each section of Stream between two junctions, the stream head and a junction or a junction and the outlet.

An example of Stream Link output.



The Strahler Stream Order function assigns an order to each stream segment according to the system proposed by Strahler (1952). The order of the stream section starting at the stream head is assigned to 1. The order increases by 1 only when two sections of the same order intersect. For example if two sections of order 1 intersect, the following section will be assigned an order of 2. When two sections of different order intersect, the following section preserves the higher order.

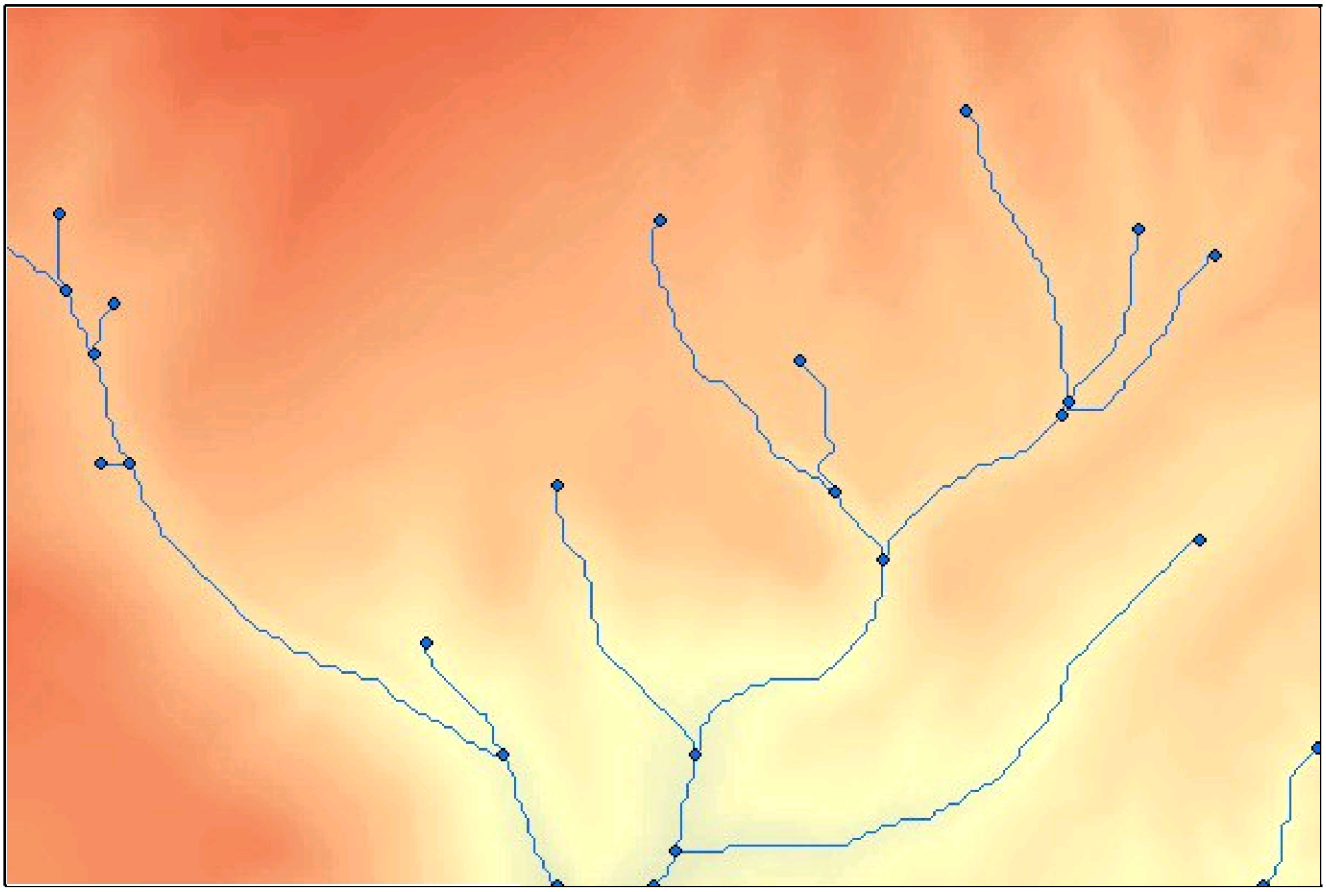
An example of Strahler Stream Order result.



The Stream Raster To Features function converts a Stream raster to a polyline feature class and optionally creates a point feature class with the Nodes of the stream network.  It requires the D8 Flow Direction raster as input to determine the correct direction of flow.

Each polyline in the output feature class represents a section of the stream and is assigned a unique ID.  The start and end nodes are recorded as attributes.  The direction of the polyline is always downstream.

An example of Stream Raster to Features output - Streams and Nodes.



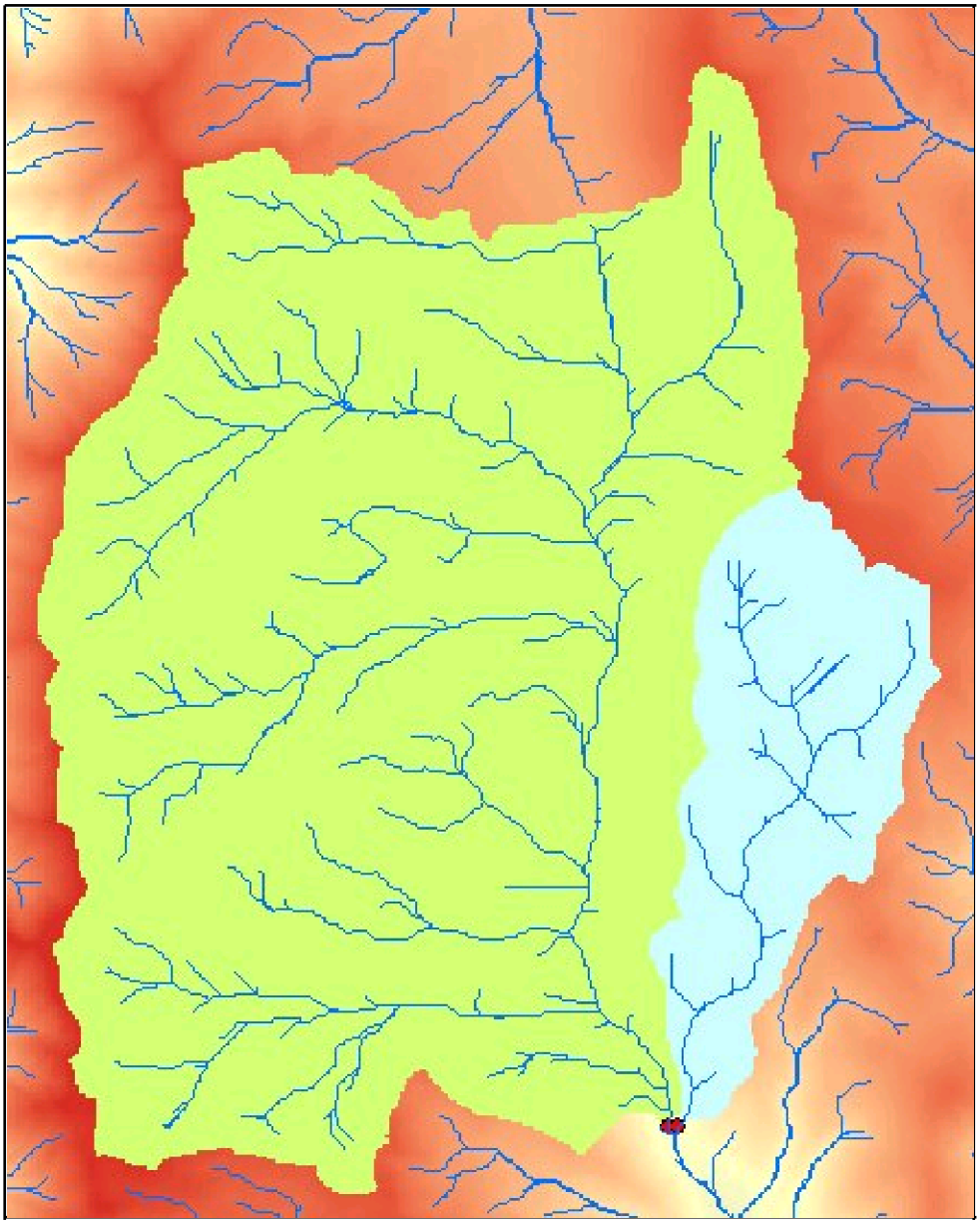
**Watershed**

The Watershed function delineates the area contributing flow to a set of Outlet points.  The Outlet is the lowest point in the Watershed.

Watersheds are delineated from the D8 Flow Direction raster. An optional Outlets feature class can be specified as input. In this case Watersheds are delineated for the Outlet points. An ID is assigned to each Watershed from a field in the Outlets feature class.

If no Outlet points are specified, all Outlets for the DEM are determined (the points at which flow leaves the DEM) and Watersheds are delineated for them.  In this case the Watersheds are assigned unique generated values starting from 1.

An example of Watershed output using Outlet points .



If Outlet points are specified it is recommended to ensure that they are within cells with sufficient flow to delineate a reasonable Watershed.  Otherwise it is possible only a few cells (or even just one) to be delineated.

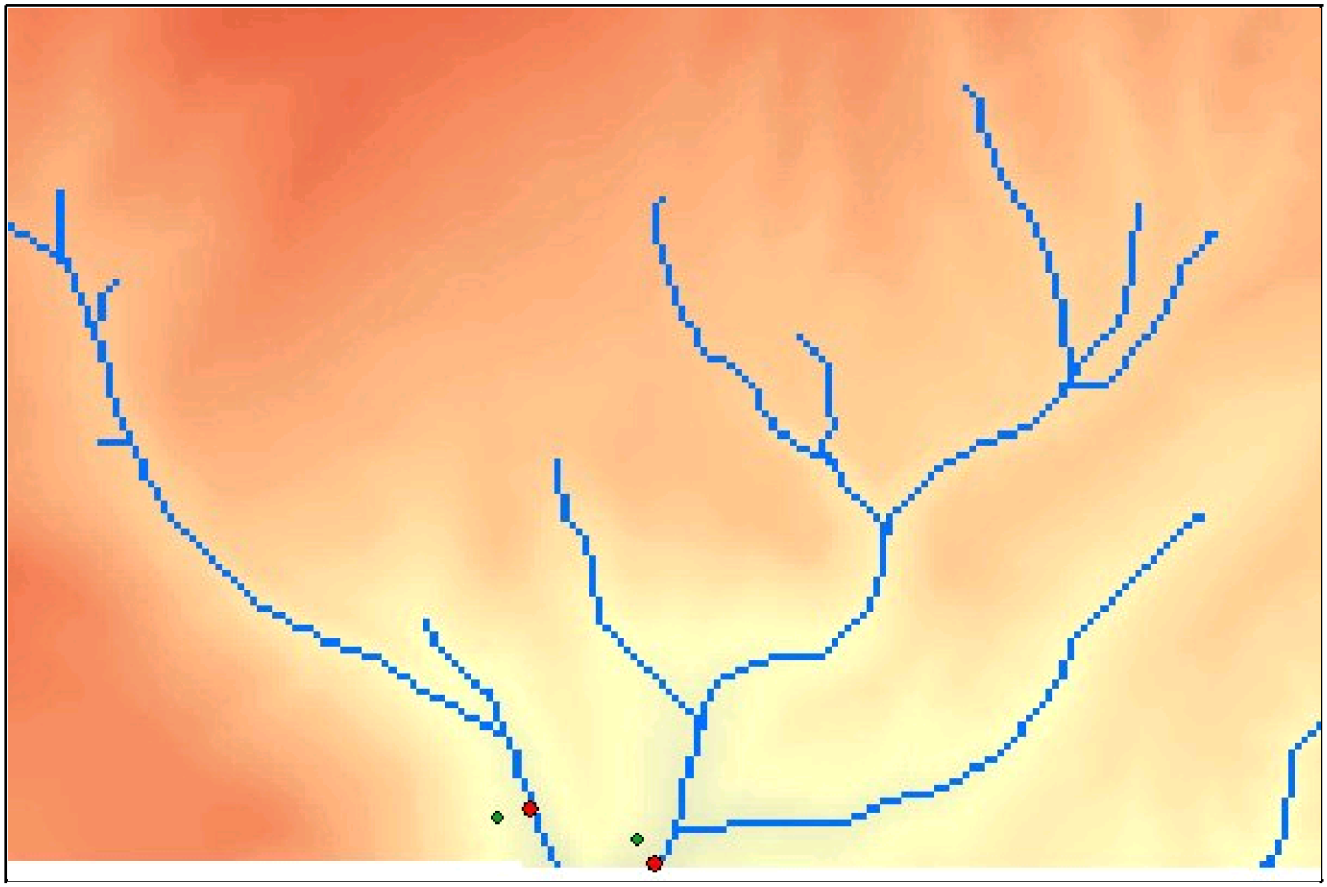
For this the Snap Pour Points function can be used.  It allow the Outlet points to be moved within a specified distance based on the values of a reference raster.

There are three Snap options:

- Snap to nearest Stream
- Snap to lowest Elevation
- Snap to highest Flow Accumulation

The relevant raster and maximum snap distance have to be provided as input parameters.

An example of Snap Pour Points using the Stream option - initial points in green and after snapping in red.



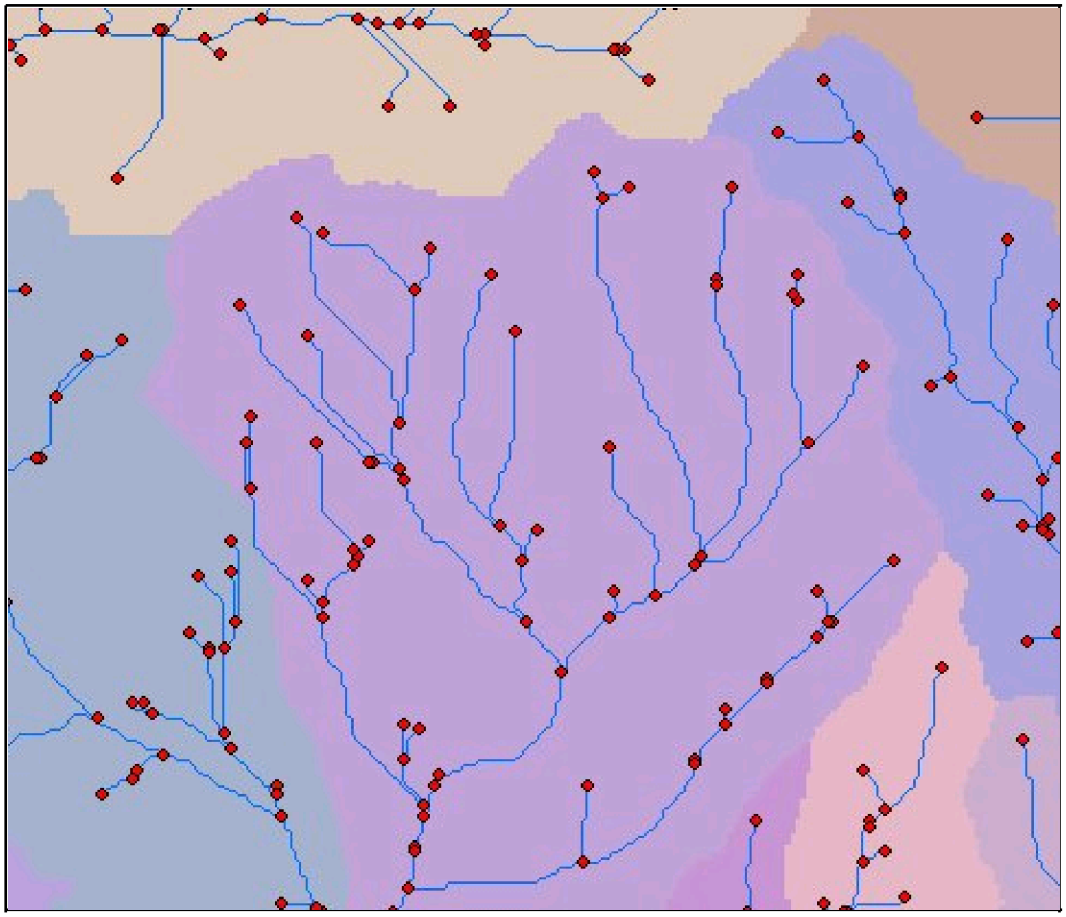
**Streams And Watershed**

The Streams And Watershed function allows the creation of Stream and Node features as well as a Watershed raster from a DEM, The function calculates as temporary datasets the required Flow Direction and Flow Accumulation rasters based on the selected flow model (D8 or D-infinity).

If the D-infinity flow method is selected, the flow accumulation below the specified Stream Initiation Threshold is determined by the D-infinity method and above the threshold the D8 method is used.

The Watershed Raster represents Watersheds for all Outlet points determined from the Flow Direction raster and Watershed IDs are automatically generated.

An example of Streams And Watershed output - Streams and Nodes Features and Watershed raster.



**References:**


O'Callaghan, J. F., and Mark, D. M., (1984). "The extraction of drainage networks from digital elevation data", Computer Vision, Graphics and Image Processing, Vol. 28, pp. 328-344.

Planchon, O. and F. Darboux, (2001), "A fast, simple and versatile algorithm to fill the depressions of digital elevation models", Catena, 46: 159-176.

Tarboton, D. G., (1997), "A New Method for the Determination of Flow Directions and Contributing Areas in Grid Digital Elevation Models", Water Resources Research, 33(2): 309-319

Tarboton, D. G., R. L. Bras and I. Rodriguez-Iturbe, (1991), "On the Extraction of Channel Networks from Digital Elevation Data", Hydrologic Processes, 5(1): 81-100.

# ETS Hydrology Toolbar

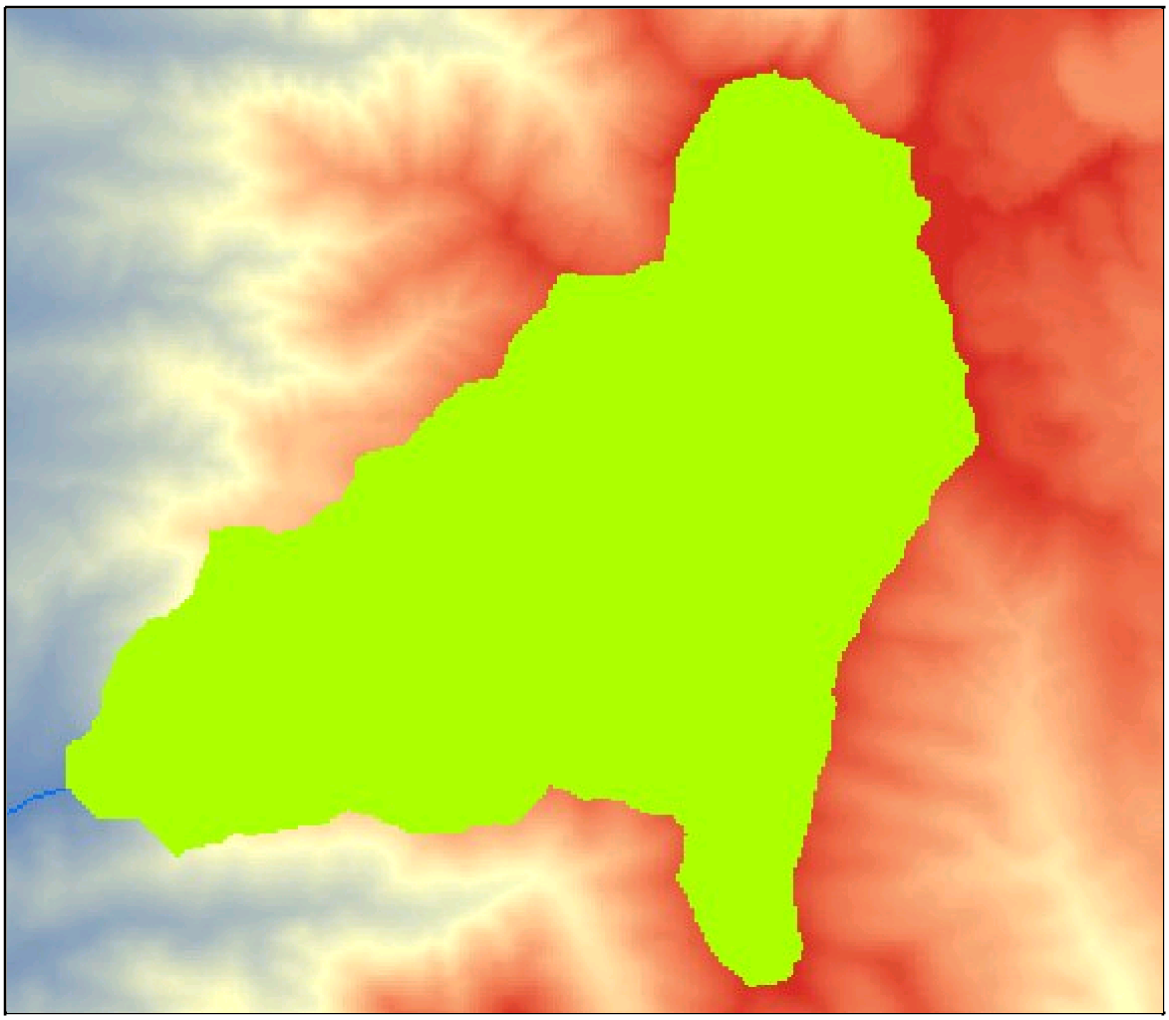

ETS Hydrology  toolbar  is introduced in ET Surface version 6.0 with the Hydrology group of functions

- At present it contains two tools and will be expanded with the development of the Hydrology group of functions.
- The selection Combo box allows the selection of a Raster which will be used by the tools for performing the tool function.  At present both available functions require a raster representing Flow Direction D8.
-  The Watershed tool is used to delineate a Watershed for a selected point on the surface. The delineation is based on a D8 Flow Direction raster which has to be selected in the Combo box.

  For better visualization the DEM raster should be displayed on top of the Flow Direction raster.

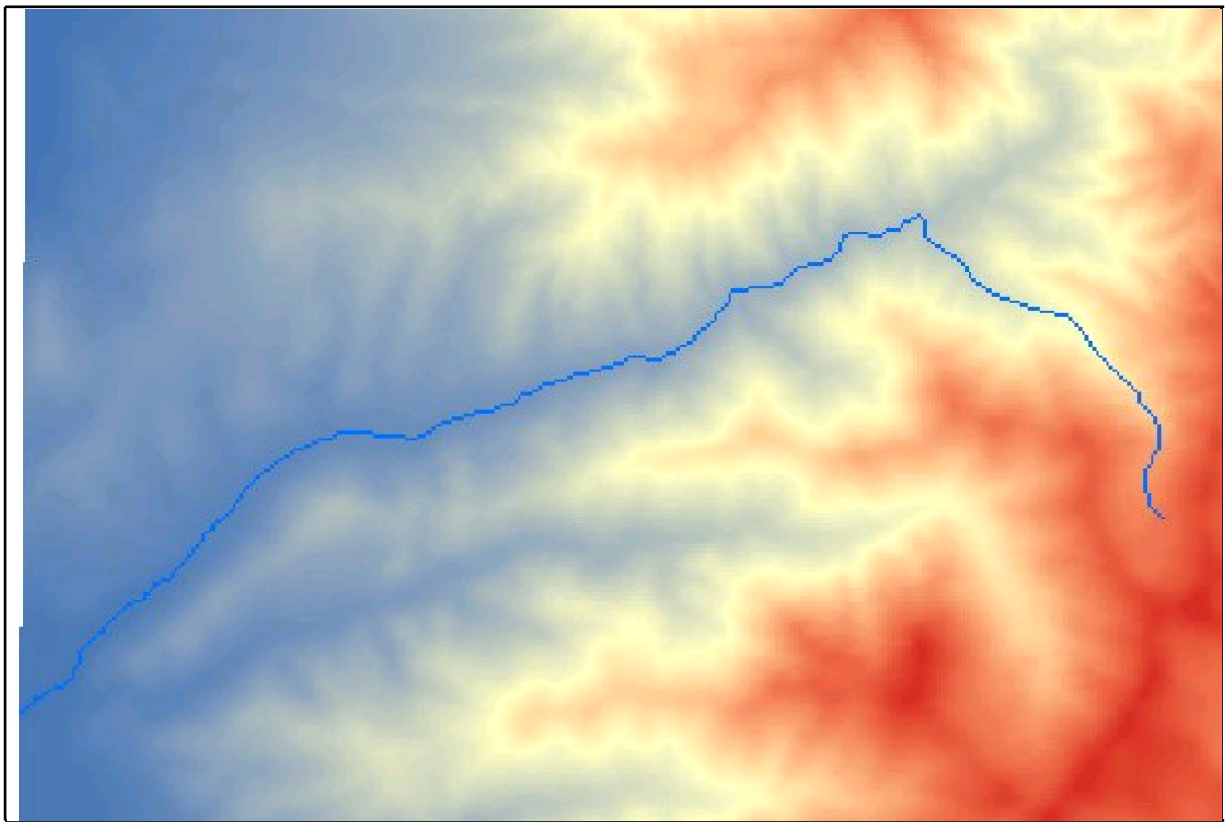
  The delineation is usually done for a point on a stream and if there is no Stream layer available sometimes it is difficult to select the correct location.  In such a case it is useful to use the Flow trace tool to create a Flow path and then use the Watershed tool based on the flow path.

  An example of the Watershed tool output.



-  The Flow Trace tool is used to create a Flow path for the water from the specified point over the terrain to the Outlet. A Flow Direction D8 raster has to be selected in the Combo box in order to use the tool.

  An example of the Flow Trace tool output.



- Both tools create temporary rasters in the ET Surface Temp folder.

## ET Surface Main Dialog

The Main Dialog of ET Surface gives access to all the functions of the software except the Profile Extractor and the Terrain Viewer.

Selecting the tab for specific category of functions will display a list of all functions in this category.  Next to each function there is an icon indicating the status of the function

- indicates that the function is available with no limitations
- indicates that the software is not registered and if you run the function the limitations of the unregistered software apply
- appears when a licensed (or free) function is selected. Clicking on the icon will execute the function.
- appears when a non-licensed function is selected. Clicking on the icon will execute the function with the applicable to the unregistered software limitations.

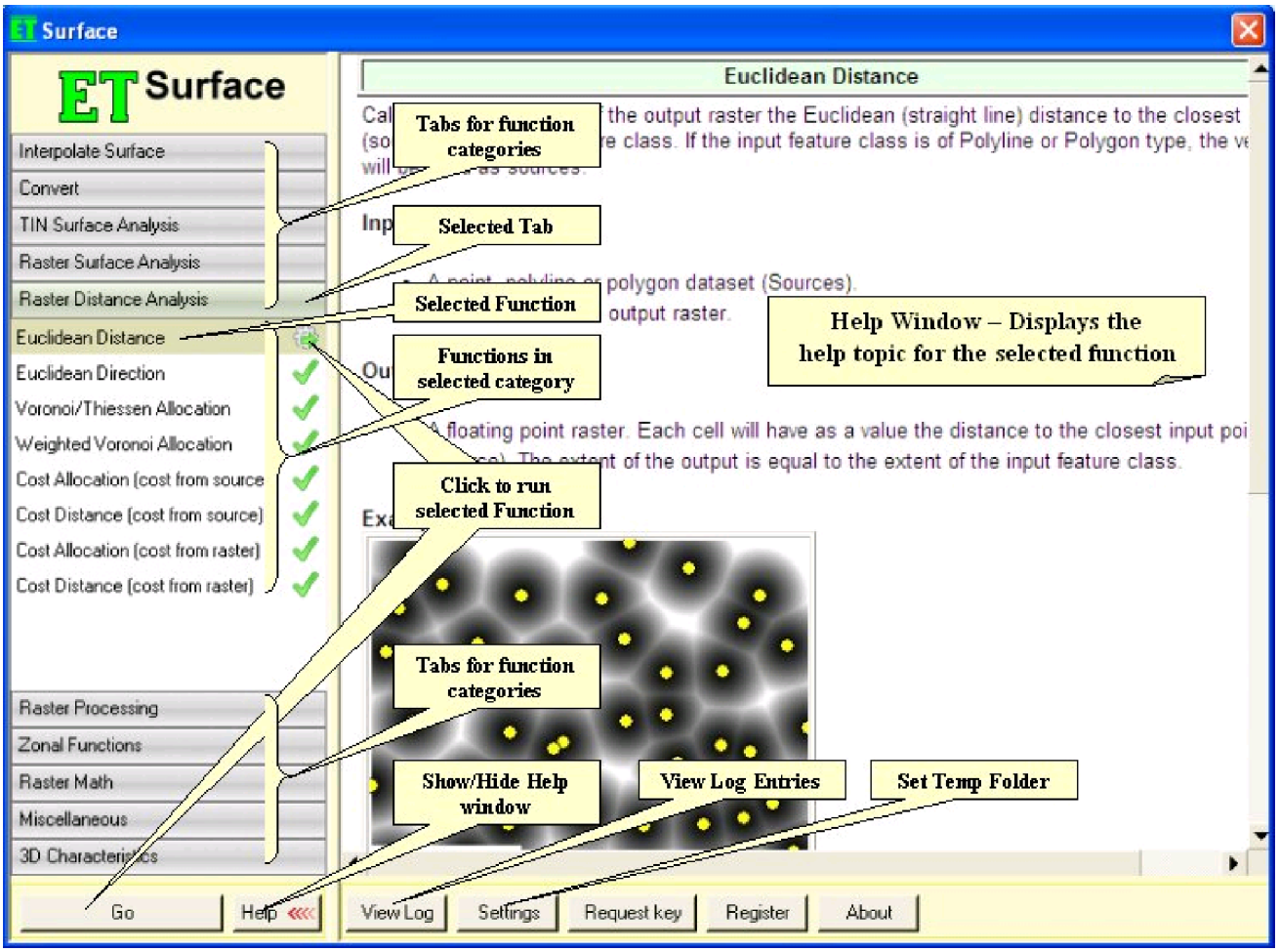
On the registered software only and icons should appear.

Clicking on the GO button will execute the selected function.

The User Guide is embedded in the main dialog - whenever you select a function, the Help Window will display the appropriate help topic. You can use the Help button to hide or show the Help Window.

The View Log button displays the entries recorded in the ET Surface log file. The dialog allows deleting the current entries. It is recommended to clean the log file on regular intervals

The settings button opens the settings dialog of ET Surface. On this dialog you can view the current temp folder (where all intermediate datasets created by the functions of ET Surface are stored) or set a new folder to be used for such purposes. ET Surface cleans the temp folder automatically, but it is a good practice to delete all the contents of this folder from time to time.



# How to use ET Surface functionality in .NET

Most of the functions of ET Surface (starting from version 6.1) can be used in custom applications (stand alone, ArcGIS Add - Ins, custom controls). The syntax of each ET Surface function is described in the main topic of the function ==> .NET implementation

Quick start - VB.NET example for stand alone application

Prerequisites

- ArcGIS - installed and licensed
- ArcObjects SDK for NET Framework - installed
- Microsoft Visual Studio
- ET Surface 6.1 and above - installed and registered
- Start Visual Studio
- Go to File ==> New ==> Project
  - In the dialog go to Installed Templates ==> Visual Basic ==> ArcGIS ==> Extending ArcObjects and select Windows Application (Desktop)
  - Give a name to your project and click OK
  - In the ArcGIS Project Wizards that will open select your product (Basic, Standard or Advanced) and click Finish
- Go to Project ==> Properties ==> References:
  - Add reference ==> .NET ==> find ESRI.ArcGIS.Geodatabase and select it ==> Click OK
  - Add reference ==> Browse ==> navigate to the installation folder of ET Surface and select ETSurface61.dll. Make sure that "Copy Local" is set to true in the reference properties.
- Save the Assembly and Build it.
- Create a button on your form and name it
- Double click on the button to start editing the code
- Paste the code below

```
Imports ETSurface
Imports ESRI.ArcGIS.Geodatabase
Public Class Form1
 Private Sub BuildTin(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnBuildTin.Click
  Try
   Dim etsRun As ETSCore = New ETSCore
   Dim resultTin As ITin = Nothing
   'Input point shape filev    Dim shapePath As String = "F:/Test"
   Dim shapeName As String = "elevPoints.shp"
   Dim pointFC As IFeatureClass = featureClassFromString(shapePath, shapeName)
   'Output TIN name
   Dim outTin As String = "F:\Test\Output\areaTin4"
   'Elevation Field in the shape file
   Dim elevationField As String = "ET_Spot"
   'Triangulation method
   Dim triangulationTipe As String = "Mass points"
   'Run the function
   resultTin = etsRun.BuildEsriTin(pointFC, outTin, elevationField, triangulationTipe)

   If resultTin Is Nothing Then
    MsgBox("Tin creation failed")
   Else
    MsgBox("Tin created successfully")
   End If
  Catch ex As Exception
   MsgBox("Error:" & ex.Message)
  End Try
 End Sub

 Private Function featureClassFromString(ByVal sPath As String, ByVal sName As String) As IFeatureClass
   Try
     Dim pFeatureClass As IFeatureClass

     Dim t As Type = Type.GetTypeFromProgID("esriDataSourcesFile.ShapefileWorkspaceFactory")
     Dim obj As System.Object = Activator.CreateInstance(t)
     Dim pWorkspaceFactory As IWorkspaceFactory = CType(obj, IWorkspaceFactory)
     Dim pFWS As IFeatureWorkspace = CType(pWorkspaceFactory.OpenFromFile(sPath, 0), IFeatureWorkspace)
     pFeatureClass = pFWS.OpenFeatureClass(sName)

     Return pFeatureClass
   Catch ex As Exception
     MsgBox("Error getting Feature Class" & ex.Message)
     Return Nothing
   End Try
 End Function
End Class
```

- Copy from the ET Surface installation folder **all DLL files and EtgApp.exe** to the folder of your application executable.
- Make sure that the specified files and folders exist. Compile and run the application.

Below is another example - using the TinToRaster function:

```
Private Sub btnTinToRaster_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnTinToRaster.Click
  Try
    Dim etsRun As ETSCore = New ETSCore
    Dim resultRaster As IRasterDataset2 = Nothing
    'Input TIN
    Dim tinPath As String = "F:\Test\Output"
    Dim tinName As String = "areaTin"
    Dim areaTin As ITin = TinFromPath(tinPath, tinName)
    'Output Raster name
    Dim outRaster As String = "F:\Test\Output\areaRaster.img"
    'Output raster cell size
    Dim cellSize As Double = 20
    'Run the function
    resultRaster = etsRun.EsriTinToRaster(areaTin, outRaster, cellSize)

    If resultRaster Is Nothing Then
      MsgBox("Raster creation failed")
    Else
      MsgBox("Raster created successfully")
    End If
  Catch ex As Exception
    MsgBox("Error:" & ex.Message)
  End Try
End Sub


Public Function TinFromPath(ByVal sPath As String, ByVal sName As String) As ITin
  Try
    Dim pTin As ITin = Nothing

    Dim t As Type = Type.GetTypeFromProgID("esriDataSourcesFile.TinWorkspaceFactory")
    Dim obj As System.Object = Activator.CreateInstance(t)
    Dim pWF As IWorkspaceFactory = CType(obj, IWorkspaceFactory)
    Dim pTinW As ITinWorkspace = CType(pWF.OpenFromFile(sPath, 0), ITinWorkspace)
    If pTinW.IsTin(sName) Then
      pTin = pTinW.OpenTin(sName)
    Else
      MsgBox("Invalid Tin name")
      Return Nothing
    End If
    Return pTin
  Catch ex As Exception
    MsgBox("Error getting TIN" & ex.Message)
    Return Nothing
  End Try
End Function
```

# ET Surface - Functions and surface types

| Features | | | | |
|---|---|---|---|---|
| Function | Surface Type | | | Features |
| | Raster | ESRI TIN | PolygonZ TIN | |
| Profile Extractor | | | | |
| Draw Cross-Section | ☐ | ☐ | ☐ | |
| Profile for selected graphic | ☐ | ☐ | ☐ | |
| Profile for selected feature | ☐ | ☐ | ☐ | |
| Line of Site | ☐ | ☐ | ☐ | |
| Digitize Z Geometries | | | | |
| PointZ | ☐ | ☐ | ☐ | |
| PolylineZ | ☐ | ☐ | ☐ | |
| PolygonZ | ☐ | ☐ | ☐ | |
| ET Terrain Viewer | | | | |
| | ☐ | | ☐ | ☐ |
| ET Surface Functions | | | | |
| Interpolate Surface | | | | |
| Build TIN | | | | ☐ |
| Modify TIN | | ☐ | ☐ | ☐ |
| Contours To Raster | | | | ☐ |
| IDW | | | | ☐ |
| Density | | | | ☐ |
| Features To 3D | ☐ | ☐ | ☐ | |
| Convert | | | | |
| ESRI TIN to PolygonZ TIN | | ☐ | | |
| PolygonZ TIN to ESRI TIN | | | ☐ | |
| TIN to Edges | | ☐ | ☐ | |
| TIN to Nodes | | ☐ | ☐ | |
| Polygon To Multipatch | | ☐ | ☐ | |
| Multipatch To Polygon | | | | ☐ |
| Features To Raster | | | | ☐ |
| ESRI TIN To Raster | | ☐ | | |
| TIN Surface Analysis | | | | |
| TIN Slope | | ☐ | ☐ | |
| TIN Aspect | | ☐ | ☐ | |
| Interpolate Contours | | ☐ | ☐ | |
| Volume of TIN | | ☐ | ☐ | |
| Volume of Polygons | | ☐ | ☐ | |
| Cut/Fill Analysis | | ☐ | ☐ | |
| Visibility Analysis | ☐ | ☐ | ☐ | |
| Identify Peaks and Sinks | | ☐ | ☐ | |
| Raster Surface Analysis | | | | |
| Raster Slope | ☐ | | | |
| Raster Aspect | ☐ | | | |
| Raster Hillshade | ☐ | | | |

| | | | | |
|---|---|---|---|---|
| Interpolate Contours from Raster | ☐ | | | |
| Viewshed | ☐ | | | |
| Cut/Fill Analysis | ☐ | | | |
| Raster Volume | ☐ | | | |
| Raster Distance Analysis | | | | |
| Euclidean Distance | ☐ | | | |
| Euclidean Direction | ☐ | | | |
| Euclidean (Voronoi) Allocation | ☐ | | | |
| Weighted Voronoi Allocation | ☐ | | | |
| Cost Allocation (Cost from source) | ☐ | | | |
| Cost Distance (Cost from source) | ☐ | | | |
| Cost Allocation (Cost from raster) | ☐ | | | |
| Cost Distance (Cost from raster) | ☐ | | | |
| Raster Processing | | | | |
| Clip Raster with Envelope | ☐ | | | |
| Clip Raster with Polygons | ☐ | | | |
| Erase Raster with Polygons | ☐ | | | |
| Smooth Raster | ☐ | | | |
| Clean Boundaries | ☐ | | | |
| Create Constant Raster | ☐ | | | |
| Create Random Raster | ☐ | | | |
| Change Raster Data Type | ☐ | | | |
| **Raster Maths** | | | | |
| Raster Calculator | ☐ | | | |
| Replace NODATA | ☐ | | | |
| **Zonal Functions** | | | | |
| Zonal Statistics | ☐ | | | |
| 3D Characteristics | | | | |
| PolylineZ Characteristics | | | | ☐ |
| Polygon 3D Characteristics | | ☐ | ☐ | |
| Analyze TIN | | | ☐ | |
| Miscellaneous | | | | |
| Multiply Zs | | | ☐ | ☐ |
| Offset Zs | | | ☐ | ☐ |
| Split PolylineZ Based on Slope | | | | ☐ |
| Clean Spikes in PolylineZ | | | | ☐ |

# ET TerrainViewer

**Overview**

ET Terrain Viewer allows you to view your GIS data in 3 dimensions. You can load in the viewer Raster or TIN surfaces, Z enabled features and even plain 2D features by draping them on an existing surface. Then you can navigate through the 3D environment using several different navigation techniques.

Main features of ET Terrain Viewer are:

- **Simple and intuitive user interface.**
  The software was designed to be immediately intuitive and easy to learn
- **Hardware-accelerated 3D visualization.**
  ET Terrain Viewer utilizes the available graphics hardware to provide better performance. Rendering is performed using Direct3D. A graphics card with hardware transformation and lighting (T&L) is recommended.
- **Mini-map of the loaded terrain.**
  An interactive mini-map with an overview of the terrain, showing the position and looking direction of the observer.
- **Multiple layer support.**
  ET Terrain Viewer can load and display multiple raster and / or shape files simultaneously.
- **Layer manipulation:**
  - User-defined elevation, drape layer over terrain
  - Extrusion
  - Z-scale
  - Resolution reduction for increased performance.
  - Symbology supports gradient colors and textures based on elevation or custom attributes
  - Customizable point-markers for point layers
- **Navigation using both mouse and keyboard**

- **Tools:**
  - Identify space coordinates and elevation
  - Walking over the terrain at user-defined offset from the ground
  - Define a route and have the observer follow it automatically
  - Dynamic symbols in graphic layer
  - Customizable lights, skyboxes and fog effects
- **Export to popular graphics formats - JPEG, PNG, BMP.**
  Save a snapshot of the 3D image as a picture.

See what is new in version 3.0.

ET Terrain Viewer is a single-document application, and you can work with only one project at a time. ET Terrain Viewer starts with an empty project. To add layers to your project, click **Add Layer** from menu **Layers**. Note that the first layer you add to a project must be a terrain layer - raster grid or TIN shape file. To open an existing project, click **Open Project** from menu **File**.

Get started with the user interface.

To learn how to navigate the surface and use the tools, read section Tools and Navigation.

**What is New in version 3.0**

The new featues of ET Terrain Viewer version 3.0 are:

**Enhanced Symbology**:

Symbology has been completely redesigned:

- Symbols now can be defined for data attributes from the shapefile
- For extruded layers, extrusion symbology can be configured separately
- More than one texture symbol can be used per layer
- Surface layers blend textures to improve visual effects

**Image Layers**

Three band rasters can be draped over the surface. See supported Data formats.
Image layers need to be in the same coordinate system as the surface.

**Graphic Layers**

You can now add dynamic point markers to the surface, and store them as part of the project.

**Tools**

Two new tools - Journey and Graphic Layer Editor are now available. See section Tools for more information.

**New Environment Settings**

The intensity of illumination can now be adjusted. Further, you can set a sky box from a list of pre-defined skies - clear, light clouds, heavy clouds, night. You can add fog to the scene to simulate the distance fading, or really bad weather.

**System requirements**

Minimum requirements:

- Intel Pentium 4 processor, running on 2.4GHz
- 1 GB RAM
- Video card, supporting DirectX 9 with 64 MB video memory
- OS: Windows XP with Service Pack 2

Recommended requirements:

- Intel Pentium 4 processor, running on 3 GHz
- 2 GB RAM
- Video card, supporting DirectX 9 with hardware transformation and lighting

## Configuring

### General Options

The following options apply to all projects.

- Fill Mode - defines how layers are rendered. Possible values are:
  - Solid (default) - Layers are rendered as a solid surface.
  - Wire frame - Only the outline of layers is rendered.
- Backgrounds - Here you can set the background colors of the 3D surface, and the mini-map.

### Ramps

You can create a library of predefined color ramps, which you can use later to quickly create desired layer symbology.

Color ramps consist of at least two colors. You can add up to 50 colors in a color ramp. ET Terrain Viewer will automatically create a gradient between the distinct colors of the ramp if the number of classes of the symbology is greater than the number of distinct colors in the ramp.

### Point Markers

You can create a library of point markers, which you can use later for multi-point layers.

Point markers can be created from:

1. Images - click button Add Image to select an image file. The image must be no larger than 128 x 128 pixels. It is recommended that you use images in format PNG with defined transparency to achieve best effect.
2. Symbols - you can add symbols from installed fonts on your computer. ET Terrain Viewer provides a font named ETTV2, which contains a number of symbols. Click button Add Symbol, then select the desired font, select the color for the symbol, and type in the desired symbol, or select one from the drop-down list. ET Terrain Viewer will convert the symbol into a transparent image, and add it to the library.

**Data Formats**

ET Terrain Viewer supports the following data formats

**Surface layers**:

- Raster datasets - currently single band rasters are supported only
  - ESRI binary grid
  - Erdas Imagine image (.img)
  - TIFF format (.tif)
  - ESRI Ascii Grid
- TIN datasets - PolygonZ TIN created with ET Surface or ET GeoWizards and stored in a shapefile. Note that if you have a data in ESRI TIN format, you can easily convert it in PolygonZ TIN using the functions of ET Surface

**Feature layers** - currently shapefiles are supported only:

- Point, PointZ
- Polyline, PolylineZ
- Polygon, PolygonZ
- Multi-Patch

**Image Layers** - three band rasters are supported:

- Erdas Imagine image (.img)
- TIFF format (.tif)
- JPEG

Note that the data to be used needs to be in a projected coordinate system. All the layers loaded in a single project need to have the same coordinate system.

For reading raster datasets ET TerrainViewer utilizes Geospatial Data Abstract Library (GDAL) by Frank Warmerdam. - http://www.gdal.org/. GDAL is released under anX/MIT style Open Source license by the Open Source Geospatial Foundation.

## Working with Layers

### Layer Types

ET TerrainViewer uses four types of layers

- Surface layers - the sources of the surface layers can be
  - PolygonZ TIN created with the Build TIN function of ET Surface or ET GeoWizards
  - Raster datasets - see supported formats here
- Feature layers - shapefiles with or without Z values.
- Image layers - three-band raster datasets (e.g. aerial photos). See supported formats here.
- Graphic layers - a custom layer of point markers, created manually. One graphic layer per project is supported.

The software checks the dataset that is loaded and determines the type of the layer automatically. The first layer loaded in a new project must be a surface layer.

### General Layer Properties

#### Z Values

You can define how the Z values of the layer will be derived.

- Feature layers:
  - User-defined Z value: this options allows you to set a constant Z value for all features in the layer. The option is not applicable for surface layers.
  - Drape to layer: this option forces ET Terrain Viewer to obtain the Z value for all features in a layer from the elevation of the surface layer at the same location. A constant offset above the source surface can be set. The option is not applicable for terrain layers.
  - From original shape file: If the Shapefile has Z values (PointZ, PolylineZ or PolygonZ or Multi-patch).
- Surface layers: The Z values are derived from the original data - pixel value for rasters, Z values of the triangles for the PolygonZ TINs
- Image layers: By default Z values are set to zero. Image layers can be draped to the surface layer.

#### Extrusion

Extrusion turns points into lines, and lines into walls. The lowest point of the extrusion is the minimum of the terrain layer. The option is not applicable for terrain and image layers.

#### Z Scale

Z scale can be used to increase the Z values of the layer by multiplying them by the factor entered. The multiplying factor cannot be zero. A multiplying factor of 1 basically negates scaling.

#### Resolution of Surface layers

You can adjust the level of details for terrain layers. 100% means that the layer is displayed with in its finest details. The higher the resolution, the better is the quality of the layer; the lower the resolution, the better is the rendering performance of the viewer.

ET Terrain Viewer automatically reduces the resolution of large terrain layers - raster or TIN. If your hardware can cope with

better details, you can increase the resolution once the layer is added to the project.

The option is applicable to terrain layers only.

Decreasing resolution of large terrains is a slow operation. The lower the desired resolution, the more it takes to simplify the terrain.

**Layers Symbology**

Symbology defines how a layer would be visualized on screen.

**Definitions:**

- **Marker** - symbol for drawing point features or points on the graphic layer. Point markers can be created from:
  - Images - click button Add Image to select an image file. The image must be no larger than 128 x 128 pixels. It is recommended that you use images in format PNG with defined transparency to achieve best effect.
  - Symbols - you can add symbols from installed fonts on your computer. ET Terrain Viewer provides a font named ETTV2, which contains a number of symbols. Click button Add Symbol, then select the desired font, select the color for the symbol, and type in the desired symbol, or select one from the drop-down list. ET Terrain Viewer will convert the symbol into a transparent image, and add it to the library.
- **Texture** - an image to be used for rendering features representing area (polygons, extrusion walls) features or surfaces. Supported image formats are BMP, JPEG and PNG. The maximum size of textures is 512 x 512 pixels. The texture images can be tiled to fill the polygons, so in order to avoid getting ”waffle” effect you need to use so called seamless images. Some seamless textures are provided with the installation of the software.
  Note: Using large and many textures or using finest tiling can have a negative impact on performance.
- **Color Ramp** - array of colors used to render continuous data. ET Terrain Viewer provides several pre-defined color ramps. The user can create new color ramps on the Options panel (Tools - Options - Ramps - Add) by providing two or more colors to be used in the ramp.

**Classification:**

- **Single Value** - A single symbol is used to render the entire surface or all features of the feature class.
- **Range** - You specify the number of classes required. Initially, the software calculates the class breaks using the minimum and maximum values and the number of classes requested, creating equal intervals. You can later to adjust the boundaries of the each interval. Adjacent intervals will be adjusted automatically to avoid fragmentation. You can change the class symbol (color, texture or point marker) and class breaks by double clicking on a specific class and change the values in the class properties dialog.
  If the selected symbol is “Color” the current color ramp is used to populate the symbols for the different classes. If the selected color ramp contains less colors than the number of classes, ET Terrain Viewer will generate the necessary gradient colors to match the number of classes.
  If “Texture” or “Point Marker” is selected the user needs to select texture image / point marker for each class.
- **Unique Values** – If the attribute table of the feature class has a string or integer field that defines different categories of features, unique value classification can be used. The user needs to select an integer or string field that has no more than 50 unique values.

ET Terrain Viewer can display three major layer types:

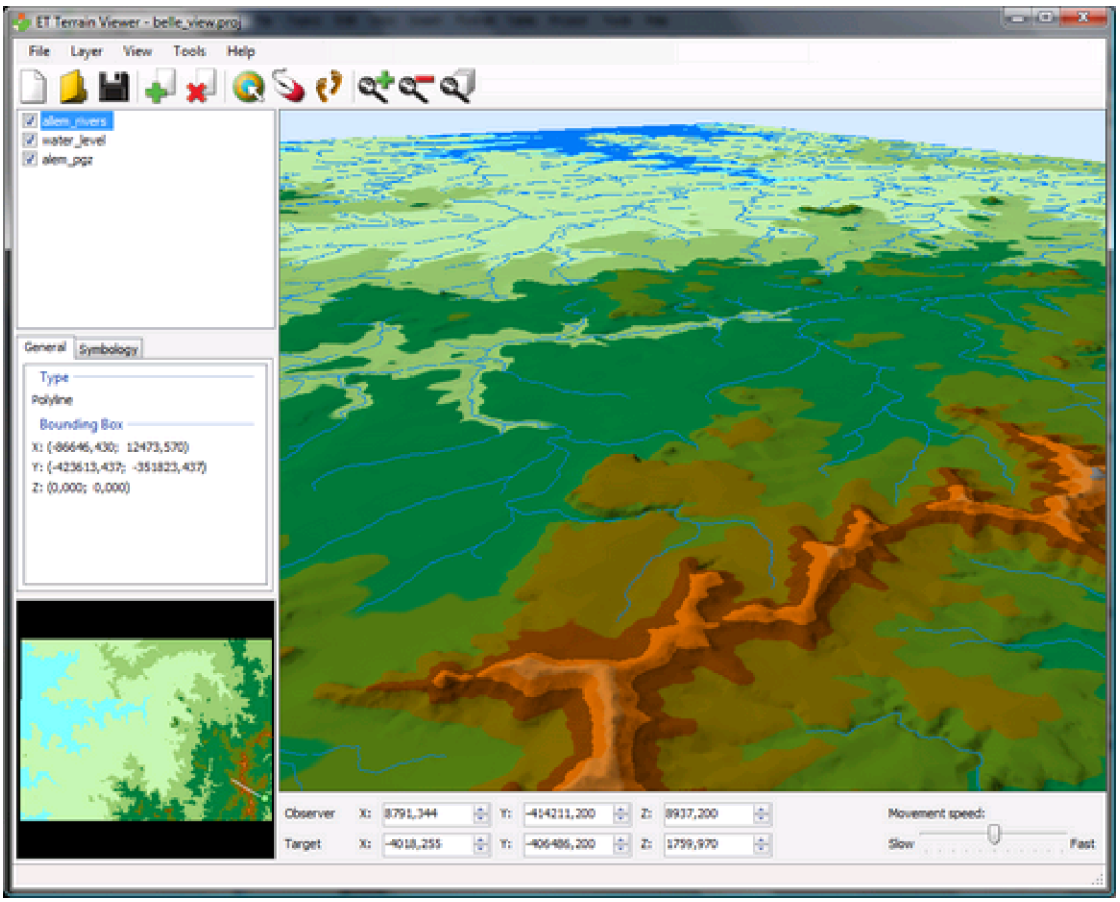
- Surfaces (Rasters or TINs) can be rendered only based on elevation. You can use for rendering a color ramp or texture images.
- Features can be rendered based on their elevation or the values in the selected attribute field. Depending on the geometry type the following symbol options are available:
  - Point – By default, when a new point layer is added to a project, the first marker from the list of predefined markers is assigned to the layer. You can select a marker for the layer from the already existing ones, or add a new one.
  - Polyline – Only color can be used.

    - Polygon – Colors and textures can be used.
    - Multipatch - Colors and textures can be used.
    - Polygon and Polygon Extrusion (the walls created if the features are extruded) - Colors and textures can be used.
    - Extruded Point (the lines created) - Only color can be used.
- Georeferenced images (e.g. aerial photos) – the actual image colors. No symbology is available.

**User Interface**

The user interface of ET Terrain Viewer consists of:

- Menu and Toolbar
- Project Manager
- Layer information panel
- Interactive mini-map
- Main 3D display
- Navigation cockpit



# ET TerrainViewer

**System requirements**

#### Default Navigation

By default, no tool is active, and you can navigate the surface using the keyboard and the mouse.

- Keyboard:
    - Arrow Up - move the observer towards the target.
    - Arrow Down - move the observer backwards from the target.
    - Arrow Left - turn left.
    - Arrow Right - turn right.
    - Left Shift + Arrow Left - move the observer to the left.
    - Left Shift + Arrow Right - move the observer to the right.
    - Page Up - look up.
    - Page Down - look down.
    - Home - zoom in.
    - End - zoom out.
- Mouse:
    - Double-click - move the observer at the position of the double-click. If the elevation of the observer was lower than the elevation of the terrain at the new position, the observer is elevated to be above the terrain.
    - Right-click - turn the observer to the point of the click.
    - Rotate Mouse Wheel forward - zoom in.
    - Rotate Mouse Wheel backwards - zoom out.

#### Mouse Navigation Tool

This tool enables more extensive navigation using the mouse:

- Hold the left button and move the mouse to look around.
- Hold the right button and move the mouse to pan the surface.
- Hold the wheel and move the mouse up and down to change the elevation of the observer.

Navigation with arrow keys is active and the same as in Default Navigation mode.

#### Walk Mode

Use this tool to simulate a walking around the terrain, with the observer always being at a fixed offset from the terrain. You can adjust this offset from the Walk Mode popup windows. The offset cannot be less than 3 elevation units. If the observer is walking too close to the terrain, you can experience transparency of the nearest faces of the terrain.

- Keyboard
  Navigation using the keyboard is the same as in Default Navigation mode, except for zooming (Home and End), which is disabled.

- Mouse Navigation:
  - Hold the left button and move the mouse to look around.
  - Rotate mouse wheel to walk forwards or backwards.

## Tools

All commands on the toolbar are also accessible from the menu.

 **Start New Project**

Start new project. If the current project has already contents added, the user will be prompted to save or discard the changes.

 **Open Existing Project**

Open an existing project. If the current project has already contents added, the user will be prompted to save or discard the changes.

 **Save Project**

Saves the current project. To save the current project into a new file, click command **Save as** from menu **File**.

 **Add Layer**

Add a new layer to the current project. If the current project does not contain layers yet, this layer must be a terrain layer. See supported Data Formats for more details.

 **Remove Layer(s)**

Remove the currently selected one or multiple layers from the current project. You will be prompted to confirm the operation.

 **Identify Tool**

Use this tool to get the coordinates of a point of the terrain surface. When the tool is active, a popup window with coordinates of the selected point will appear. Left-click on the terrain to get coordinates of the point. The popup window displays the coordinates of the last selected point, and does not change when navigating. Navigation works just like default navigation, except for moving the observer by double click. To move the observer to a desired location, use arrow keys or the mini-map.

 **Mouse Navigation**

Toggles mouse-navigation mode. See section Navigation for more details.

 **Walk Mode**

Toggles walk mode. See section Navigation for more details.

 **Journey Tool**

Use Journey tool to define a route through the terrain surface, and have the observer move along that route. The observer can either walk at a defined offset, fly over the route at a defined height. The default flight height is the maximum terrain height plus 10%. You can import a route from a polyline shapefile, but only the first polyline in the Shapefile will be used.

 **Graphic Layer Tool**

Use Graphic Layer tool to add and edit a graphic layer to your project. Graphic layers are special point layers, which allow users to dynamically display point markers on the terrain surface. When Graphic Layer tool is active, you can use the left mouse button to plant the selected symbol on the surface. Identical symbols are grouped together into *groups*. You can delete the last added symbol, a whole group of symbols, or the whole graphic layer. You can change the symbol of a group through the Layer Symbology dialog for the Graphic layer.

Graphic layers are saved within their project main file.

 **Zoom In**

The function moves the observer closer to the scene, following the looking direction.

 **Zoom Out**

The function moves the observer farther from the scene, following the looking direction.

 **Zoom To All**

Reset the position of the observer to a point, from which all layers are visible in their full extent.

**Using the Project Manager**

A project in ET Terrain Viewer is the current set of loaded layers with their properties. You can save your work in a project and open it later. When you start ET Terrain Viewer, an empty project is automatically created. The first layer in a project must be a terrain layer. This is either a raster file, or a TIN file of type Polygon with Z values, where polygons have three vertices. See supported Data Formats for more information.

The Project Manager displays a list of all currently loaded layers. Layers are rendered in reverse order, i.e. the bottom layer is rendered first, and the top layer is rendered last.

You can perform the following tasks:

- **Exclude a layer from rendering**
  Uncheck the check box next to the layer to exclude it from the scene. Check the box to render it again.
- **Change the order of layers**
  Layers are rendered in reverse order. Use drag and drop to reorder layers. Drag a layer to its new location upper or lower in the list and drop it there.
- **Remove one or more selected layer from the current project**
  Click command Remove from the context menu to remove the selected layer(s).
- **Zoom to selected layer(s)**
  Click command Zoom to layer from the context menu. The command will zoom to the selected layer. If multiple layers are selected, the command will zoom to the union extent of the selected layers.
- **Zoom to all layers in the project**
  Click command Zoom to all from the context menu to zoom to the union extent of all layers in the project.
- **Set layer properties**
  Click command Properties from the context menu to open the dialog with the properties of the selected layer.
  ***NOTE: In order to avoid incorrect data display all the layers loaded in a single project need to have the same coordinate system and fall within the same extent.***

## Using Mini-Map and Cockpit

### Mini-Map

The mini-map displays an overview of the loaded terrain surface, as seen directly from above. Other layers are not visualized.

You can use the mini-map to quickly position the observer at a certain location on the terrain, or to change the looking direction.

**Position the observer at a new location:**

- Double-click on the desired location on the mini-map

  or

- Click Set observer from the context menu of the mini-map.

**Change the direction at which the observer looks**

Click Look at Direction from the context menu of the mini-map.

### Cockpit

The cockpit displays the current position of the observer in 3D space, and the point at which the observer is looking. The cockpit is automatically updated during navigation.

You can change the coordinates of both the observer and looking target point to achieve precise control over the positioning and orientation of the observer.

Further, you can use the Movement speed track bar to change the navigation speed. The speed is relative to the extent of the terrain surface.

Note: After you have changed a value in any of the boxes of the cockpit, you need to click on the main 3D window in order to be able to use the navigation controls

# Profile Extractor

**Profile Tools**

All profile tools are applied to the surface layers loaded in ArcMap. Surface layers supported by Profile Extractor are ESRI TIN, Raster, PolygonZ TIN created by ET Surface. If the PolygonZ TIN is created by ET GeoWizards or third party, it should be analyzed first using the Analyze TIN function of ET Surface.

When a profile tool is used, the resulting profile(s) are displayed in the Profile Window. If the Profile Window does not exist, profiles are created for all visible surfaces in the Map view. The user can add and remove surfaces from the Profile Window. If the Profile Window exists, when a tool is applied, profiles are created for all surfaces defined in the Profile Window.

Three tools are available to the user to input/identify the cross-section lines to be used for drawing a profile:

- Draw Cross-Section line tool  - the user draws free hand polyline on the view. If the Profile Window is not open already, the profile for this polyline is extracted from all visible surfaces in the Map view and displayed in the Profile Window. If the Profile Window exists, profiles are extracted from the surfaces defined in the Profile Viewer.
- Draw profile for feature  . The tool allows the user to point to a feature from a feature layer in the Map and creates profiles for this feature. The tool is only active, if there is a feature layer selected in the Surface Layer box on the ET Surface Toolbar. The selected feature should be of Polyline or Polygon type. With the profile window open, the user can change the selection, the contents of the profile will be updated to display the data for the newly selected feature. If the selected feature layer has Z values, a profile will be created and displayed using the Z values from the feature in addition to the surface profiles (with profile name of PolylineZ). If the Profile Window is open and does not contain PolylineZ profile, when the user selects a feature with Z values, he can add the PolylineZ profile by selecting Add Surface from the File Menu and selecting PolylineZ from the selection list.
- Draw profile for graphic  . The tool allows the user to point to a graphic on the Map and creates profiles for this graphic. The selected graphic element should of Polyline or Polygon type. When the tool is applied, the graphic is selected and the profile is extracted and displayed in the Profile Window. With the profile window open, the user can change the selection, move or reshape the graphic using the standard graphic tools and the contents of the profile will be updated to display the data for the newly selected or edited graphic.

Notes:

- In order to improve the performance of the tools, switch OFF the visibility of the surface layers for which you do not want to extract profiles.
- If any of the tools above is used when the Profile Window is closed several of the parameters are calculated based on the length of the cross-section polyline and the difference between the Min and Max Z values and the settings of the Profile Windows are adjusted accordingly.

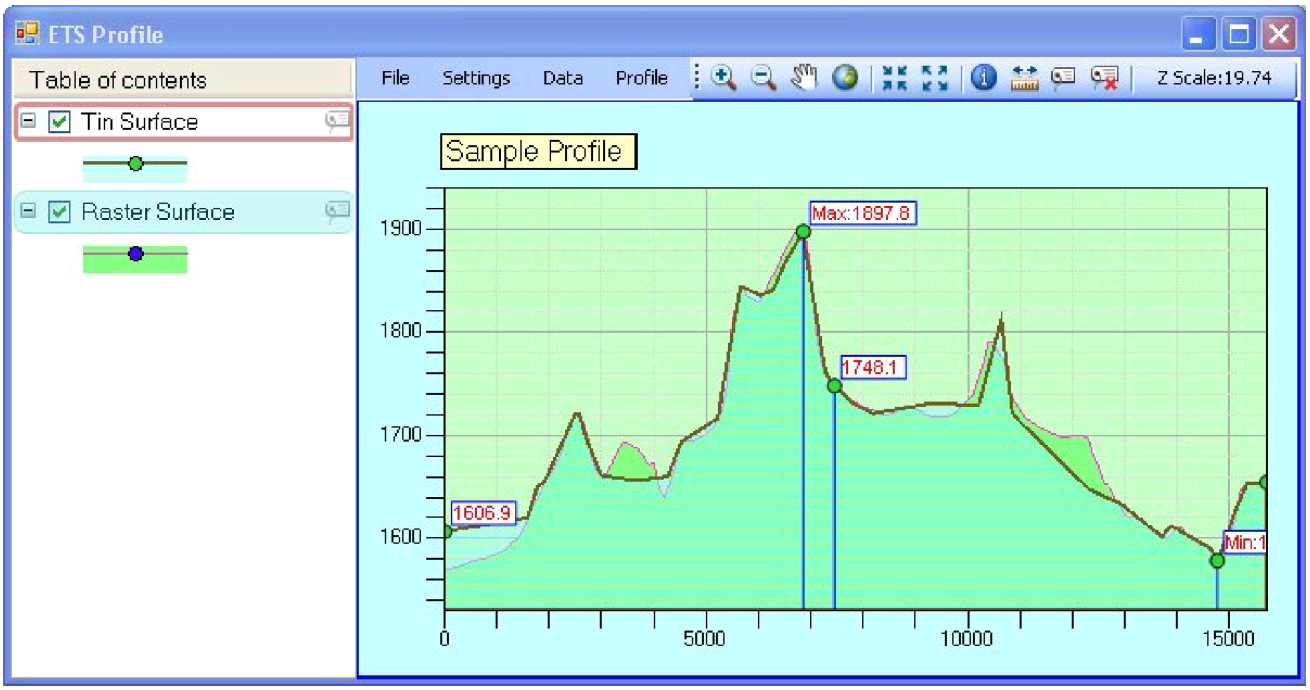
# Profile Extractor

**Profile Window**

The Profile Window is the dialog where the profiles are displayed. The user can adjust various settings and once happy with the results, export the data or image of the profile, draw the profile in the Data Frame or the Layout.

Functionality in the Profile window is available through the Main Menu displayed on top of the dialog, the Main Toolbar (next to the Main Menu) and the Context Menu for the Table of Contents (right click on a layer name in the TOC).

Over the chart area the user can use the mouse wheel to zoom in by turning it forward or zoom out by turning it backward. Pressing the mouse wheel (or middle button) and moving the mouse will pan the profile accordingly. Pressing the right mouse button will create a tooltip displaying the Station and Height at the mouse location. To remove the tooltip, press the right button again.




- Table of contents (TOC) - located on the left side shows the names of layers displayed in the Profile Window. The layer at the top of the table is displayed on top in the Profile Window. From the TOC the user can control the following:
  - The order in which the layers are displayed by dragging and dropping the TOC items up and down the list
  - The visibility of each layer by clicking on the check box
  - Expand the layer by clicking on the plus/minus sign to show the layer symbol(s)
  - The style for drawing the layer by opening the Properties Window - either double click on a Layer name of right lick and select Properties from the context menu
  - Show Data Points for a profile by right clicking on the Layer name and checking the "Show Data Points" status
  - Set the Identity layer by selecting the "Set Identify" from the context menu. There can be only one Identity layer in the Profile Window. It is displayed with a brown outline in the Table of contents.
  - Include or exclude the Profile in the list of layers for which labels will be created when using the Label tool (by checking the value in the context menu). The layers enabled for labels have a label icon on the right hand side in the Table of contents.
  - Remove the Layer from the Profile Window by right click and select "Remove layer" from the context menu
- Main menu
  - File Menu - include the following items
    - Hide/Show Table of contents - hides or shows the Table of contents
    - Save Project - allows the user to save the current settings in a project file. The drawing styles for all layers are saved as well as the general profile settings. If a project file is used its name is displayed in the title of the window.
    - Save Project As - saves the current settings in a new project file which becomes the current project file
    - Open Project - opens a project file (.etprj). When the user selects the project file, its contents are checked against the current Map to determine the existence of layers. A status window is displayed showing the required layers and their availability in the Map. If all the layers in the project file are found in the Map and comply with the requirements, the file is loaded and applied to the current profile polyline.
    - Add Surface - allows the user to add another surface layer to the Profile Window. All surface layers, which are not already present are included in the selection list (including not visible layers) If all surface layers in the Map are already present in the Profile Window, a message is displayed to inform the user.
    - Add Layer - see Adding Layer options
    - Save profile in currently edited layer - see Export options
    - Save profile in a new feature class - see Export options
    - Export profile as text file - see Export options
    - Save as image - see Export options
  - Settings Menu
    - Profile Settings - opens the Options window allowing the user to change general settings - Title, colours and fonts for display of the

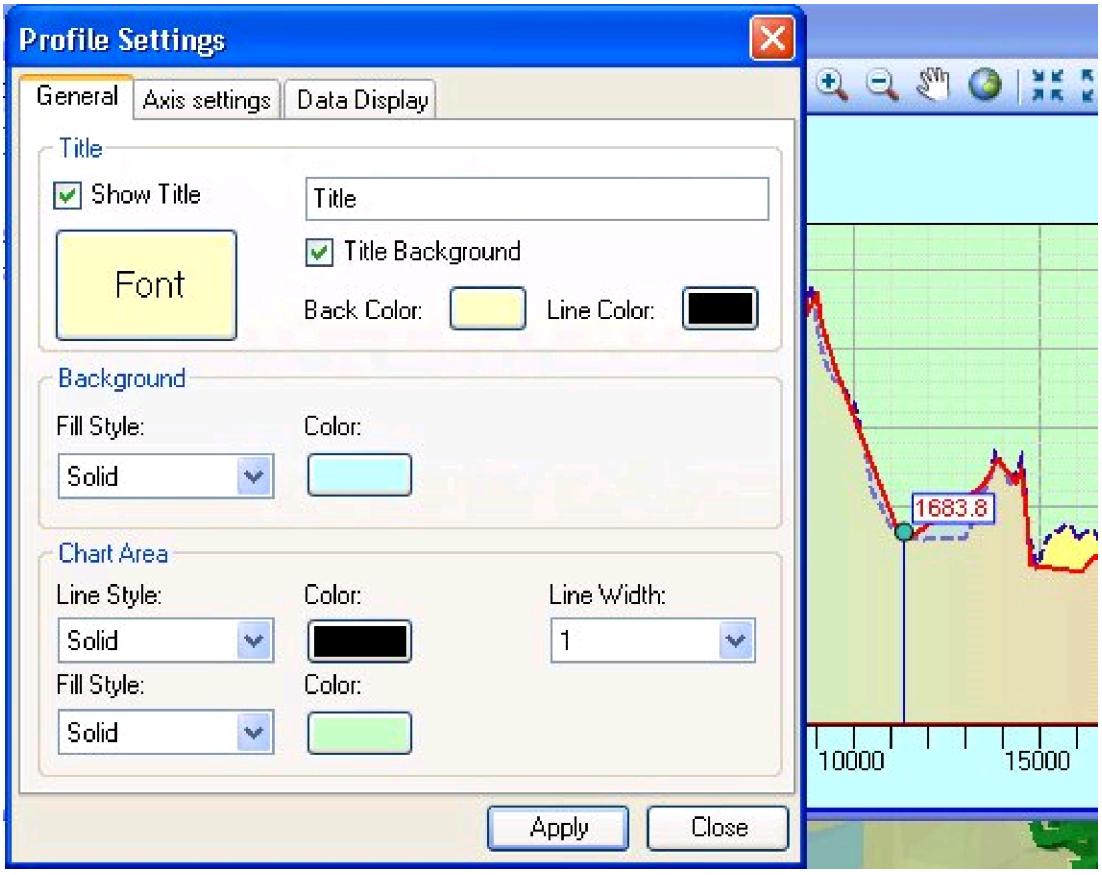
background, grids, axis settings as well as settings for data display
        - Line of sight settings - allows the user to change settings for Line of Sight display - only available with Line of Sight display
        - Sample Distance - allows the user to change the sample distance for profile generation
    - Data Menu
        - View Profile Data - displays X, Y and Z for each data point of a profile together with the station (distance from the start of the cross-section polyline, along the polyline). If multiple profiles are drawn in the Profile Window, the data for all profiles can be displayed by selecting the "All" option. Since profiles of different types may have different data points, the data points are merged and sorted, and Z values for each surface calculated for the resultant list of data points.
        - View Profile Statistics - displays several important characteristics of each profile:
            - 3D Length
            - 2D Length
            - Maximum Z value
            - Minimum Z value
            - Length Uphill
            - Length Downhill
            - Average Slope Uphill
            - Maximum Slope Uphill
            - Average Slope Downhill
            - Maximum Slope Downhill
    - Profile Menu
        - Add Label at Station - adds a label on the profiles at user specified station along the cross-section polyline
        - Delete All Labels - removes all labels displayed on the Profile Window
        - Profile Cross-sections Along Route - see Animate Profile
        - Draw Profile on View - see Draw on View
        - Draw Profile on Layout - see Draw on Layout
- Tool Bar - has several tools, many of which do not need explanations
    - Zoom In
    - Zoom Out
    - Pan
    - Zoom Full Extent
    - Fixed Zoom In
    - Fixed Zoom Out
    - Identify - displays for the user clicked location on the profile Z, Station and Slope for the Identity layer. A point is drawn for the same location on the cross-section line in the Data Frame (the View). The Identity layer can be changed by right clicking on the layer name in the table of contents and selecting "Set Identify".
    - Measure - allows measurement of horizontal or vertical distances in the Profile Window. With the Measure tool selected press and hold the mouse button and drag the mouse cursor across the window. The measured distance is displayed in the toolbar next to the Measure tool
    - Label - Draws a label with Z value for the profiles at user clicked point.
    - Delete Label - deletes the clicked label.
    - Z scale indicator - displays the current Z scale - the difference of the representation of the distances in length and height. Z Scale = 2 for example will mean that 250 meters height will be equal to 500 meters length on the profile display. Initially the Z Scale is calculated automatically for best fit in the Profile Window. The user can adjust the value of the Z Scale at any time - see Profile Settings

# Profile Extractor

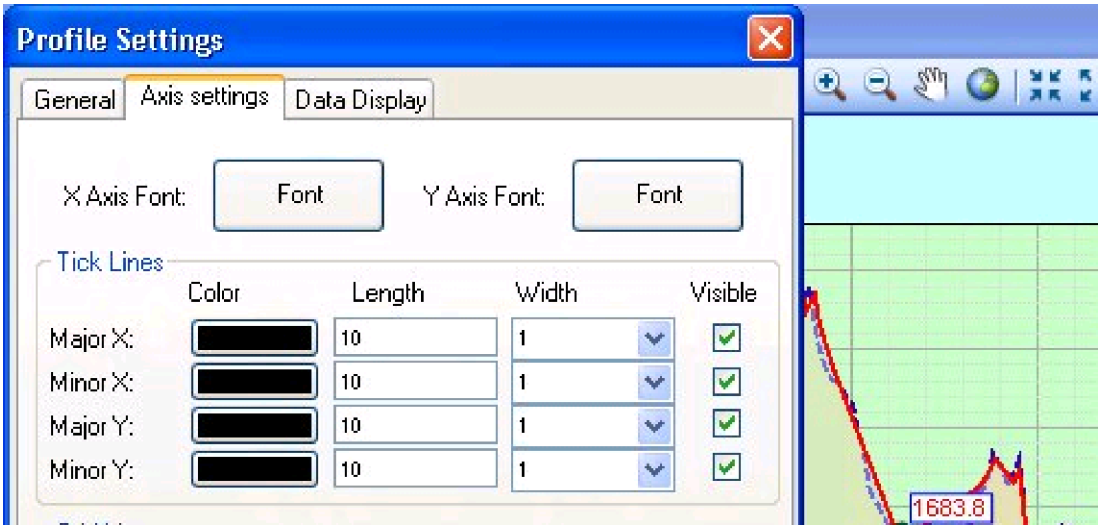
**Profile Settings**

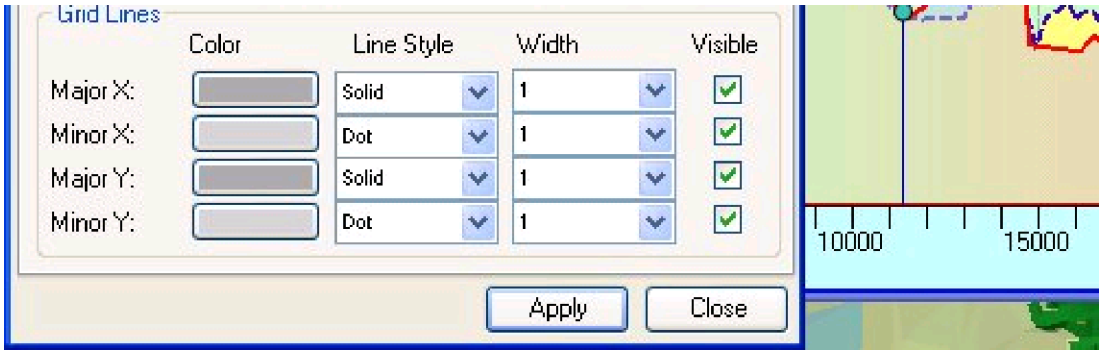
General settings for the Profile Window can be adjusted by the user from the Profile settings dialog (Settings Menu ==> Profile Settings)

- On the General tab the user can set a title for the profile, its font and colour as well as styles and colours for the background and the profile area.

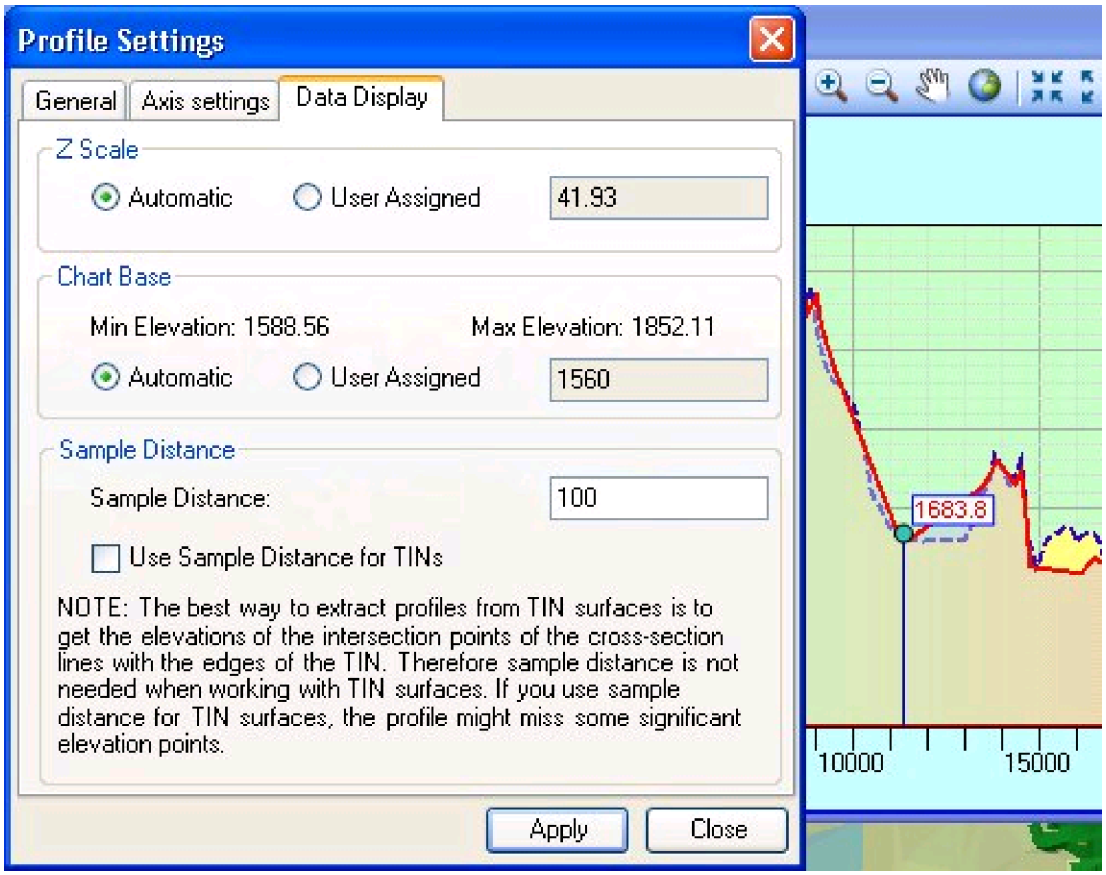



- The Axis Settings tab allows adjusting
  - the fonts and colours for the axis labels
  - the color, width, length and visibility of tick marks
  - the style, width and color of grid lines






- Data Display options
  - Adjusting Z scale (the difference of the representation of the distances in length and height. Z Scale = 2 for example will mean that 250 meters height will be equal to 500 meters length on the profile display).
    - Automatic - the software will calculate Z Scale based on the profile data and the size of the Profile Window.
    - User Assigned - the user can set explicitly the Z Scale for drawing the profile. Note that this might cause not the entire profile to be visible. The user might need to adjust the size of the Profile Window or the zoom factor.
  - Profile Base (the lowest Z value to be displayed). Initially the Profile Base is calculated by the software based on the profile data and the size of the Profile Window. The user can adjust the Profile Base explicitly. To facilitate this the highest and lowest Z values of the current profile are indicated in the dialog.
  - Sample Distance - the distance (in the units of the spatial reference of the data frame) along the cross-section polyline data points are interpolated from the surface. The sample distance can be adjusted from the Data Display tab of the Profile Settings or directly from Settings ==> Sample Distance. The Profile Settings Dialog allows the user to indicate whether the Sample Distance will be used for profiles derived from TIN surfaces. See the discussion about Sample Distance.

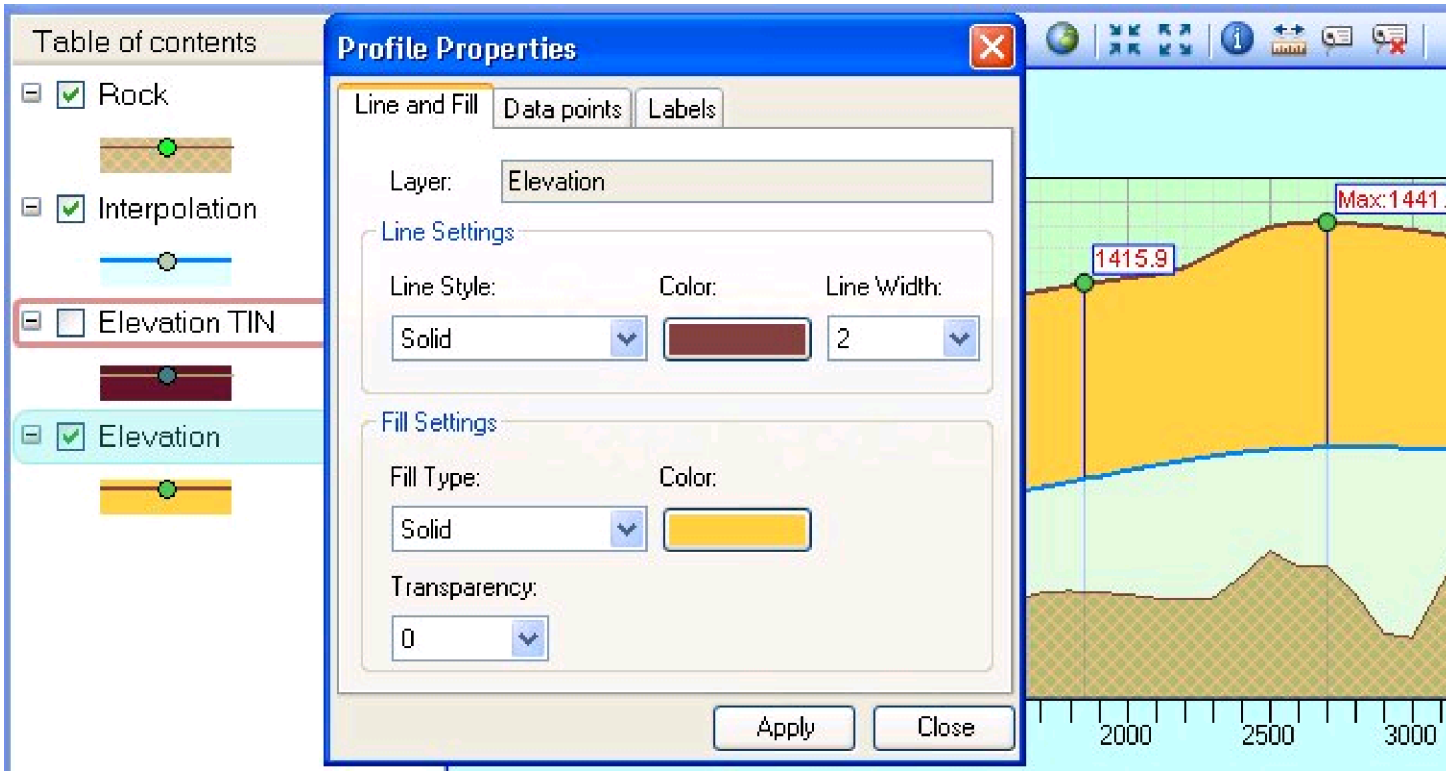


# Profile Extractor

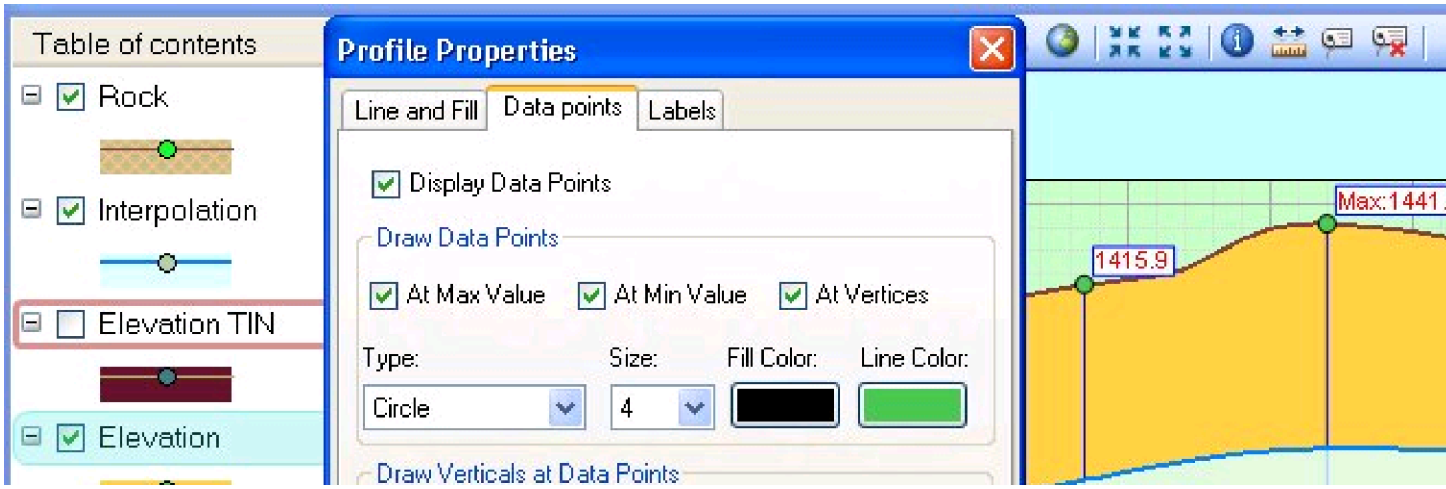
**Profile Properties**

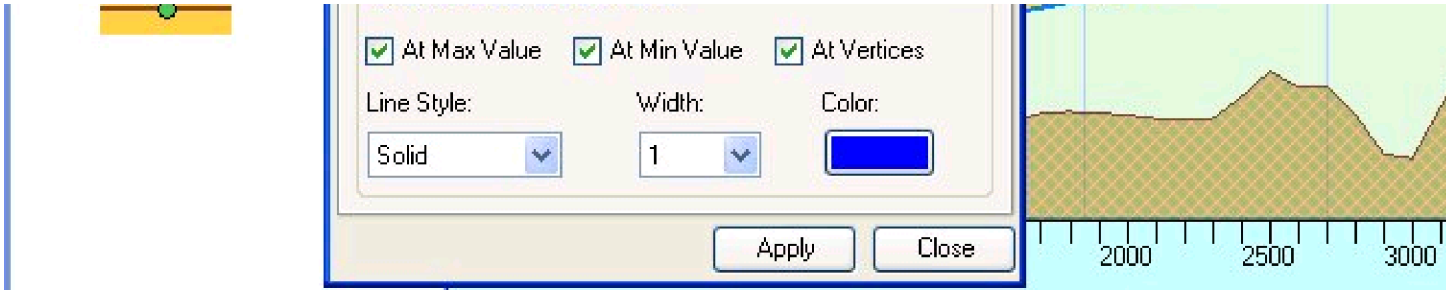
Property settings for each layer can be adjusted in the Property Windows (TOC Context Menu ==> Properties). Depending on the layer type, different Property Windows. When the layer is a surface, the Profile Properties Window is displayed. For layers - see Layer Properties

- Profile Properties - apply settings for a Surface layer
  - Line and Fill tab - allows for changes in the line style, color and width and the fill type, color(s), style and transparency.

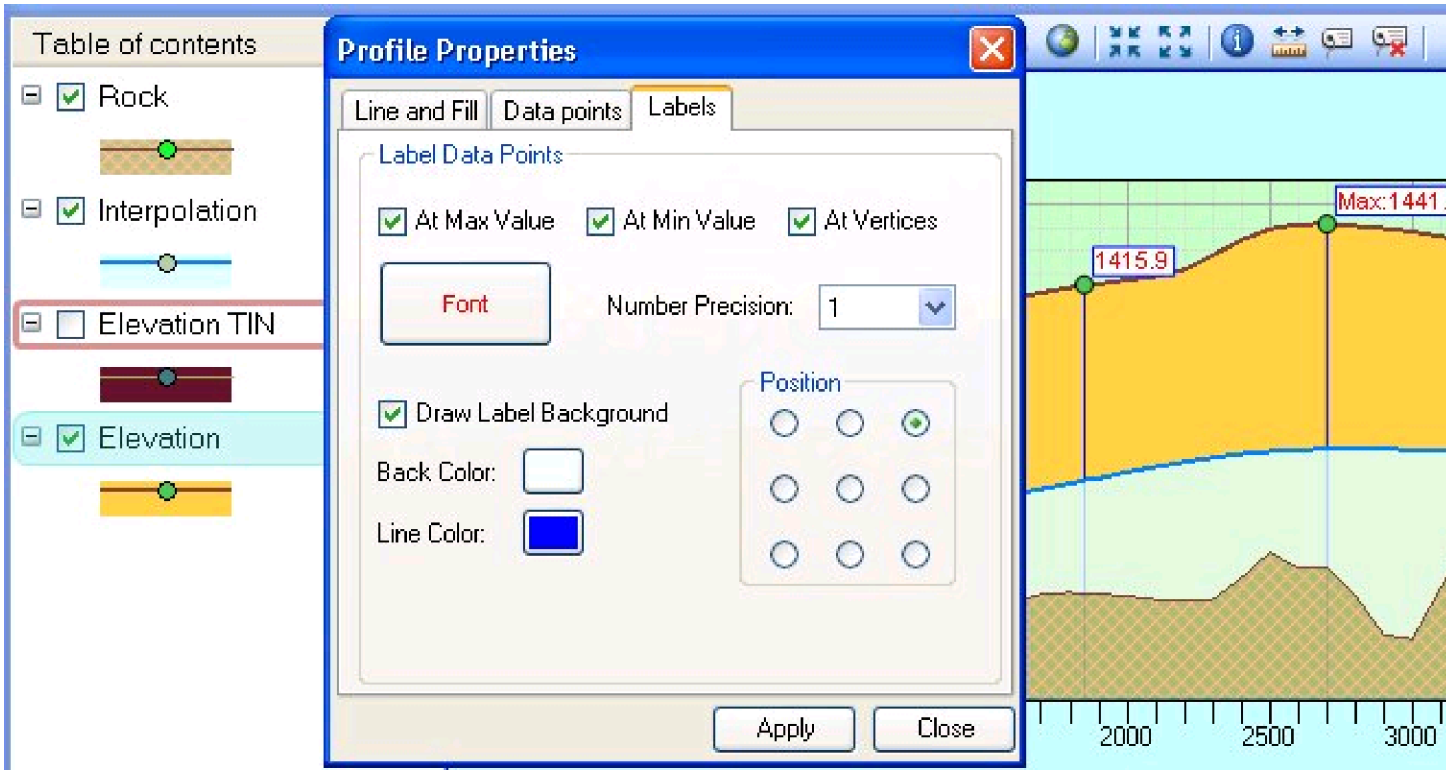



  - Data points tab
    - Display Data Points - when not checked, no Data Points are displayed including Verticals and Labels. When checke below are used for the Data Points display. This option can also be changed for a layer by right clicking on the Laye TOC and checking "Show Data Points".
    - Draw Data Points - the user can specify which significant data points will be drawn automatically on the profile and c symbol, size and colors.
    - Draw Verticals at Data Points - the user can specify at which significant data points a vertical lines will be drawn aut profile and change their line style, width and color.






- Labels tab
  - Label Data Points - the user can specify which significant data points will be labelled on the profile, change the font, and position of the Labels



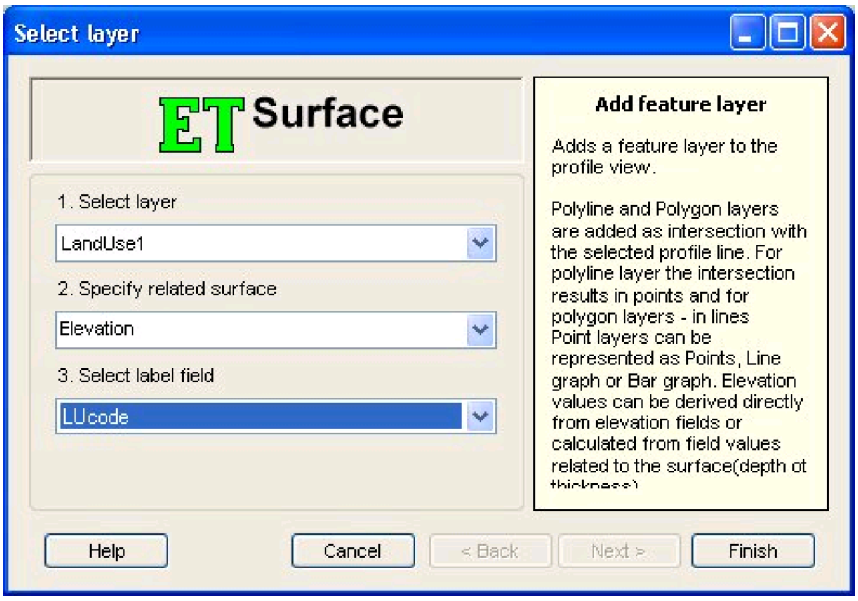
# Profile Extractor

**Add Layers to Profile**

Feature layers loaded in the Map can be added to the Profile window and draped on a profile by selecting File Menu ==> Add Layer. Depending on the type of layer to be added, the following is displayed in the Profile Window:

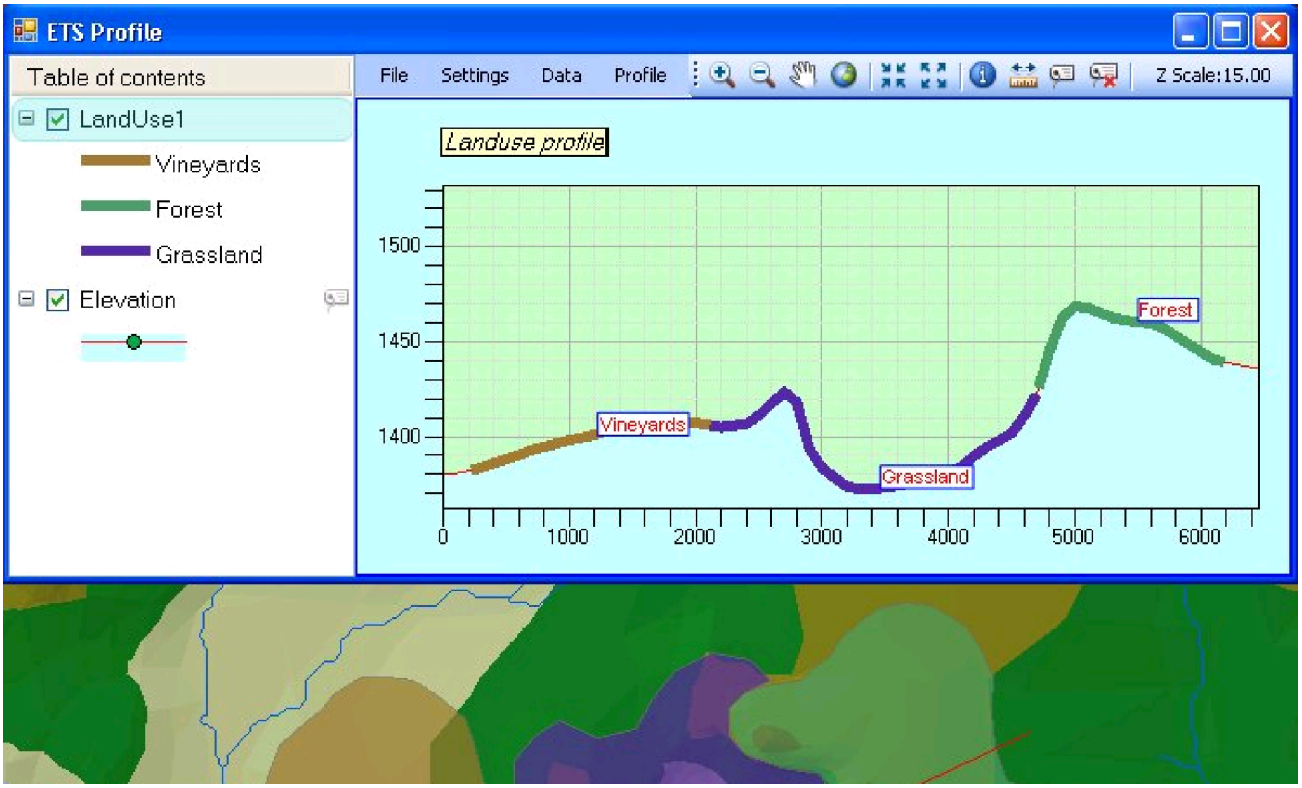
- If the selected layer is of Polygon type, an intersection of the Profile polyline with the Polygon layer is derived and draped on the related surface. The intersection results in line(s) and they are displayed on top of the related surface.
- If the selected layer is of Polyline type, an intersection of the Profile line with the Polyline layer is derived resulting in points - all the intersections of the Profile polyline with polylines from the selected layer. These points are displayed on top of the related surface.
- If the selected layer is of Point type, the user has to specify a buffer distance. All points within this distance from the Profile polyline are selected and draped on the related surface. Points can be represented on the Profile window as Point graph, Line graph or Bar graph - see detail further down this page.

Selecting File Menu ==> Add Layer open the Select Layer dialog.

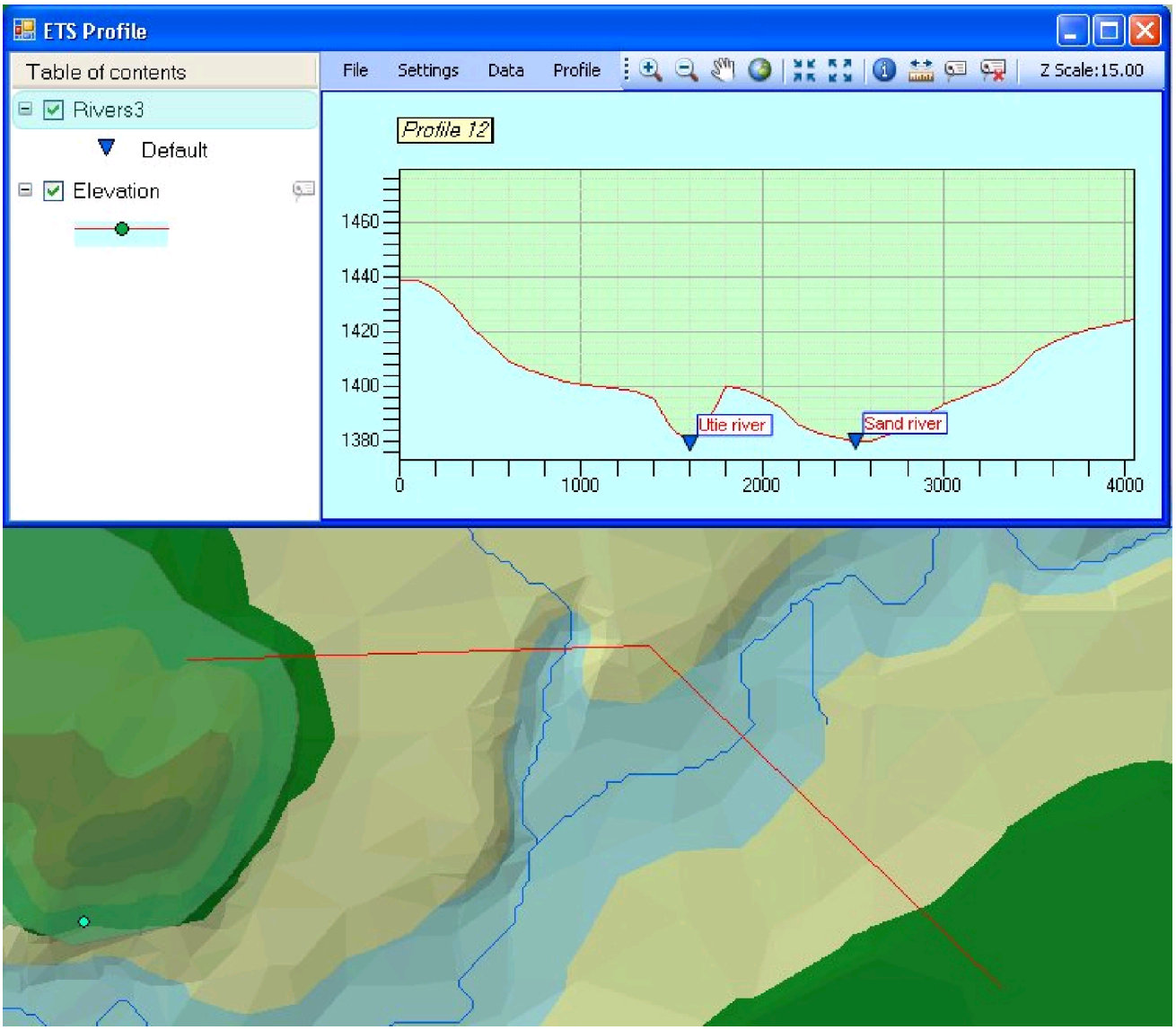



If the selected layer is of type Polyline or Polygon, the user has to specify the related Surface on which the layer will be draped and a label field, the values from which will be used for labeling the features.

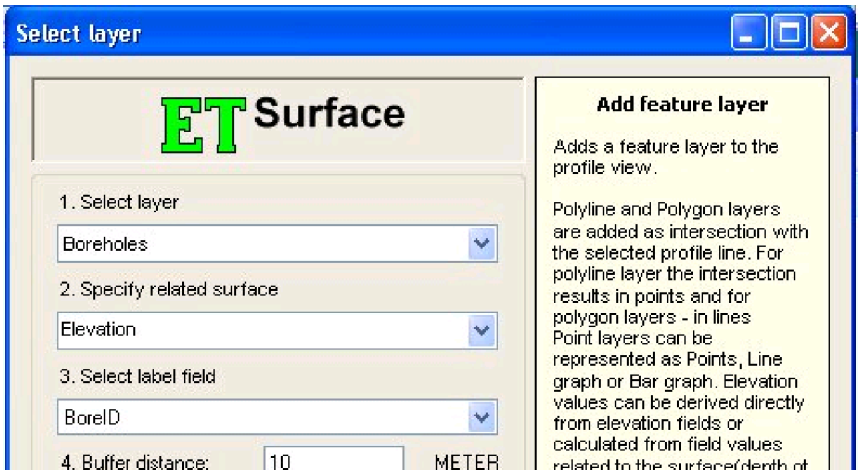
A sample display for a Polygon layer. To change the display, right click on the layer in the TOC and select "Properties" - see Layer Properties.

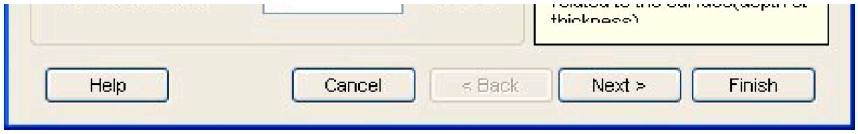


A sample display for a Polyline layer.  To change the display, right click on the layer in the TOC and select "Properties" - see Layer Properties.



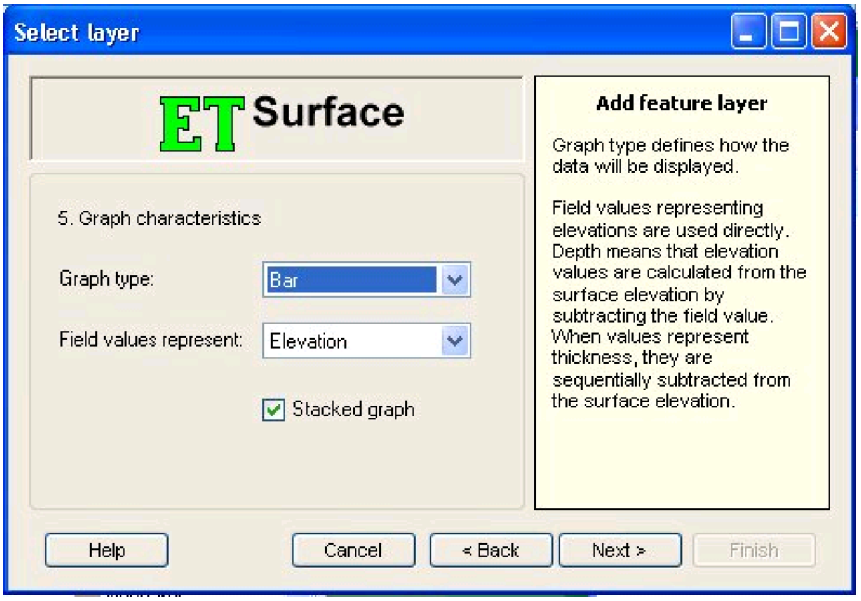

If the selected layer is of type Point the user has to define additionally the Buffer distance which will be used for the point selection.






By selecting "Finish" at this stage, all points within the buffer area of the Profile Polyline will be selected and displayed with their surface value in the Profile window.

By selecting "Next" the user can set additional properties for the display

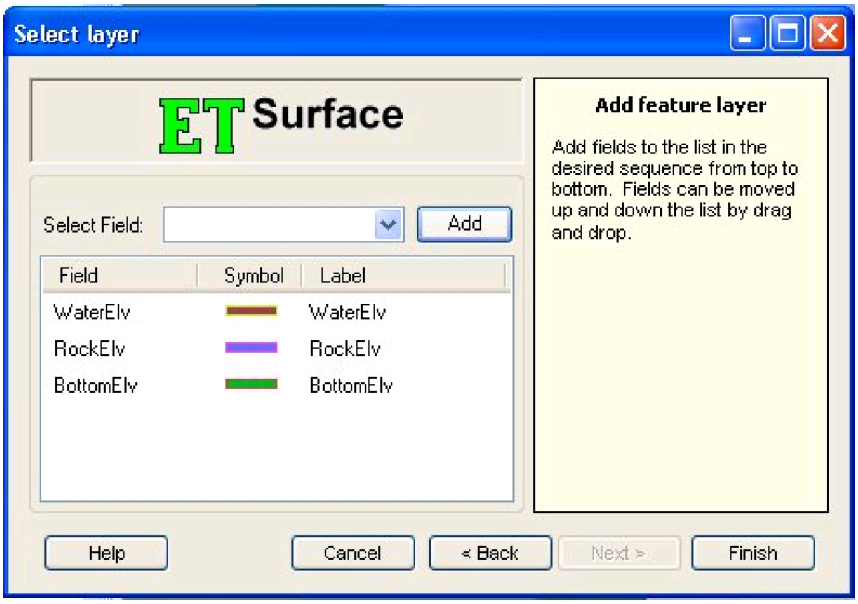



The Graph type can be Point, Line or Bar. In order to display Line or Bar graphs, additional values derived from fields need to be specified.

The field values can represent Elevation, Depth or Thickness and the calculation of the resulting heights are based on this selection. Elevation values are derived directly from the field. Depth values are subtracted from the Surface value at the point to calculate the elevation. Thickness values are used to sequentially subtract the thickness from the previous elevation value.

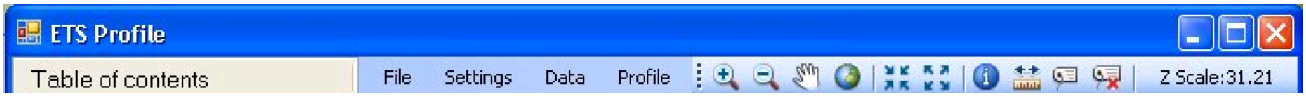
If the field values represent Elevation or Depth, the user can specify that the graph is stacked - meaning all height values are calculated and sorted and drawn sequentially from top to bottom. If the "Stacked graph" option is not checked, each line or bar starts from the Surface value to the calculated elevation.

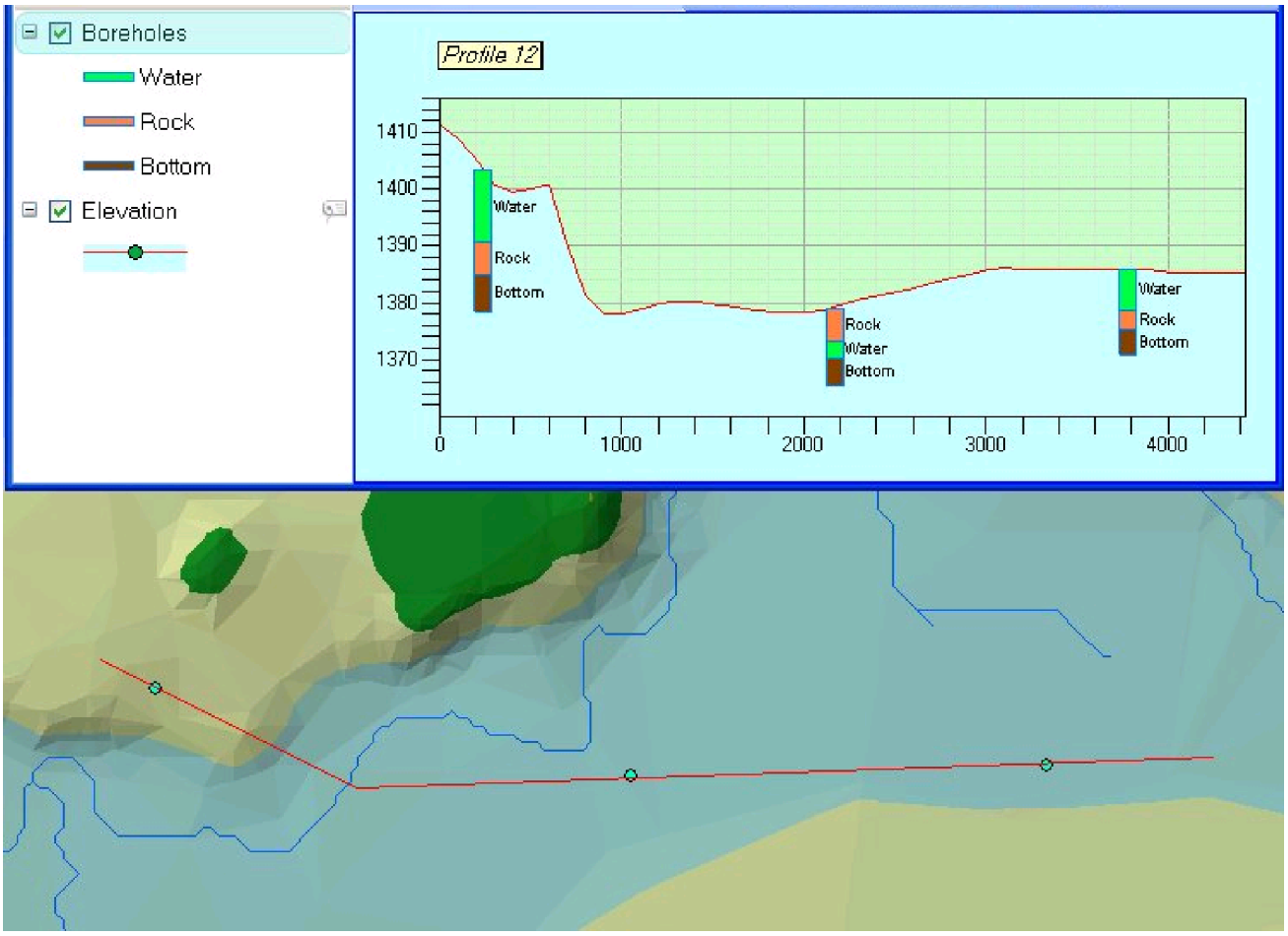
By selecting "Next" the user goes to the next step - selection of fields.




Select and add all field from which values for the graph will be derived. The Label column can be edited to change the label, which will be displayed. The table rows can be moved up and down to change the sequence of calculating elevations. Please note that in some cases the sequence does not have an influence on the display - for example if the selected Graph type is Point or if the Stacked option is selected.

Sample display for Point layer. To change the display, right click on the layer in the TOC and select "Properties" - see Layer Properties.





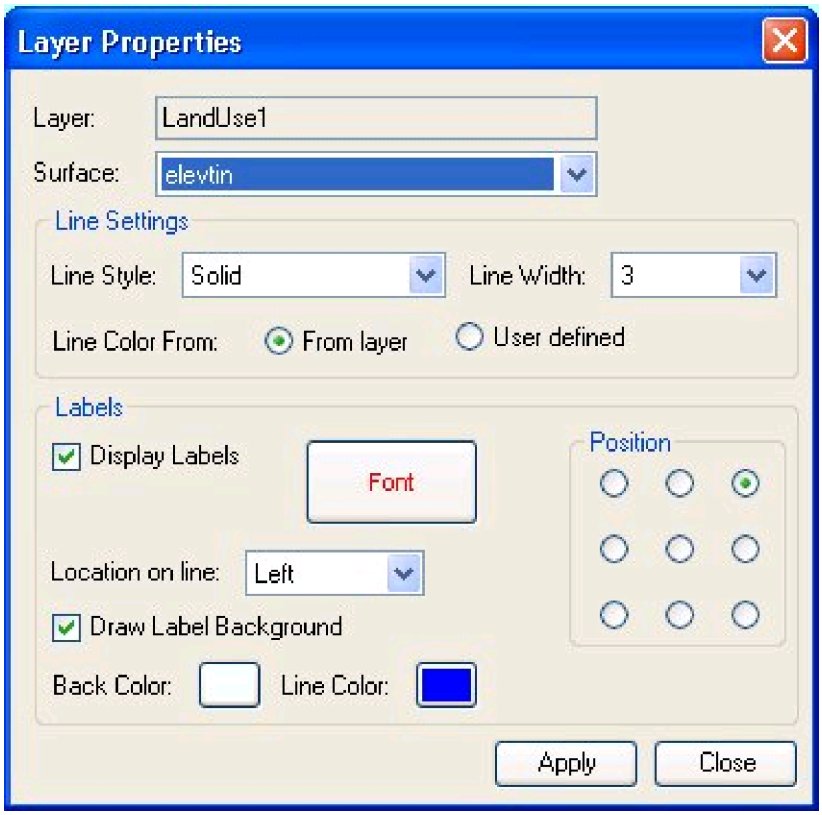
# Profile Extractor

**Layer Properties**

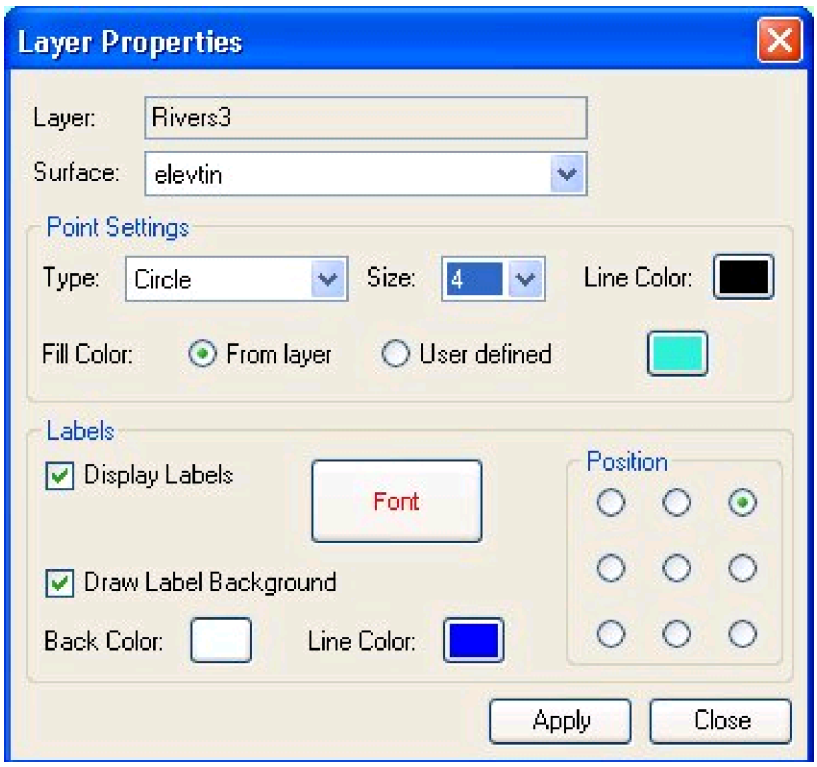
Changing the display options for a layer can be done from the layer Properties window (right click on the Layer in the TOC and select "Properties" from the context menu. Click "Apply" to apply the changed settings. Clicking "Close" only closes the Properties window.

Depending on the type of layer, different Properties window will be displayed.

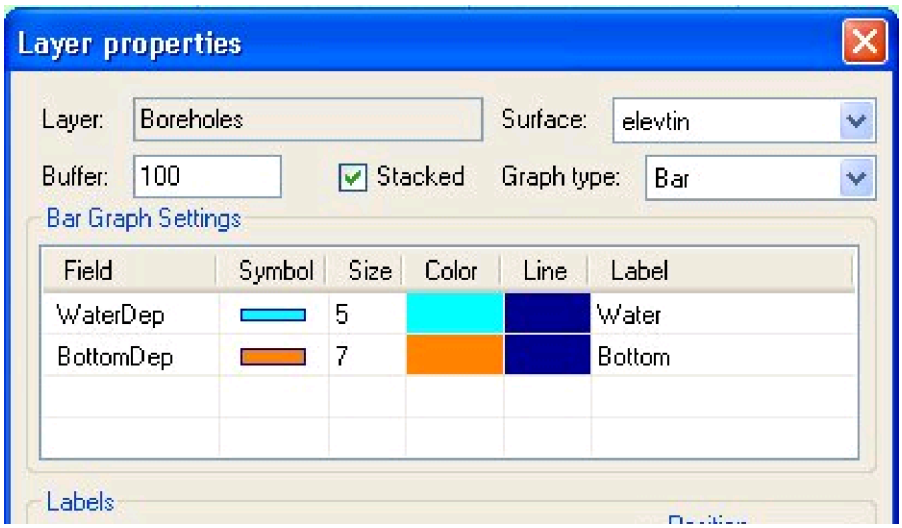
- Polygon layer - displayed when the selected layer is of Polygon type and so the representation in the Profile Window is lines draped on the Profile line. The user can adjust the following:
  - The related surface on which the layer will be draped.
  - The display of the derived lines by changing their line style and width
  - Line Color From option. When "From layer" is specified, the line color of the draped polygons is taken from the Layer polygon symbols on the Map (polygon fill color). This honours Category (only by single field) or Quantity classification that has been applied. When "User defined" is selected, only one color is applied to all lines and is selected by the user in the Properties form.
  - Labels - the user can specify whether labels are displayed, their location along the line (left, centre or right), their font, color, style, background and position.

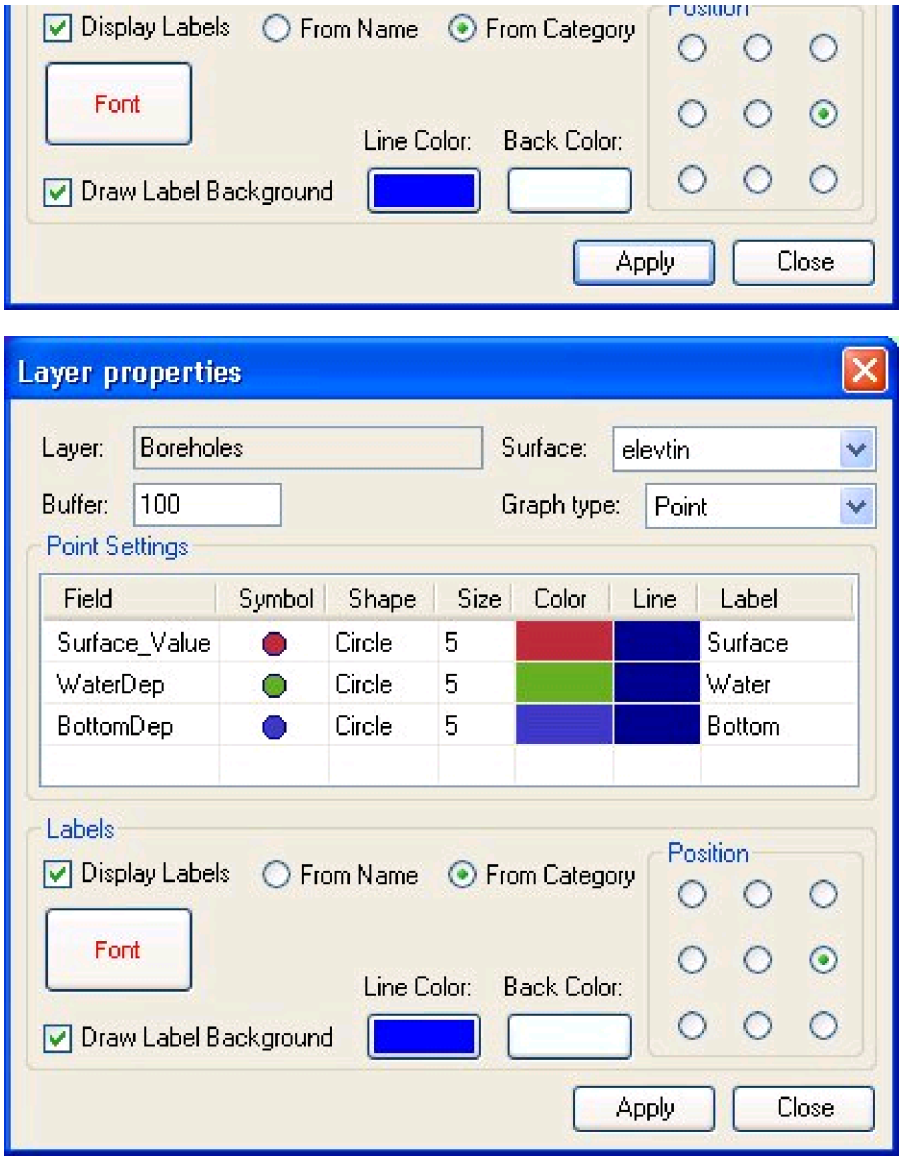



- Polyline layer - displayed when the selected layer is of Polyline type and so the representation in the Profile Window is points draped on the Profile line. The use can adjust:
  - The related surface on which the layer will be draped.
  - Point Settings - the user can change the type, size and line color of the draped point. The "Fill Color" has two options. When "From layer" is specified, the fill color of the symbols is taken from the Layer line symbols on the Map (line color). This honours Category (only by single field) or Quantity classification that has been applied. When "User defined" is selected, only one color is applied to all symbols and is selected by the user in the Properties form.
  - Labels - the user can specify whether labels are displayed, their location along the line (left, centre or right), their font, color, style and position.




- Point layer - displayed when the selected layer is of Point type and so the representation in the Profile Window can be Points, Line graph or Bar graph. The use can adjust:
  - The related surface on which the layer will be draped. Please note that for Line and Bar graphs, the Surface Value at the point is always taken into account. Non stacked graphs are displayed from the surface elevation to the calculated field elevation and stacked graphs include the surface elevation in the sorted elevation list. For Point graphs the surface elevation can be included or excluded by the user.
  - The buffer distance for selection of points
  - The Stacked option for the Line or Bar chart. The Stacked option has no meaning when the graph is based on Thickness values.
  - The graph type - Point, Line or Bar. Please note that when the Surface Value is included in the list for Point graph, it will not exist in the list when the type is changed to Line or Bar graph. Still the Surface Value is included in the graph display as described above.
  - The size, color, symbol and label for each graph category. Graph categories can be moved up and down in the table by drag and drop - the topmost category is displayed on top of the lower categories.
  - Graph categories can be moved up and down in the table by drag and drop - the topmost category is displayed on top of the lower categories. By specifying a size of 0, the category will not be displayed.
  - The display of labels with adjustment of their font, color, background and location.
  - The label for each graph can be taken from the Name field or from the Category Label





# Profile Extractor

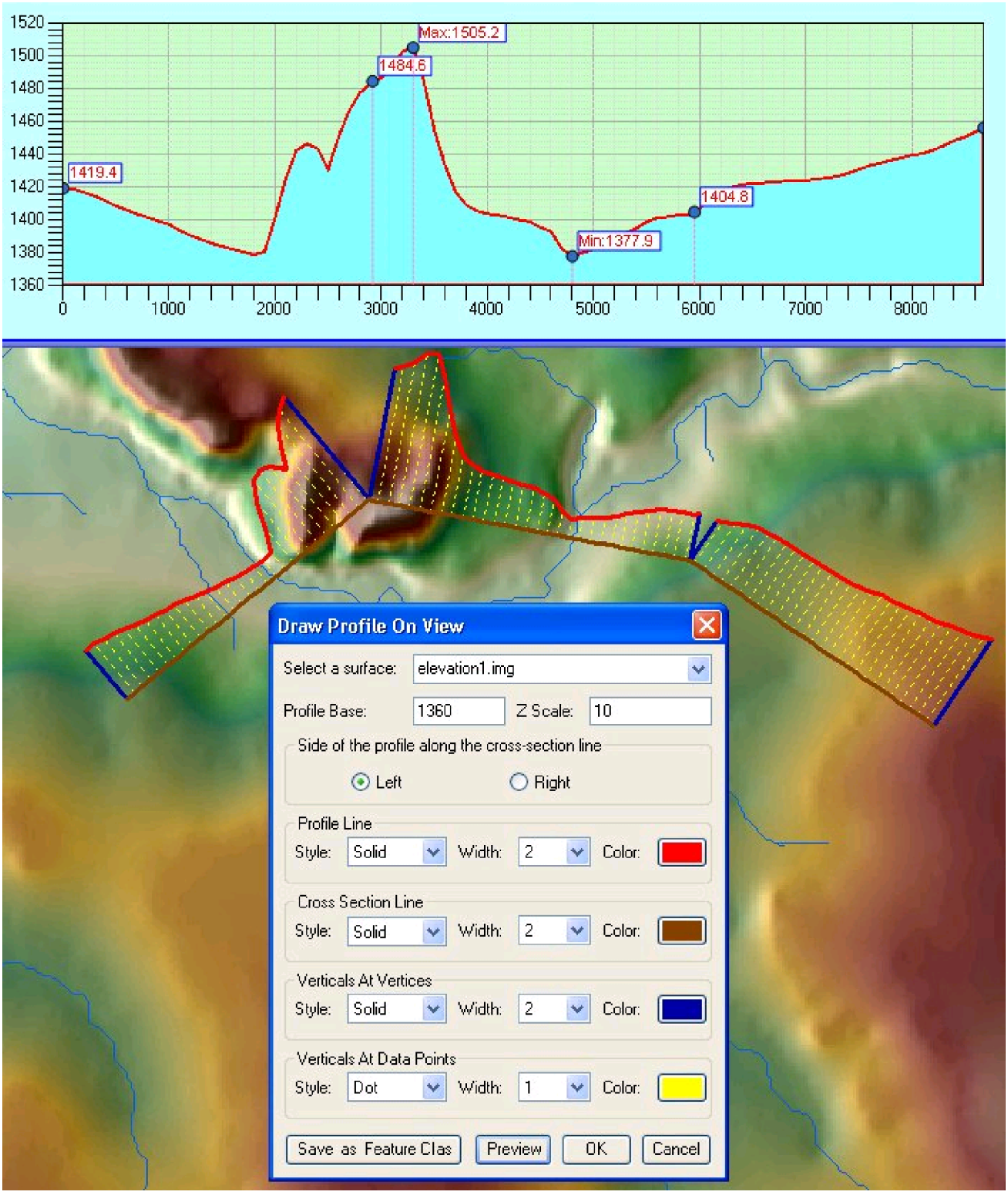
**Draw Profile on View**

The profile currently displayed in the Profile Window can be drawn on the Data Frame (the View). This function can be accessed from Profile Menu ==> Draw Profile on View. A dialog will be presented to the user to adjust the drawing options. The user can change the Profile Base and the  Z Scale. Depending on the cross-section polyline the user can select the side of the polyline on which the profile will be drawn.

Only s single surface profile can be drawn on the view. The user needs to select which surface layer is to be used.

The Preview button draws the profile elements as temporary graphics. If the user selects Cancel the graphics will be deleted. If the user clicks OK the graphics will be grouped in a single group.

The graphics can also be saved in a new feature class by selecting the "Save as Feature Class" button.  This will open another dialog, in which the user can specify which elements should be saved and provide a name for the new Feature Class.



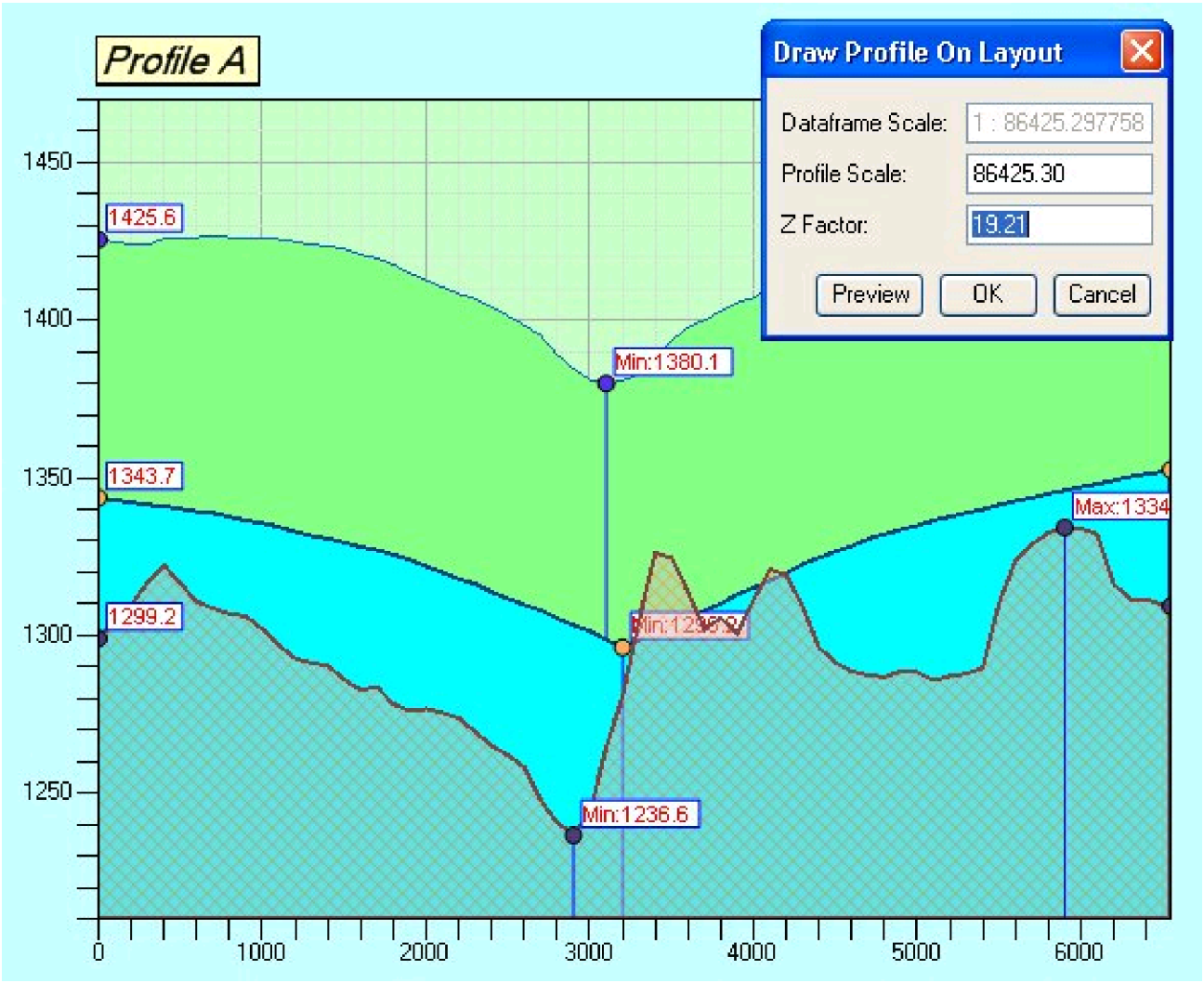

NOTE:  Line styles are only valid if the line width is 1.  For line width bigger than 1 a solid line will be drawn.

# Profile Extractor

**Draw Profile on Layout**

The profile currently displayed in the Profile Window can be drawn on the layout. This function can be accessed from Profile Menu ==> Draw Profile on Layout. A dialog will be presented to the user:

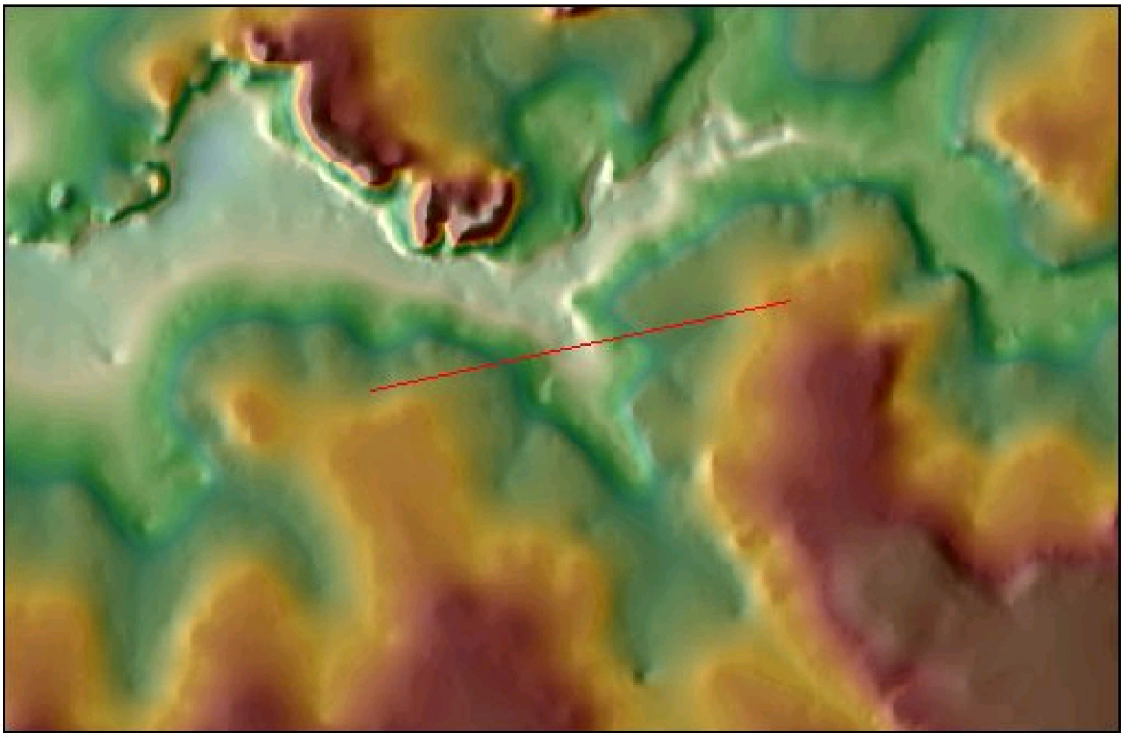
- Current scale of the data frame will be indicated in the dialog
- The user can change the scale for drawing the profile or draw the profile with the scale of the data frame
- The current Z Scale will be shown and can be adjusted.

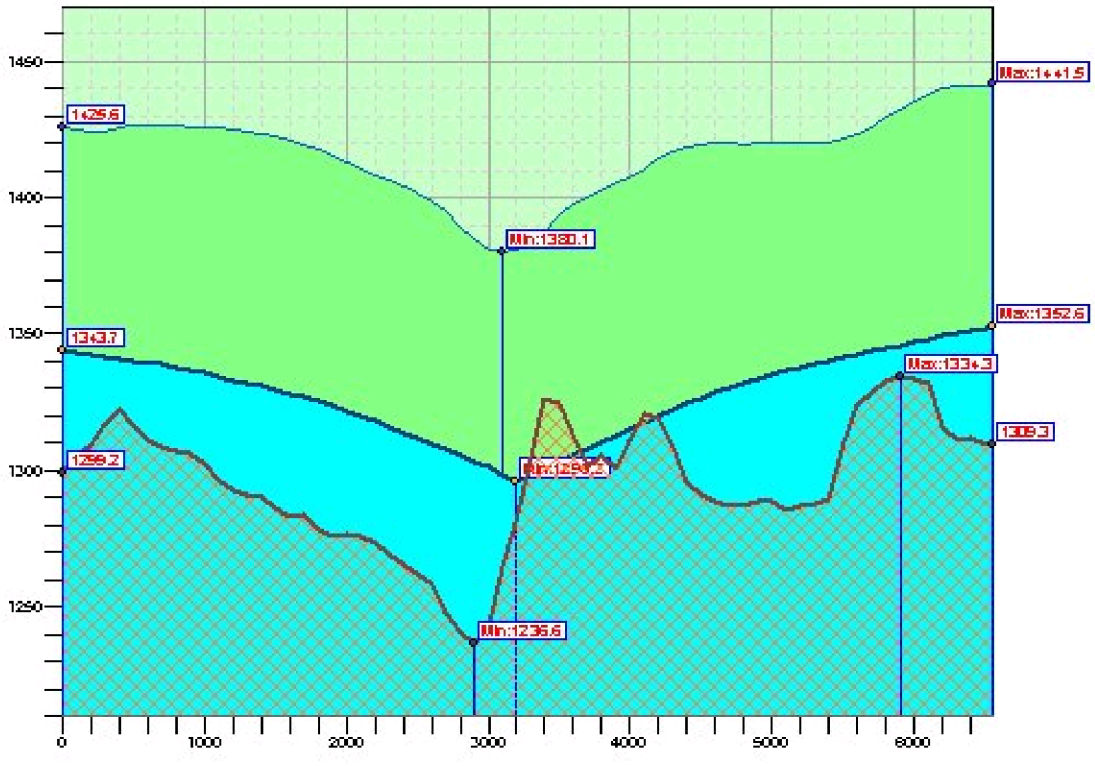



The Preview button draws the profile elements on the layout as temporary graphics. If the user selects Cancel the graphics will be deleted. If the user clicks OK the graphics will be grouped in a single group. Then the user can move it to a position of choice.

- The profile will be drawn to the user defined scale.
- All the symbology from the Profile Window will be preserved in the profile drawn on the layout with the following exceptions:
  - Profile transparency is not supported
  - Point types x, y and z are not supported - these will be drawn as circles

- Line types are only supported, when the line width is 1, if the line type is not Solid and line width is more than 1, the lines will be drawn as solid lines





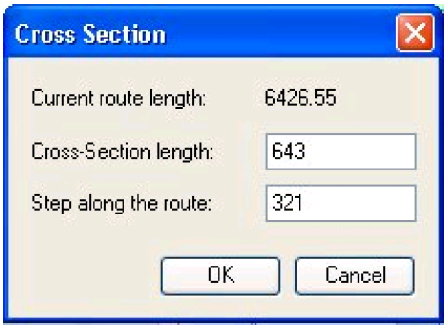
# Profile Extractor

**Draw profile for cross-section lines moving along a route.**

- Use the Draw Cross-Section line tool  , Draw profile for feature 

tool or Draw profile for graphic 

tool to define the route along which the cross-section line will move. The profile for the route will be drawn in the Profile Window
- Adjust the order of the surfaces and the symbology of the profiles.
- Click on Profile Menu == > Profile Cross-sections Along Route - a small dialog will be displayed.

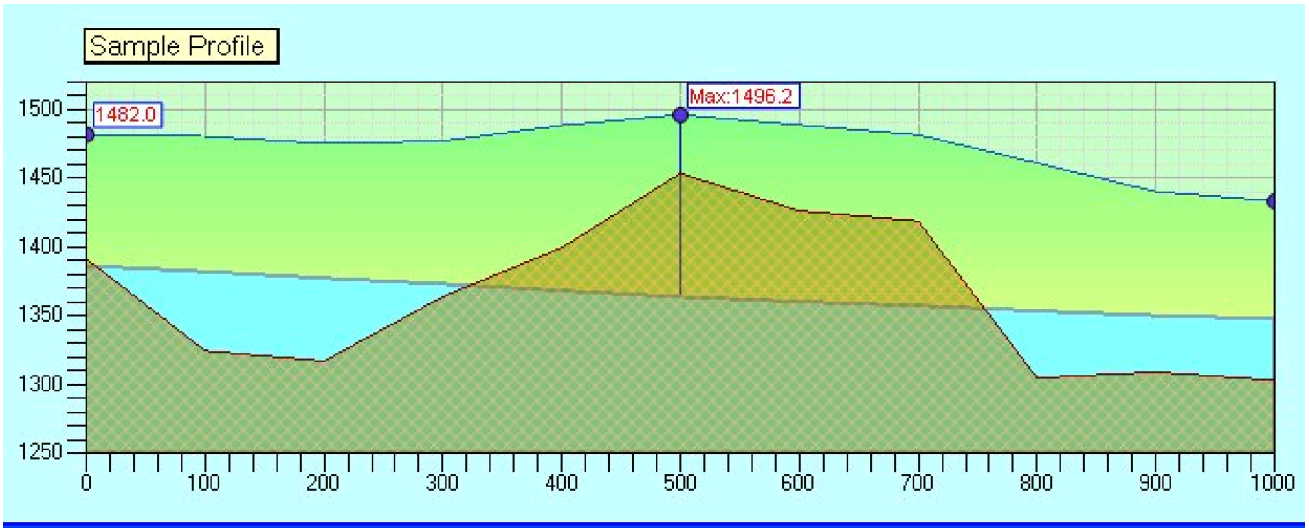


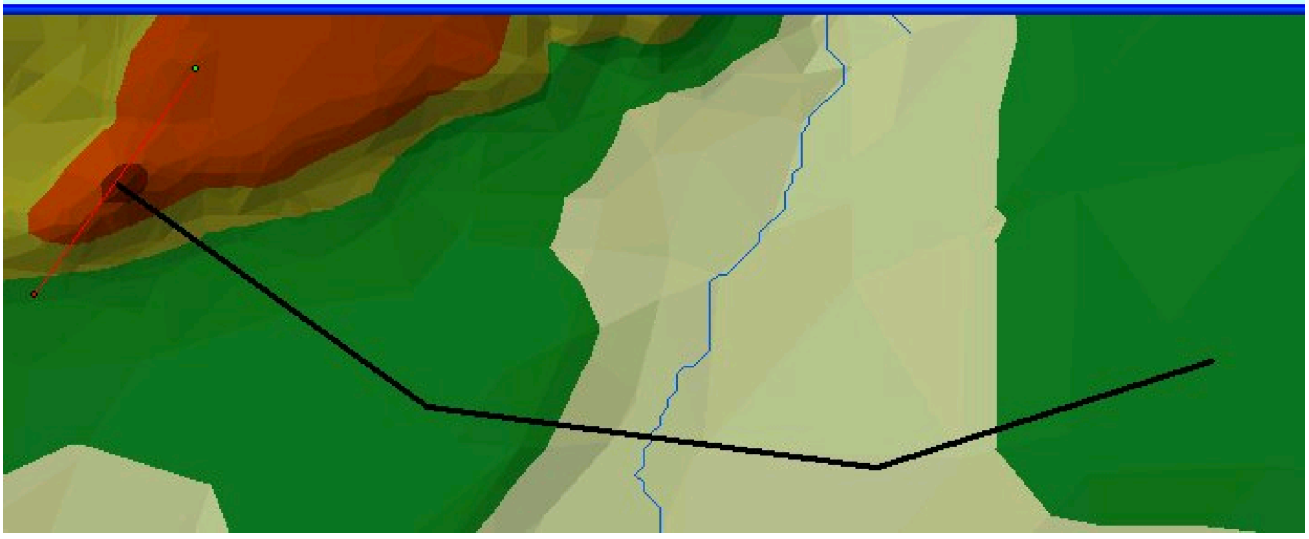

On the dialog the total length of the route will be indicated.

Specify the desired length of the cross-section line that will move along the route and will be perpendicular to the route.

Specify the step with which the cross-section line will move along the route.

- On clicking OK a new toolbar  will be displayed at the top of the Profile Window next to the main toolbar, the first cross-section line will be generated and the profile will be extracted for it.



- Using the command buttons navigate the cross-section line
- On clicking the Stop button will close the animation mode and the profile of the route will be drawn in the Profile Window.
- During the animation it is possible that the extent of the Profile view will change and go outside of the extent of the window. The Recalculate Extent button  can be used to recalculate the extent of the current Profile view and zoom to it.

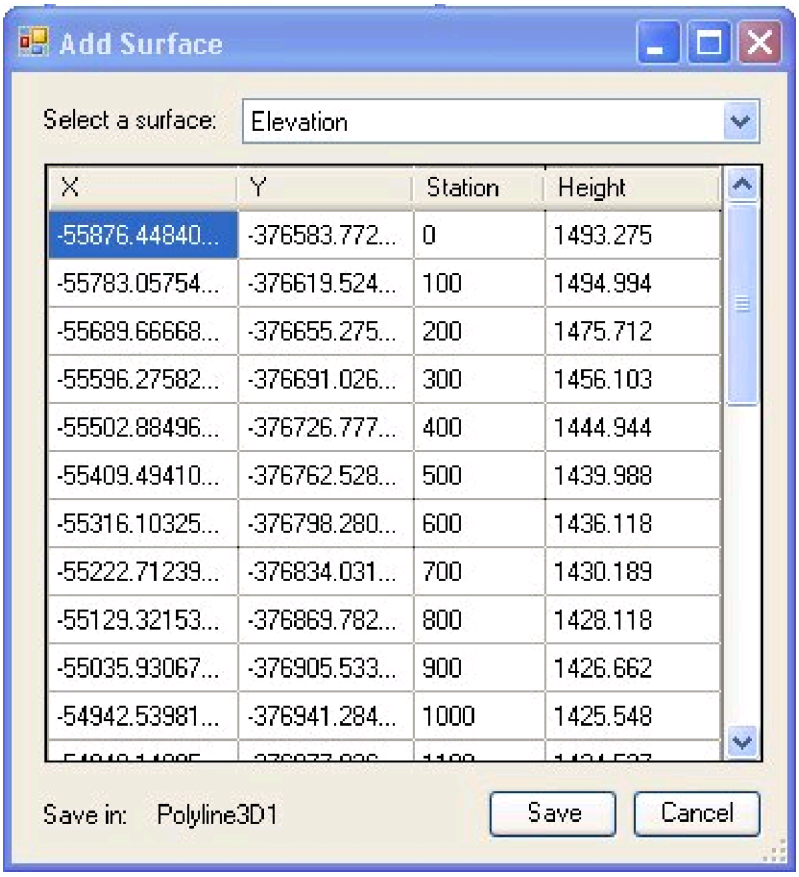
See the Profile Extractor online tutorial for a video about this functionality.

# Profile Extractor

**Draw profile for cross-section lines moving along a route.**

Several options are available for exporting the profile data:

- Save data in the currently edited layer. The layer can be PointZ or PolylineZ feature layer

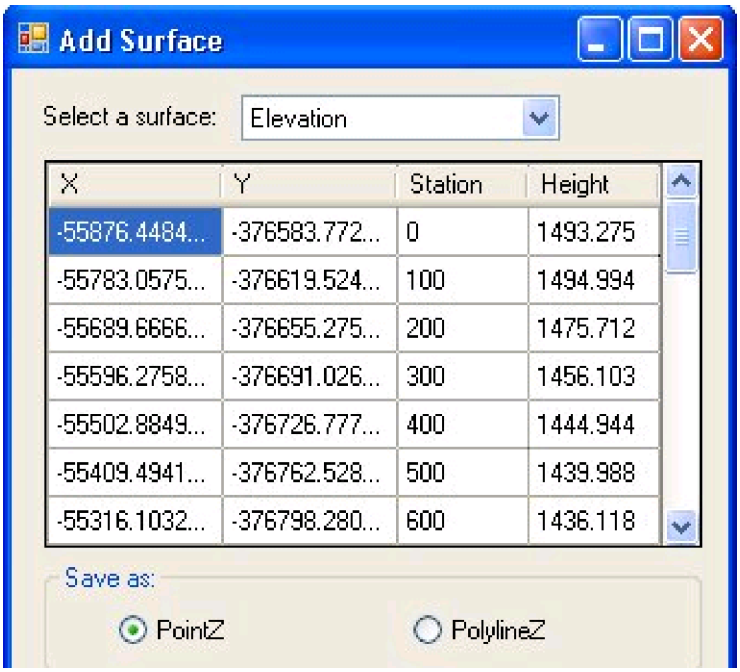



The data for a single surface can be exported only

The user needs to select a surface for which the data will be exported.

If the currently edited layer is of PointZ type and has a field called ET_Station, the stations of the data points along the profile will be stored in this field.

- Export the data of the current profile to a new feature class.

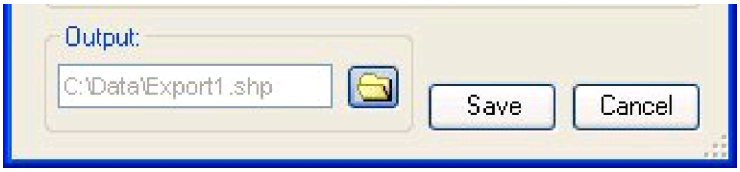



The data can be exported to a new PointZ or PolylineZ feature class.

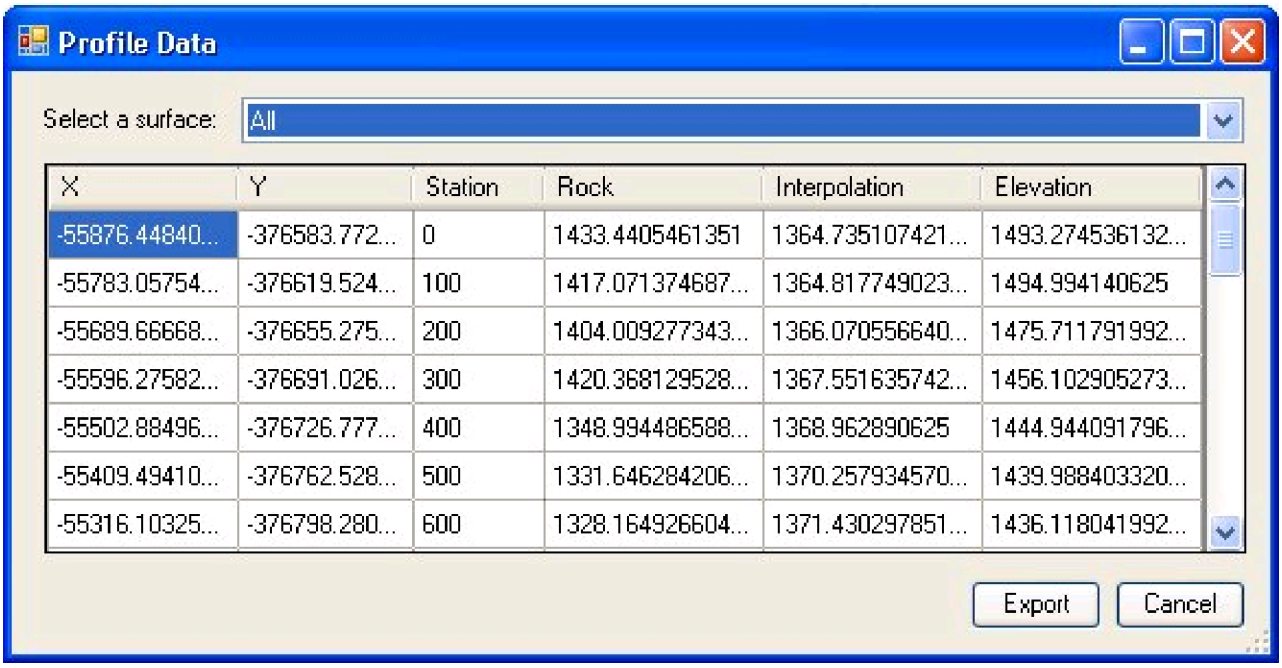
The data for a single surface can be exported only

The user needs to select a surface for which the data will be exported.

If the output is set to PointZ type a new field called ET_Station will be created, the stations of the data points along the profile will be stored in this field.




- Export profile data as a text file



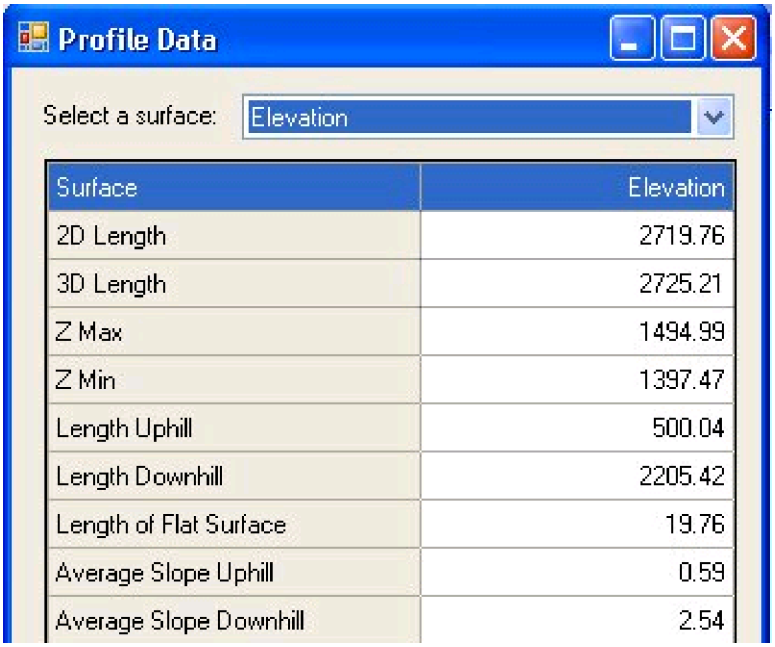

The user needs to select a surface for which the data will be exported. If the user selects "All", then the data for all surfaces in the Profile Window will be exported. Since profiles of different types may have different data points, the data points are merged and sorted, and Z values for each surface calculated for the resultant list of data points.

The data will be exported in a comma delimited text file and will have the following structure

X,Y,Station,Surface1,Surface2,Surface3.....

where Surface1 will have the Z values for the point extracted from Surface1

- Export Profile Statistics




The statistics of the profile for the specified surface will be saved to a text file.


- Save As Image - an image of the current profile will be saved to a disk.  Supported export formats are Bitmap(.bmp), Gif(.gif), Jpeg(.jpg), Png(.png) and Tiff(.tif).

# Profile Extractor

**Sample Distance - Discussion**

Profile Extractor can use various type of surfaces - ESRI TIN, Raster or PolygonZ TIN to generate surface profiles for the user defined cross-section line. In general ESRI TIN and PolygonZ TIN are different in the way they store the data, but their behavior is the same so we will refer to both surfaces in this topic simply as TIN.

A profile represents the Z values of a surface along a cross-section polyline using points on the polyline for which the Z will be interpolated from the surface. There are 2 groups of points:

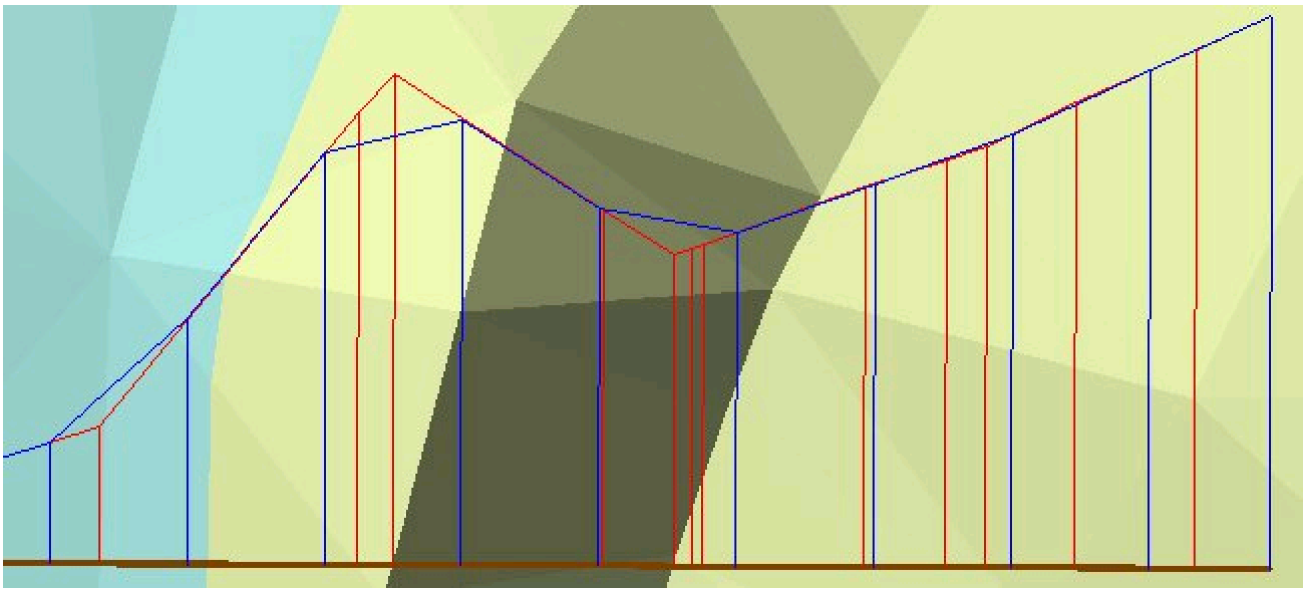
- Vertices of the cross-section polyline - the Z values of the vertices are always interpolated
- Additional data points between the vertices - these are defined by user using the sample distance assigned.

Sample Distance is a regular interval (in the units of the spatial reference of the data frame) at which data points are interpolated from the surface.

When extracting profiles from a Raster surface we need to specify sample distance in order to define the number of data points to be interpolated. The default sample distance used by the Profile Extractor is equal to the cell size of the raster. If more than one raster datasets are used, one of them will be honored in calculation the sample distance.

When extracting profiles from TIN surfaces, the sample distance is not important. Each triangle of a TIN is a plain and needs only 2 data points (the intersection of the cross-section polyline with the edges of the triangle) to be sufficiently represented in a profile.

With Profile Extractor, the user can use sample distance to extract profiles from TIN surfaces. This however in many cases will lead to losing some significant data points in the profile.



The image above shows 2 profiles extracted from a TIN surface for the same cross-section line. The profile in red is extracted without using sample distance. The data points are extracted for the significant points of the TIN (intersections with the edges of triangles). The profile in blue is extracted using sample distance. One can see that some significant points (peaks and valleys) are missing from the blue profile.

When do we need to use sample distance for profiles extracted from a TIN surface.

- If we want to compare profiles derived from different surfaces. In such a case the data points for the profile from each surface need to be the same.
- If we want the data points to be equally spaced along the cross-section polyline.

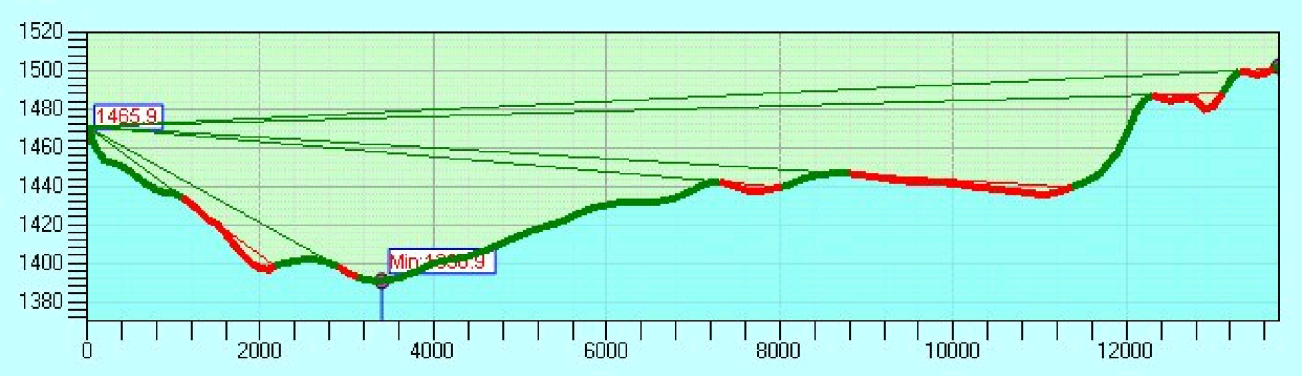
# Line of Sight (LOS)

**LOS Discussion**

Line of Sight is a direct imaginary line between two points (for example a line from the center of the eye to the center of the object viewed) uninterrupted by physical matter other than the atmosphere.

So the problem of defining LOS between 2 points can be stated simply:

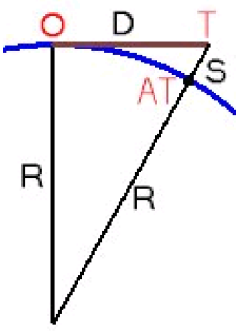
If the line between the Observer and the Target intersects the surface (buildings and other opaque objects included in the term surface here) the Target is non visible from the Observer. This can be illustrated with the following image:




The green parts of the surface will be visible and the red ones non-visible from the Observer.

Earth Curvature

There are however several other factors to consider. The above image will be correct if the Earth is flat and there is no atmosphere. We know however that the Earth is not flat and have to take into account the curvature in our calculations.




Lets assume an Observer (O) located at sea level that is looking towards a Target (AT) located also at sea level. For the calculations we will assume that the Earth is a sphere. Lets get the radius (R) of the Earth at the Observer and Target points. The radius at the Target will intersect the tangent at the Observer in point T. Lets indicate the sink of the Target due to the curvature of the Earth with S. Using Pythagorean Theorem we can easily obtain the value of the sink

$(R+S)^2 = R^2+D^2$

$R^2 + 2RS + S^2 = R^2 + D^2$

This can be solved for S as a quadratic equation:

$S^2 + 2RS - D2 = 0$ (We know R and D)

but to simplify the formula, we'll take a different approach

$S(2R+S) = D^2$

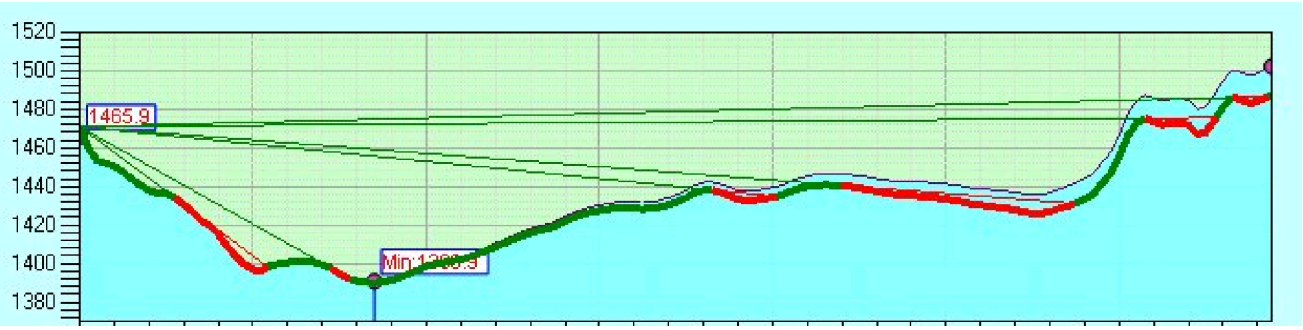
$S = D^2/(2R+S)$

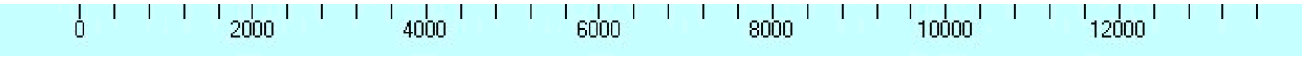
| Distance (Meters) | Sink (Meters) |
|---|---|
| 1,000 | 0.08 |
| 5,000 | 1.96 |
| 10,000 | 7.85 |
| 30,000 | 70.65 |
| 50,000 | 196.23 |

Since the radius of the earth R = 6,370,000 meters is significantly (hundreds times) larger than the sink (S) we can accept that 2R+S = 2R (this will give 1 millimeter difference compared to the exact results if calculated for D = 50,000).

Therefore our formula for the sink becomes: **$S = D^2/2R$**

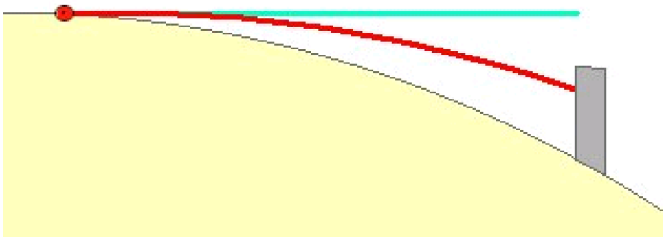
From the table on the left we can see that the sink of the target significantly increases with the increase of the distance to the target. Since some obstacles that are between the observer and the target will sink less than the target itself, in many cases we'll have a target that to become invisible






The image above illustrates how the target sinks (compared with the profile of the surface) with the increase of the distance to the Observer.

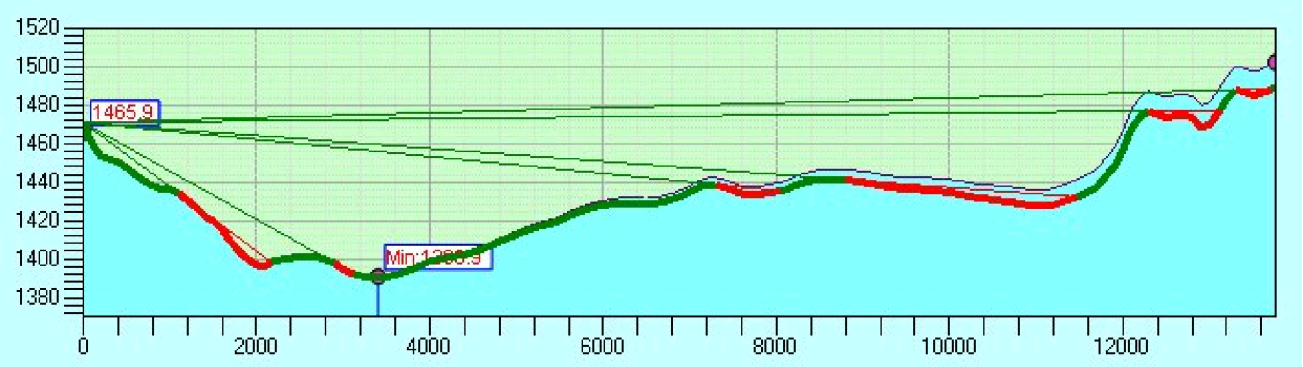
Taking into account the refraction of the light



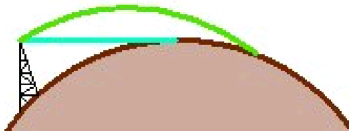
As a general atmospheric condition the density of the air decreases as height increases. As a result of this the light tends to bend as it travels long distances through air. This causes distant objects near the horizon to appear higher than they actually are. This negates to some extent the sinking caused by the curvature of the Earth. The refraction coefficient might differ for different atmospheric conditions coefficient of 0.13 can be used for achieving reliable results: **$R = KD^2/2R$**

K is the refraction coefficient

.



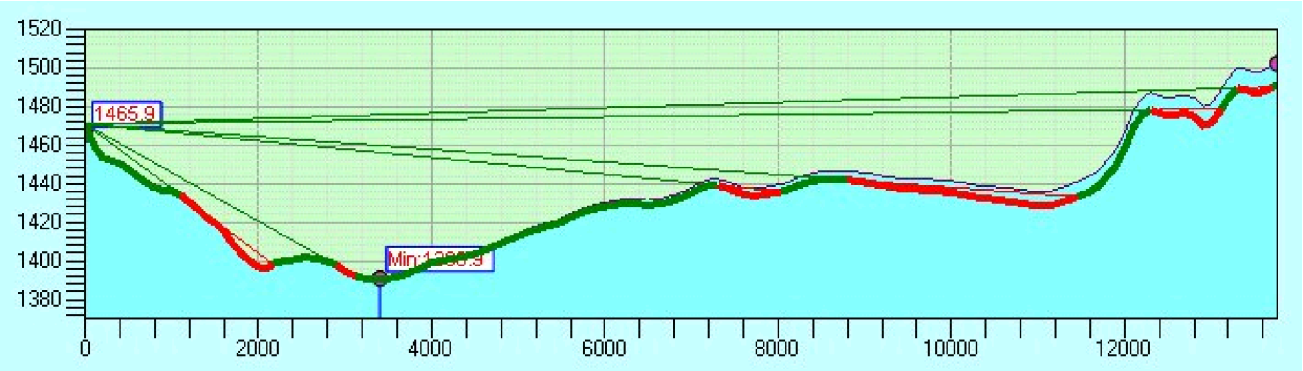

Line of sight for radio propagation.



There are many factors influencing the trajectory of the radio waves - atmospheric pressure, temperature, etc., but in general it tends to bend toward the Earth and returns to the surface behind the horizon. In practice an Effective Earth Radius Ra = KR is used to take into account the refraction of the radio waves.

For typical atmospheric conditions the correction factor (K) is 4/3.

In other words the distance from the transmitting antenna to where the ray returns to the surface is equivalent to the optical horizon, had the Earth's radius been 4/3 of its actual value.

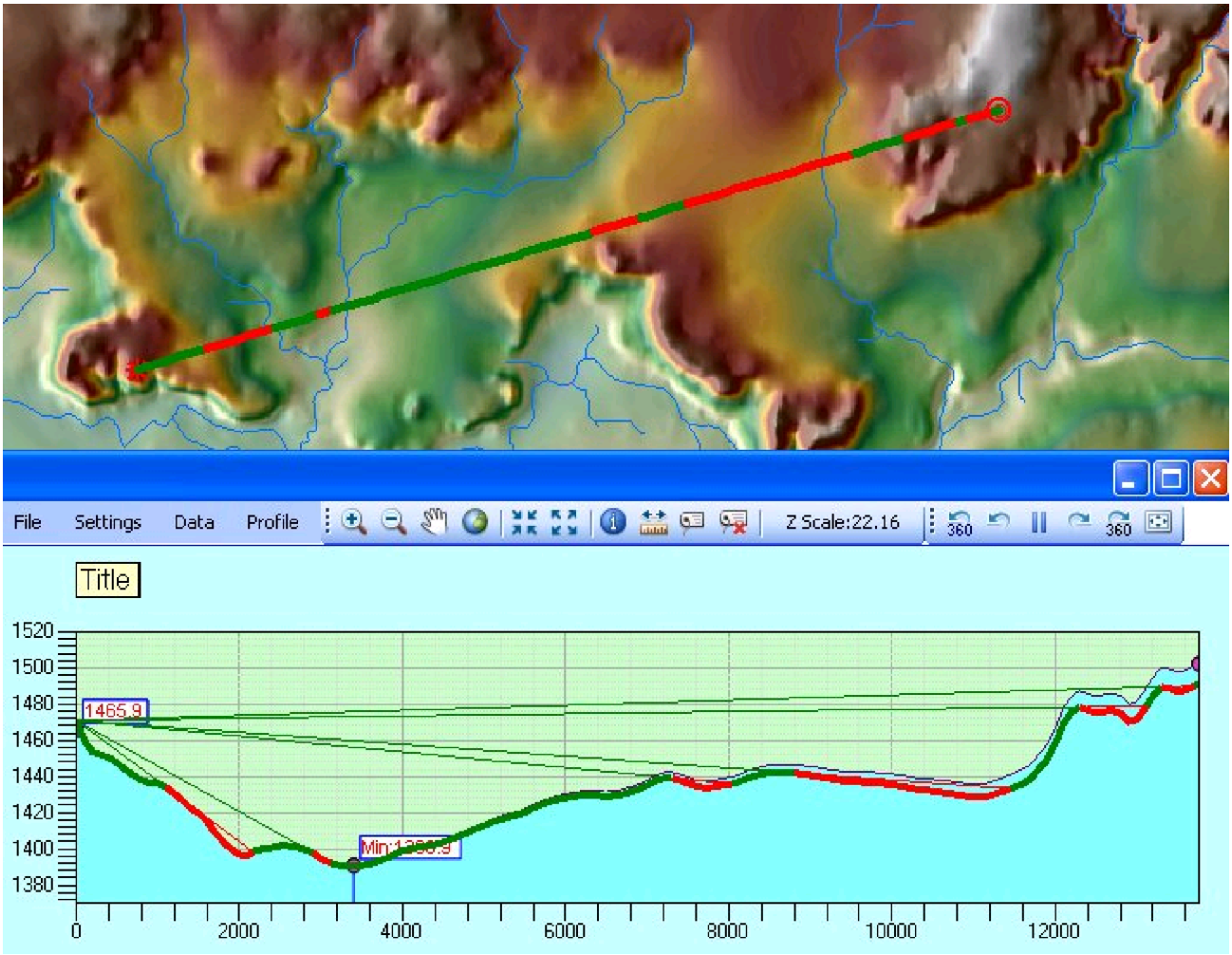


# Line of Sight (LOS)

**LOS Tools**

The Draw Visibility Line tool allows the user to draw a line defining the Observer location and the Target location. Then it calculates the line of sight between the Observer and Target from the surface selected in the Surface Layer box
With the tool the user drags a line on the View. The start point of the line defines the position of the Observer. The visibility is calculated for the data points along the line. The results are displayed:

- On the View - a group graphic with color coded visible and invisible portions of the line
- In the Profile Window - the line of sight is drawn together with the profile of the surface which gives the user better understanding of the results. The user has option to display also the Break Lines - LOS for which the visibility changes



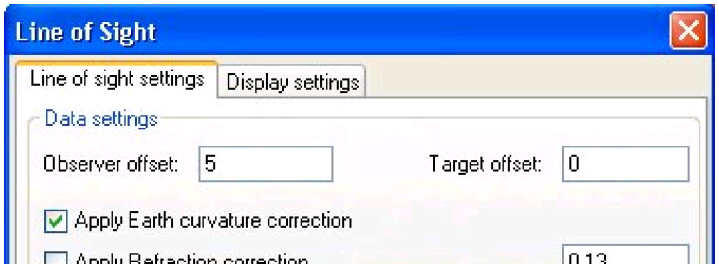

The Draw Visibility Observer tool allows the user to move the Observer location by clicking on the Map.

The Draw Visibility Target tool allows the user to move the Target location by clicking on the Map.

Whenever the Observer or Target locations are changed a new Line of sight is calculated and displayed.

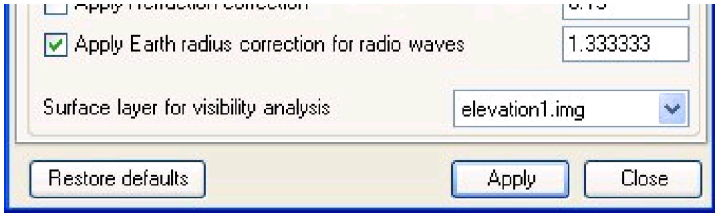
LOS Settings

The settings for the Line of Sight can be adjusted from the Line of Sight Settings dialog (Settings Menu ==> Line of Sight Settings).



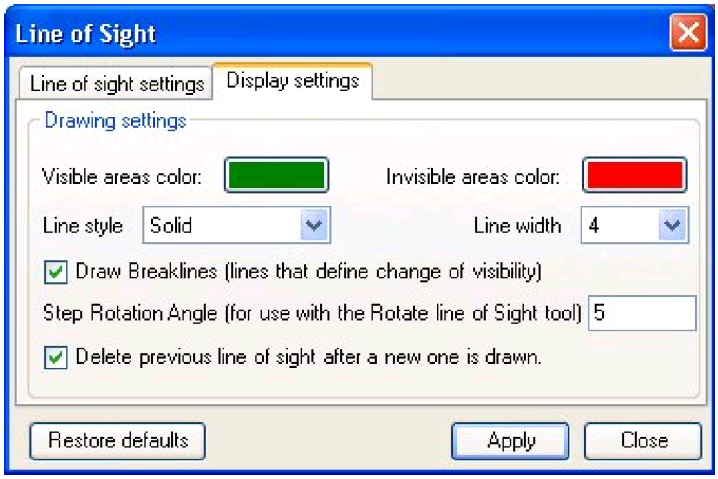

On the Line of sight settings tab the user can change the following parameters:

- Observer offset above the surface
- Target offset above the surface
- Apply corrections for
  - Earth curvature
  - Refraction of light
  - Radio waves refraction




- Change the values for Refraction correction and Earth radius correction for radio waves

[See LOS Discussion]

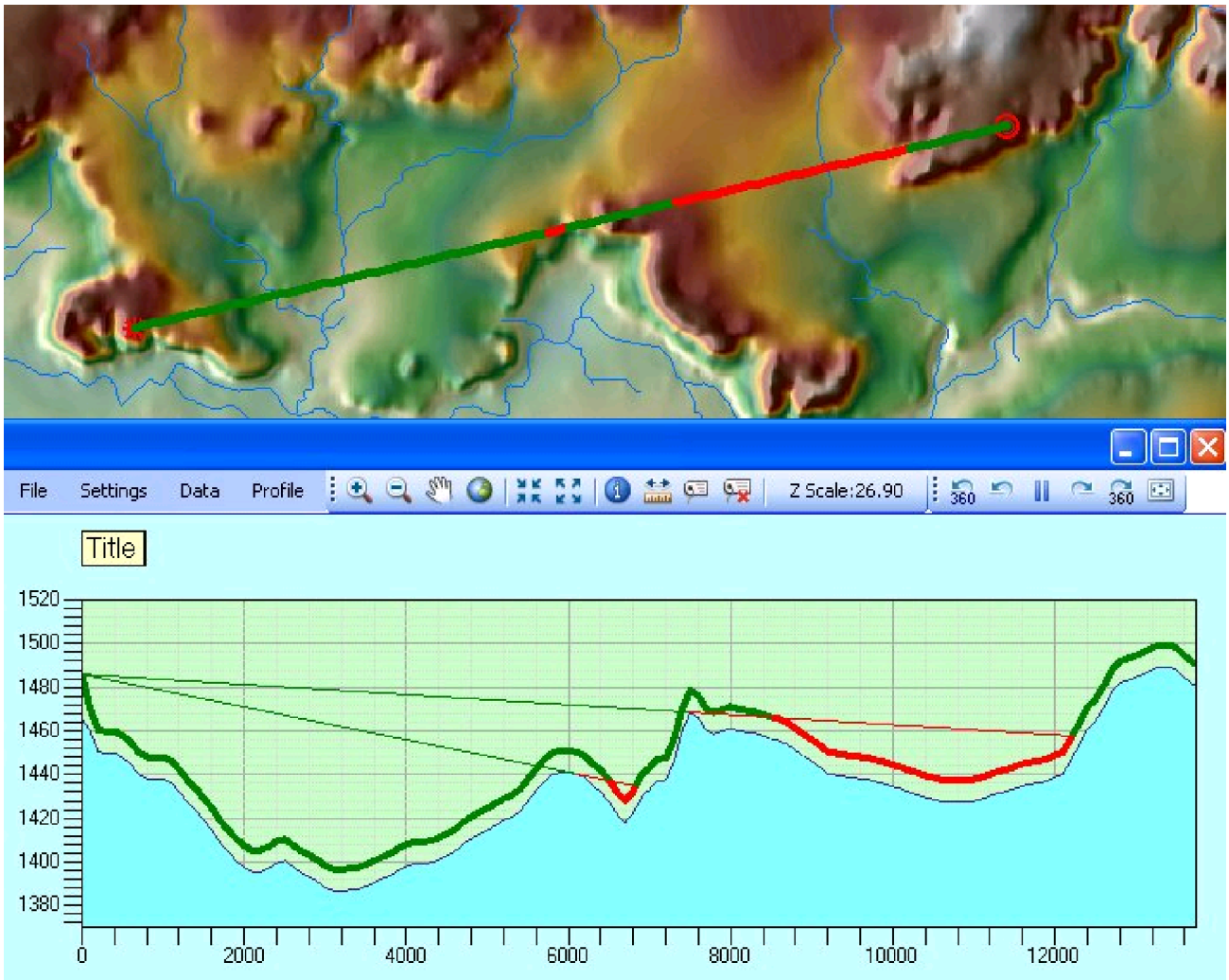



On the Display settings tab the user can change the following parameters:

- Adjust the colors for the visible and non-visible portions of LOS
- Change the Line style and width of LOS lines
- Draw the lines for which the visibility changes (Break Lines)
- Rotation angle for rotating the Observer
- Whether or not the previous LOS graphic to be deleted when a new LOS is calculated and drawn on the View.

EXAMPLE:

Observer and Target offset applied




See examples for applying Earth Curvature, Light and Radio Waves refractions [here]

# Line of Sight (LOS)

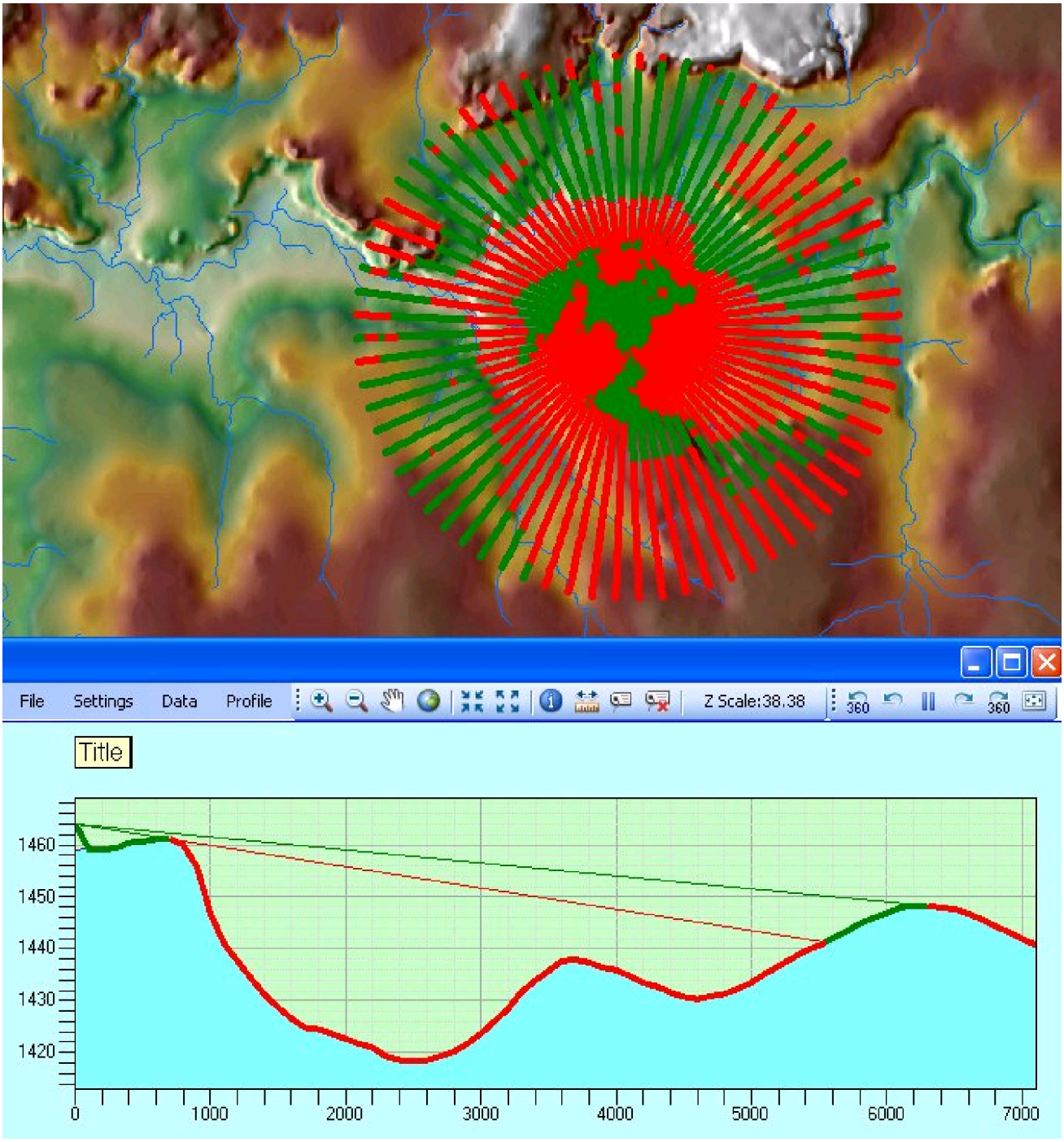
**Line of Sight - Rotate Observer**

Once a single Line LOS is calculated, the user can use the Rotation toolbar 

to rotate the line of sight around the Observer. The step of rotation (in Degrees) can be adjusted on the Line of Sight Settings Dialog. There the user can also set whether the previous LOS graphic will be deleted when a new one is drawn.

During rotation it is possible that the extent of the Line of Sight view will change and go outside of the extent of the window. The Recalculate Extent button  can be used to recalculate the extent of the current Line of Sight and zoom to it.

The example below shows the result of the rotation of the Observer full 360 degrees. The green areas are visible and the red ones are non-visible from the Observer




See the Profile Extractor online tutorial for a video about this functionality.

# Digitize Z Shapes

**Digitize Z Shapes Tools**

The Digitize PointZ  , PolylineZ  and PolygonZ  tools extract for each geometry digitized the Z values from the surface selected in the Surface Layer box and create Z shapes.

If the user digitizes polylines or polygons new vertices are introduced in the digitized geometry depending on the source surface:

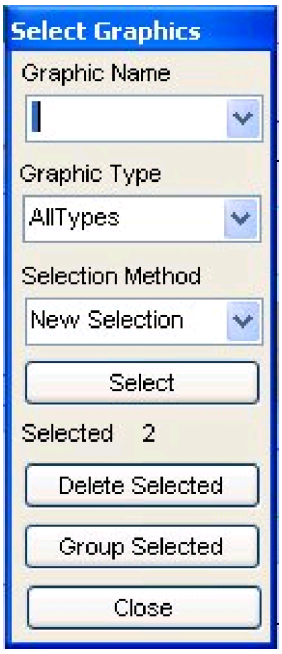
- If the source surface is Raster, the user can change the sample distance to control the number of vertices introduced in the digitized shape. To change the sample distance press the "S" key when the tool is active.
- If the source surface is a TIN, new points are inserted at the intersections of the digitized geometry with the edges of the TIN triangles.

The tools can be used to digitize features or graphics

- IN an Editing Session: If the target layer is set to a Z enabled layer of the type the user digitizes, the geometry captured will be stored as a feature in the target layer otherwise it will be stored as a graphic element. If for example the target layer set in the Editor is of PointZ type and the tool used is Digitize PointZ, the geometry will be stored as a feature. If however the tool used is Digitize PolylineZ, the geometry will be stored as a graphic element
- OUT of an Editing Session: the geometries digitized are stored as graphic elements. The default symbols (set in Drawing ==> Default Symbol Properties) are used for the graphic elements.

# Manage Graphics

The tools of ET Surface draw many graphics elements on the view. This tool allows to manage these graphics elements with ease.




- Upon executing the command all graphics elements in the view are collected. The unique names are populated in a combo box and the user can select a name from the list or type one. If the Graphic Name box is left empty, all graphics will be considered in the selection criteria.
- The types of available graphics are populated in the graphic type combo box
- There are three selection options
  - New Selection
  - Add To Selection
  - Remove From Selection
- The user has options to quickly delete or group selected graphics.

# Build TIN

ToolBox Implementation .NET Implementation

Builds a Triangulated Irregular Network from a feature layer

**Inputs:**

- A feature layer (Point, Polyline, Polygon)
- Type of output - ESRI TIN or PolygonZ TIN
- An elevation field - numeric field that will be used. If the features have Z values, they can be used for elevation.
- If the output is ESRI TIN - triangulation method to be used - "Mass points" or "Hard breaklines"
- If the output is PolygonZ TIN - the Azimuth and the Altitude of the light source

**Outputs:**

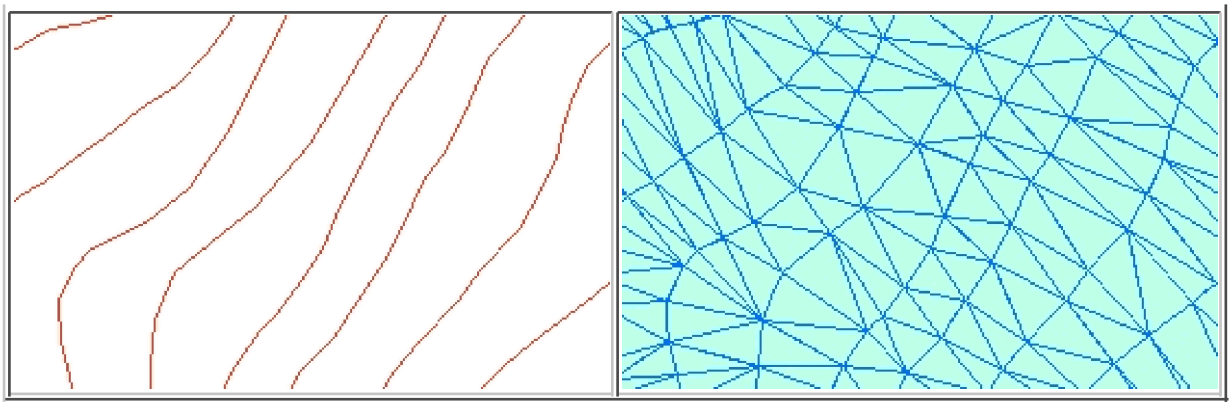
- New ESRI TIN or PolygonZ feature class. All the polygons are triangles that comply with the Delaunay criteria. See TIN notes for more information about Triangulated Irregular Network.
- If the output is ESRI TIN and the input features are polylines or polygons, they can be triangulated as Hard breaklines.
- If the output is PolygonZ TIN, several characteristics are calculated and added in the attribute table for each triangle.
  - ET_ElMin - minimum elevation values for each triangle
  - ET_ElMax - maximum elevation values for each triangle
  - ET_ElMean - mean elevation values for each triangle
  - ET_Slope_D - the slope (maximum rate of elevation change) of each triangle in Degrees (from 0 to 90)
  - ET_Slope_P - the slope (maximum rate of elevation change) of each triangle in percents (from 0 to 100%)
  - ET_Aspect - the aspect - compass direction of the slope (horizontal direction in which a slope faces) - 0 is North, 90 degrees - East, 180 degrees - South, 270 - West of each triangle
  - ET_ACode - aspect categories
    - N - North ( 0 to 22.5 and 337.5 to 360)
    - NE - North East (22,5 to 67.5)
    - E - East (67.5 to 112.5)
    - SE - South East (112.5 to 157.5)
    - S - South (67.5 to 112.5)
    - SW - South West (202.5 to 247.5)
    - W - West (247.5 to 292.5)
    - NW - North West (292.5 to 337.5)
    - U - Undefined - Slope = 0
  - ET_AreaZ - the 3D area of each triangle

**Notes :**

- The process goes through several steps
  - Collecting the elevation points from the source layer. If the source is a polygon or polyline layer, all the vertices are used.
  - Removing duplicate points
  - Creating the TIN structure
  - Analyzing and storing the TIN
- In version 4.0 the TIN creation has been redesigned and can handle much bigger datasets with improved speed. On 32 bit operating systems (Windows XP 32 bit or Windows 7 32 bit) with 4 GB of RAM the function should work with no problems on datasets with up to 6 million points. On 64 bit systems bigger datasets can be processed depending on the available memory

**Example:**

| Source Layer (polyline) | Result TIN |
| --- | --- |



**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the colour coding for specifics.**

ETS_GPBuildESRITIN <Input Dataset>  <Out TIN>  <Elevation Field> <Triangulation type>

ETS_GPBuildPolygonZTIN <Input Dataset>  <Out Feature Class>  <Elevation Field> {Light Azimuth} {Light Altitude}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Point, Polyline or Polygon feature layer or feature class |
| <Out TIN> | A String - the full name of the output ESRI TIN |
| <Out Feature Class> | A String - the full name of the output feature class. |
| <Elevation Field> | A String representing the name of the field to be used as a source for the elevations |
| <Triangulation type> | A String - possible values are "Mass points" and "Hard breaklines" |
| {Light Azimuth} | A Double representing azimuth of the light source (0 to 360). 0 indicates North, 90 - East, 180 - South, 270 - West |
| {Light Altitude} | A Double representing the altitude of the light source  in degrees (0 to 90) |

**Scripting syntax**

ETS_GPBuildESRITIN (Input Dataset, Out TIN, Elevation Field, Triangulation type)

ETS_GPBuildPolygonZTIN(Input Dataset, Out Feature Class, Elevation Field, Light Azimuth, Light Altitude)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

BuildEsriTin(inFeatureClass As IFeatureClass, sOutTinName As String, elevationField As String, triangulationType As String) As ITin

BuildPolygonZTin(inFeatureClass As IFeatureClass, sOutFName As String, elevationField As String, Optional dAzimuth As Double = 315, Optional dAltitude As Double = 45) As IFeatureClass

# Contour To Raster

ToolBox Implementation .NET Implementation

Interpolates a raster surface from contour polylines.

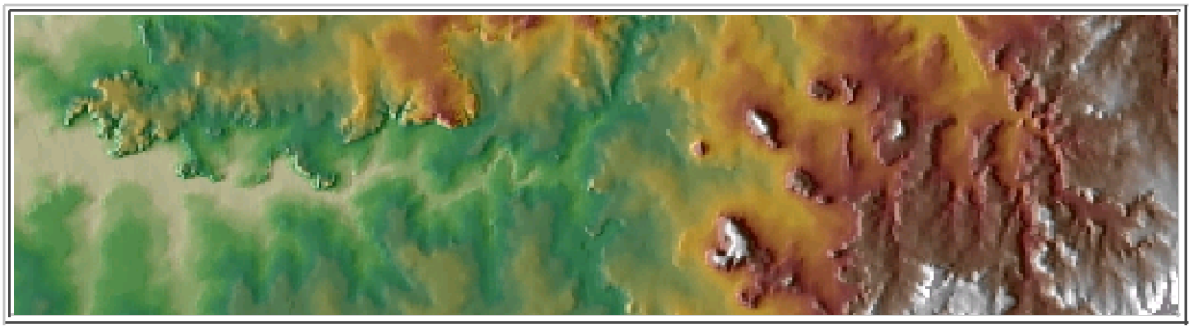
**Inputs:**

- A polyline feature layer representing contours. It is strongly recommended to clean all possible gaps in the contours before using the Contour To Raster function. If your contour dataset has gaps use the Clean Contour Gaps function.
- Output raster name and format
- Cell Size of the output raster
- Elevation field - a field from the attribute table to be used as a source for the values of the raster. The Z values of the input PolylineZ dataset can be also used as source for the raster values.

**Output:**

- A floating point raster.

**Example:**



**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The feature class must be in a projected coordinate system.
- The algorithm is specifically designed to use contours as input. If a polyline dataset that represents a different feature is used the results might be unexpected.

**ToolBox implementation**

**Command line syntax**

ETS_GPContourToRaster <Input Dataset> <Out Raster> <Elevation Field> < Cell Size>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polyline layer or feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

| | |
|---|---|
| <Elevation  Field> | A String representing the name of the field which values are going to be used for interpolation. |
| <Cell Size> | A Double representing the cell size of the output raster. |

**Scripting syntax**

ETS_GPContourToRaster (Input Dataset, Out Raster, Elevation Field, Cell Size)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ContoursToRaster (inFeatureClass As IFeatureClass, sOutRaster As String, sElevationField As String, dCellSize As Double) As IRasterDataset2

# Inverse Distance Weighted (IDW) Interpolation

ToolBox Implementation .NET Implementation

Uses Inverse Distance Weighted Interpolation (IDW) to interpolate a raster from the input features.

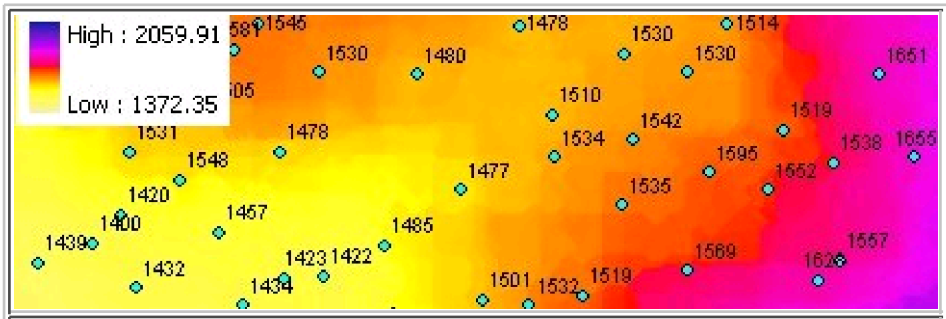
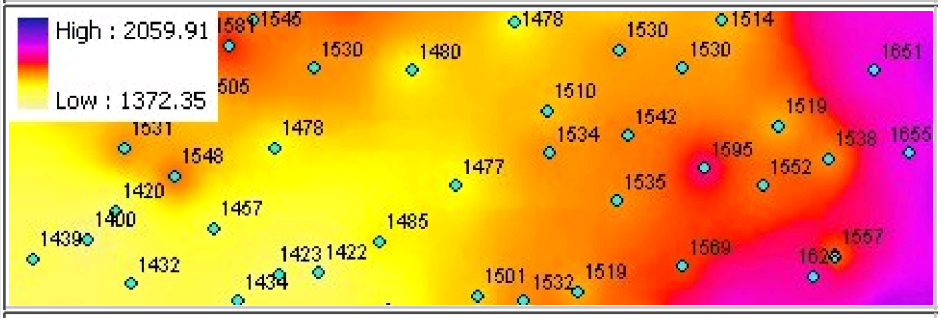
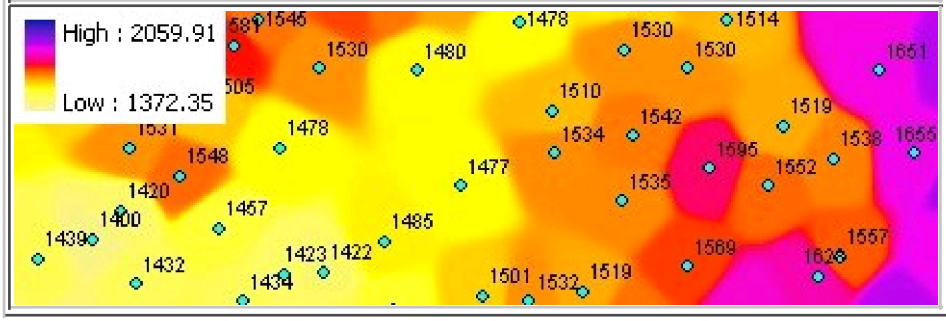
**Inputs:**

- A Point or Polyline feature layer or feature class.
- Output raster name and format
- Cell Size of the output raster
- Value field - a field from the attribute table to be used as a source for the values of the raster. If the input is of PointZ or PolylineZ type, the Z values of the features can be used as source for the raster values
- Power - A positive number that defines the weight of the distance in the interpolation process. The weight of each known point decreases as the distance from it to the interpolated cell increases. The higher the value of the Power, the faster the weight of the known point decreases. If the Power is less than 1 the appearance of the resulting surface will be sharper. If the value of the Power is greater than 1 the appearance of the surface will be smoother. Very large values of the Power will result of a surface with only few known points influencing the value of the interpolated cell. The most commonly used value is 2 (default)
- Number of sources - the number of known points to be used in the interpolation of the value of each cell. The default value is 12.
- Cut off distance (optional) - a known point will not influence the interpolation of cells that are farther than this distance from the point.

**Output:**

- A floating point raster.

**Examples:**

| | |
|---|---|
|   | Power = 0.5 |
|   | Power = 2 |
|   | Power = 15 |

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The feature class must be in a projected coordinate system

**ToolBox implementation**

**Command line syntax**

ETS_GPIDW <Input Dataset> <Out Raster> <Elevation Field> < Cell Size> <Power> <Number of Sources> {Cutoff Distance}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Point, Polyline or Polygon layer or feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Elevation Field> | A String representing the name of the field which values are going to be used for interpolation. |
| <Cell Size> | A Double representing the cell size of the output raster. |
| <Power> | A Number - see main description above |
| <Number of Sources> | A Number - see description above |
| {Cutoff Distance} | A Double representing the Cutoff distance. |

**Scripting syntax**

ETS_GPIDW (Input Dataset, Out Raster, Elevation Field, Cell Size, Power, Number of Sources ,Cutoff Distance)

See the explanations above:
<> - required parameter
{} - optional parameter

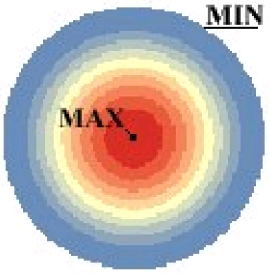
**.NET implementation**
(Go to TOP)

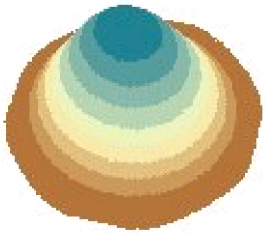
IDW (inFeatureClass As IFeatureClass, sOutRaster As String, sElevationField As String, dCellSize As Double, dPower As Double, iNumPoints As Integer, Optional dCutoff As Double = 0) As IRasterDataset2

# Kernel Density

ToolBox Implementation .NET Implementation

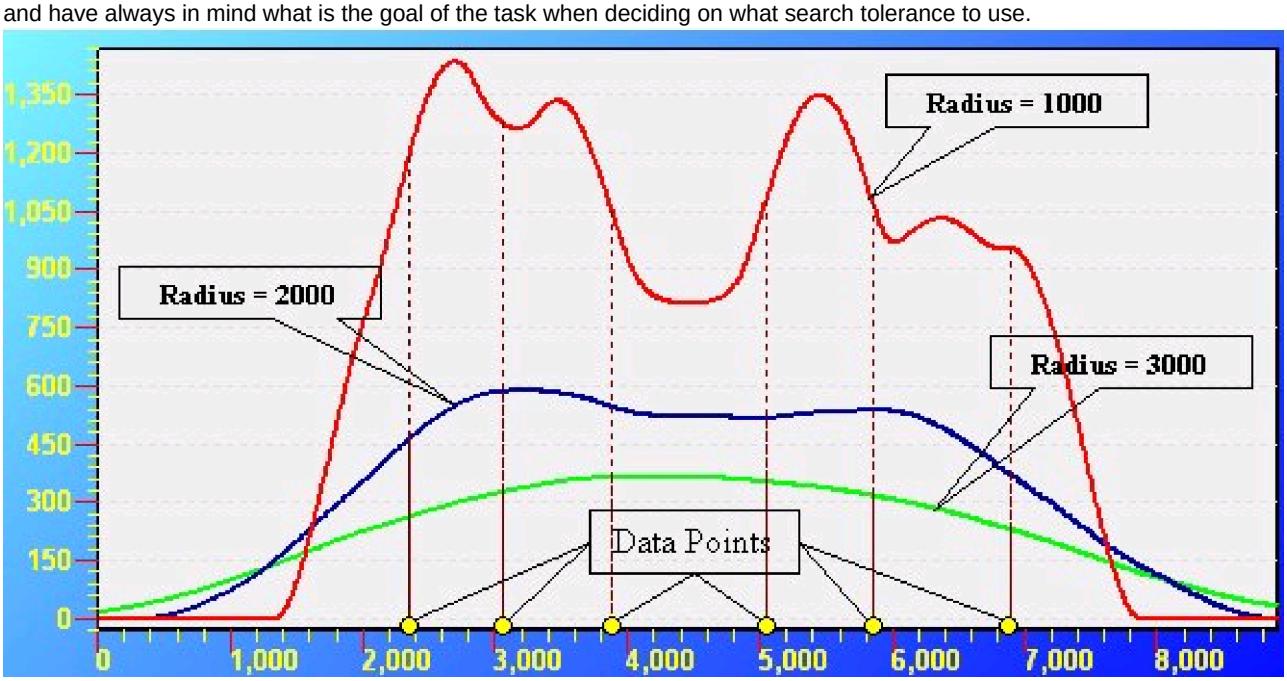
Uses Kernel Density Estimation to interpolate a surface from the input Point or Polyline features. The function fits a symmetrical surface over each input point using Gaussian kernel.



The surface has maximum value in the input point, decreases as the distance from the input point increases and has 0 values at distance equal to the search radius.
The volume of the created surface is equal to the value of the input point.
The density of each cell of the output raster is calculated by adding the values of all individual surfaces for that cell.

The search radius does not influence the volume of the surface, but has a major impact on the generalization of the data. The larger the search tolerance, the smoother and more generalized surface will be interpolated. As you can see from the image below the selection of the search radius is very important and influences greatly the surface. You should evaluate your data and have always in mind what is the goal of the task when deciding on what search tolerance to use.



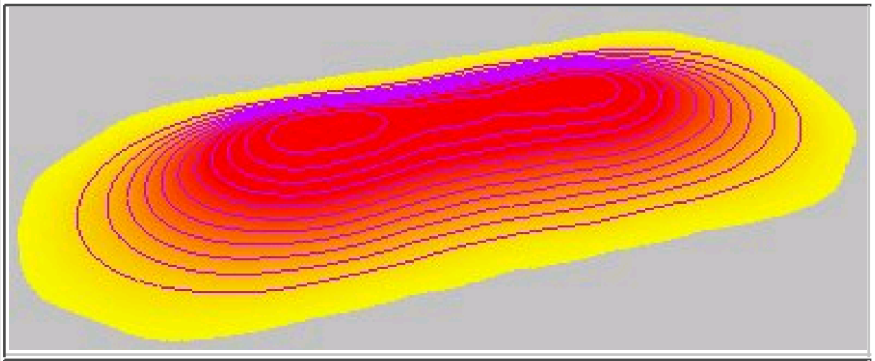
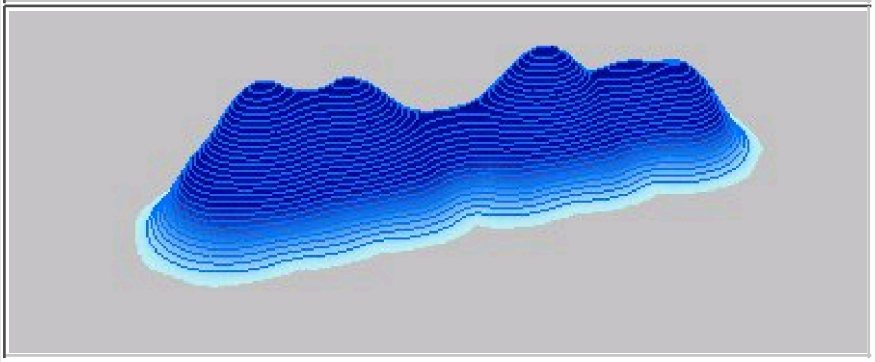
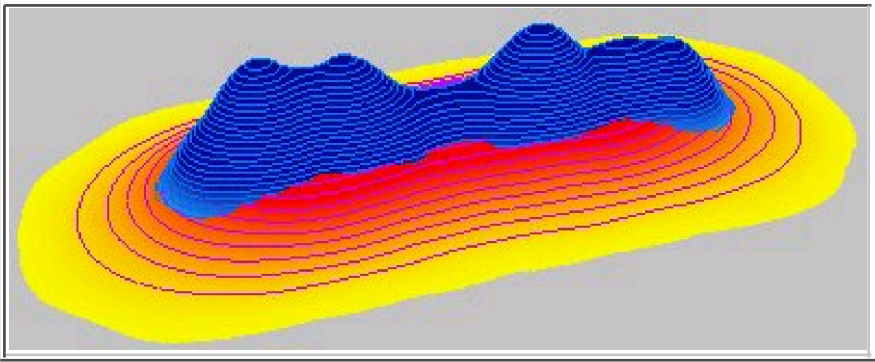

**Inputs:**

- A Point or Polyline feature layer or feature class.
- Output raster name and format
- Cell Size of the output raster
- Search radius.
- Value field - a field from the attribute table to be used as value for each point. The value might be the population at this point, the number of incidents at the location etc. If you do not have such a field in the attribute table, just create a new field and calculate the values of all records equal to 1.
- Area Units - The value of each cell of the output raster will actually have value measured in Value per square unit. The default unit is the unit of the spatial reference of the input dataset, but you might want to change this to a different area unit (for example incidents per square kilometer)

**Output:**

- A floating point raster.

**Example - the surface created from the 6 points in the profile above:**

| | |
|---|---|
|  | Search Radius = 2000 meters - smoother more generalized appearance |
|  | Search Radius = 1000 meters - more pronounced influence of the individual points |
|  | Both surfaces superimposed - two very different surfaces. The volume of both however is the same and is equal to the sum of the values of the input points. |

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class must be in a projected coordinate system

**ToolBox implementation**

**Command line syntax**

ETS_GPDensity <Input Dataset> <Out Raster> <Elevation Field> < Cell Size> <Interpolate Radius> <Area Units>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Point, Polyline or Polygon layer or feature class |

| | |
|---|---|
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Value Field> | A String representing the name of the field which values are going to be used for interpolation. |
| <Cell Size> | A Double representing the cell size of the output raster. |
| <Interpolate Radius> | A Number - see main description above |
| <Area Units> | A String - possible values are "SquareMeters", "SquareKilometers", "Hectares", "SquareFeet", "SquareMiles", "Acres", "SquareYards", "Decares", "Ares". |

**Scripting syntax**

ETS_GPDensity (Input Dataset, Out Raster, Value Field, Cell Size, Interpolate Radius, Area Units)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

Density (inFeatureClass As IFeatureClass, sOutRaster As String, sElevationField As String, dCellSize As Double, dRadius As Double, Optional areaUnits As String = "") As IRasterDataset2

# Modify TIN

ToolBox Implementation .NET Implementation

Modifies an ESRI TIN or PolygonZ TIN by adding additional data points or breaklines from a feature layer. The output TIN is of the same type as the input TIN.

**Inputs:**

- An ESRI TIN or PolygonZ TIN to be modified.
- A feature layer (Point, Polyline, Polygon)
- An elevation field - numeric field that will be used. If the features have Z values, they can be used for elevation.
- If the output is ESRI TIN - triangulation method to be used - "Mass points", "Hard breaklines" or "Soft breaklines"
- If the output is PolygonZ TIN - the Azimuth and the Altitude of the light source

**Outputs:**

- New ESRI TIN or PolygonZ feature class.

**Notes :**

- If the input is ESRI TIN, the modify features can be added as Mass points, Hard breaklines or Soft breaklines. The elevation values for Hard breaklines are defined by a field or by the Z values of the features. For Soft breaklines the elevation values are calculated from the input TIN.
- If the input is PolygonZ TIN, the modify data can only be added as Mass points.

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the colour coding for specifics.**

ETS_GPModifyESRITIN <Input ESRI TIN> <Modify Dataset> <Output ESRI TIN> <Triangulation type> <Elevation Field>

ETS_GPModifyPolygonZTIN <Input PolygonZ TIN> <Modify Dataset> <Output PolygonZ TIN> <Elevation Field> {Light Azimuth} {Light Altitude}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Modify Feature Class> | A Point, Polyline or Polygon feature layer or feature class |
| <Output ESRI TIN> | A String - the full name of the output ESRI TIN |
| <Output PolygonZ TIN> | A String - the full name of the output PolygonZ TIN (feature class). |
| <Elevation Field> | A String representing the name of the field to be used as a source for the elevations |
| <Triangulation type> | A String - possible values are "Mass points", "Hard breaklines" and "Soft breaklines" |
| {Light Azimuth} | A Double representing azimuth of the light source (0 to 360). 0 indicates North, 90 - East, 180 - South, 270 - West - default is 315 |
| {Light Altitude} | A Double representing the altitude of the light source in degrees (0 to 90) - default is 45 |

**Scripting syntax**

ETS_GPModifyESRITIN (Input TIN, Modify Dataset, Output ESRI TIN, Triangulation type, Elevation Field)

ETS_GPModifyPolygonZTIN(Input PolygonZ TIN, Modify Dataset, Output PolygonZ TIN, Elevation Field, Light Azimuth, Light Altitude)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ModifyEsriTin (inTIN As ITin, inFeatureClass As IFeatureClass, sOutTinName As String, triangulationType As String, elevationField As String) As ITin

ModifyPolygonZTin (tinFeatureClass As IFeatureClass, modifyFeatureClass As IFeatureClass, sOutFName As String, elevationField As String, Optional dAzimuth As Double = 315, Optional dAltitude As Double = 45) As IFeatureClass

# ESRI TIN to PolygonZ TIN

ToolBox Implementation .NET Implementation

Converts an ESRI TIN to a PolygonZ TIN feature class. Calculates several 3D characteristics for each triangle and stores them in the attribute table

**Inputs:**

- An ESRI TIN
- Options for calculating Z characteristics of the triangles

**Outputs:**

- New polygon Z feature class.
- The PolygonZ TIN will be analyzed and several characteristics will be added in the attribute table for each triangle.
  - ET_ElMin - minimum elevation values for each triangle
  - ET_ElMax - maximum elevation values for each triangle
  - ET_ElMean - mean elevation values for each triangle
  - ET_Slope_D - the slope (maximum rate of elevation change) of each triangle in Degrees (from 0 to 90)
  - ET_Sope_P - the slope (maximum rate of elevation change) of each triangle in percent (from 0 to 100%)
  - ET_Aspect - the aspect - compass direction of the slope (horizontal direction in which a slope faces) - 0 is North, 90 degrees - East, 180 degrees - South, 270 - West of each triangle
  - ET_ACode - aspect categories
    - N - North ( 0 to 22.5 and 337.5 to 360)
    - NE - North East (22,5 to 67.5)
    - E - East (67.5 to 112.5)
    - SE - South East (112.5 to 157.5)
    - S - South (67.5 to 112.5)
    - SW - South West (202.5 to 247.5)
    - W - West (247.5 to 292.5)
    - NW - North West (292.5 to 337.5)
    - U - Undefined - Slope = 0
  - ET_AreaZ - the 3D area of each triangle
  - ET_Hill - the brightness of each triangle based on a light source location

**ToolBox implementation**

**Command line syntax**

ETS_GPEsriTINToPolygonZ <Input TIN> <Out Feature Class> {Light Azimuth} {Light Altitude}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input TIN> | An ESRI TIN layer or dataset |
| <Out Feature Class> | A String - the full name of the output feature class. |
| {Light Azimuth} | A Double representing azimuth of the light source (0 to 360). 0 indicates North, 90 - East, 180 - South, 270 - West |
| {Light Altitude} | A Double representing the altitude of the light source in degrees (0 to 90) |

Scripting syntax

ETS_GPEsriTINToPolygonZ (Input TIN, Out Feature Class, Light Azimuth, Light Altitude)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ESRITin2PolygonZTin(inTin As ITin, sOutFeature As String, Optional dAzimuth As Double = 315, Optional dAltitude As Double = 45) As IFeatureClass

# TIN to Edges

ToolBox Implementation .NET Implementation

Exports the edges of an ESRI TIN or a PolygonZ TIN to a new polyline feature class. Calculates several 3D characteristics for each edge and stores them in the attribute table.

**Inputs:**

- An ESRI TIN or PolygonZ TIN
- If the input is an ESRI TIN the user can export specific type of edges only (Regular, Soft, Hard) or all edges.

**Outputs:**

- New Polyline or PolylineZ feature class.
- Several characteristics will be calculated for each edge and stored in the following fields.
  - ET_Slope_D - the slope of the edge in Degrees (from 0 to 90)
  - ET_Azimuth - the Azimuth of the edge in Degrees "North Azimuth" orientation - 0 = North, clockwise.
  - ET_2DLen - 2D length of the edge
  - ET_3DLen - 3D length of the edge
  - ET_Type - the type of the edge. All edges of a PolygonZ TIN will have value "Regular"

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the colour coding for specifics.**

ETS_GPEdgesEsriTIN <Input ESRI TIN> <Out Feature Class> {3D Features} <Edge Type>

ETS_GPEdgesPolygonZTIN <Input PolygonZ TIN> <Out Feature Class> {3D Features}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| {3D Features} | A Boolean indicating whether the result will be PolylineZ or Polyline |
| <Edge Type> | A string indicating the type of the edges to be exported. Valid values "All", "Hard", "Regular", "Soft" |

**Scripting syntax**

ETS_GPEdgesEsriTIN (Input ESRI TIN, Out Feature Class, 3D Features, Edge Type)

ETS_GPEdgesPolygonZTIN (Input PolygonZ TIN, Out Feature Class, 3D Features)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

EdgesESRITin (inTin As ITin, sOutFeature As String, bZ As Boolean, sOption As String) As IFeatureClass

EdgesPolygonZTin (inFeatureClass As IFeatureClass, sOutFeature As String, bZ As Boolean) As IFeatureClass

# TIN to Nodes

ToolBox Implementation .NET Implementation

Exports the Nodes of an ESRI TIN or a PolygonZ TIN to a new point feature class.

**Inputs:**

- An ESRI TIN or PolygonZ TIN

**Outputs:**

- New Point or PointZ feature class.
- A new field is added to the point attribute table.
  - ET_Spot - the Z value of the Node

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPNodesEsriTIN <Input ESRI TIN> <Out Feature Class> {3D Features}

ETS_GPNodesPolygonZTIN <Input PolygonZ TIN> <Out Feature Class> {3D Features}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| {3D Features} | A Boolean indicating whether the result will be PointZ or Point |

**Scripting syntax**

ETS_GPNodesEsriTIN (Input ESRI TIN, Out Feature Class, 3D Features)

ETS_GPNodesPolygonZTIN (Input PolygonZ TIN, Out Feature Class, 3D Features)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

NodesESRITin (inTin As ITin, sOutFeature As String, bZ As Boolean) As IFeatureClass

NodesPolygonZTin (inFeatureClass As IFeatureClass, sOutFeature As String, bZ As Boolean) As IFeatureClass

# Polygon To Multipatch

ToolBox Implementation .NET Implementation

Converts a polygon feature class to a Multipatch feature class. The Z values are interpolated from an ESRI TIN or PolygonZ TIN.

The portion of the TIN within each polygon is extracted. A new TIN is interpolated from the participating nodes of the reference TIN surface and the vertices of the polygon for which the Z value is interpolated from the TIN. The new TIN is stored as a Multipatch geometry.

**Inputs:**

- A Polygon feature class
- An ESRI TIN or PolygonZ TIN

**Outputs:**

- New Multipatch feature class.
- New fields are added to the attribute table of the Multipatch feature class.
  - ET_ElMin - the minimum Z value
  - ET_ElMax - the maximum Z value
  - ETSlopeMax - the maximum slope
  - ET_AreaZ - the 3D area
  - ET_Area - the 2D area

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPPolygonToMultipatchEsriTIN <Input Dataset> <Input ESRI TIN> <Out Feature Class>

ETS_GPPolygonToMultipatchPolygonZTIN <Input Dataset> <Input PolygonZ TIN> <Out Feature Class>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polygon feature layer or feature class |
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |

**Scripting syntax**

ETS_GPPolygonToMultipatchEsriTIN (Input Dataset, Input ESRI TIN, Out Feature Class)

ETS_GPPolygonToMultipatchPolygonZTIN (Input Dataset, Input PolygonZ TIN, Out Feature Class)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

PolygonToMultipatchEsriTIN (inFeatureClass As IFeatureClass, inTin As ITin, sOutFeature As String) As IFeatureClass

PolygonToMultipatchPolygonZTIN (inFeatureClass As IFeatureClass, polygonZTin As IFeatureClass, sOutFeature As String) As IFeatureClass

# Multipatch to Polygons

ToolBox Implementation .NET Implementation

Converts a Multipatch feature class to a PolygonZ feature class. Several 3D characteristics are calculated for each polygon and stored in the polygon attribute table.

If the input Multipatch is constructed by Triangle Strips and Triangle Fan geometries, the resulting feature class can be treated as a PolygonZ TIN

**Inputs:**

- A Multipatch feature class

**Outputs:**

- New PolygonZ feature class.
- New fields are added to the attribute table of the Multipatch feature class.
  - ET_ElMin - minimum elevation values for each triangle
  - ET_ElMax - maximum elevation values for each triangle
  - ET_ElMean - mean elevation values for each triangle
  - ET_Slope_D - the slope (maximum rate of elevation change) of each triangle in Degrees (from 0 to 90)
  - ET_Sope_P - the slope (maximum rate of elevation change) of each triangle in percents (from 0 to 100%)
  - ET_Aspect - the aspect (compass direction of the slope - 0 is North, 90 degrees - East, 180 degrees - South, 270 - West) of each triangle
  - ET_AreaZ - the 3D area of each triangle
  - ET_Area - the 2D area of each triangle

**ToolBox implementation**

**Command line syntax**

ETS_GPMultipatchToPolygon <Input Dataset> <Out Feature Class>

Parameters

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Multipatch feature layer or feature class |
| <Out Feature Class> | A String - the full name of the output feature class. |

Scripting syntax

ETS_GPMultipatchToPolygon (Input Dataset, Out Feature Class)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

MultipatchToPolygon (inFeatureClass As IFeatureClass, sOutFeature As String) As IFeatureClass

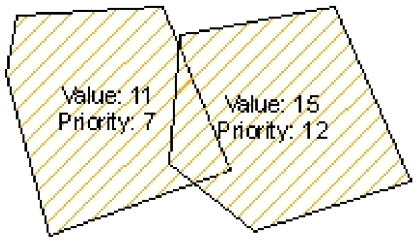
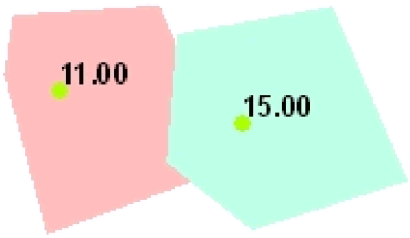
# Features to Raster

ToolBox Implementation .NET Implementation

Converts features (Points, Polylines, Polygons) to a raster dataset.

**Inputs:**

- A feature layer (Point, Polygon, Points, Polylines, Polygons)
- Output raster name and format
- Cell Size of the output raster
- Value field - a field from the attribute table to be used as a source for the values of the raster. If the input is of PointZ or PolylineZ type, the Z values of the features can be used as source for the raster values
- Priority field (optional) - a field from the attribute table which values define how the values will be stored in the raster in case two or more features cover the same cell. If for example two overlapping polygons are converted to a raster, the polygon with the larger value in this field will be stored in the raster.

| Source polygons | Result Raster |
|---|---|
|   |   |

- Extent from existing raster layer (optional). If selected the extent of an existing raster layer will be used for the output, otherwise the extent of the input feature class will be used.
- For a polyline layer only - rasterize the polyline including all cells touched (optional).

**Output**

- A raster with data type depending on the input value field. If the field is Integer the result will be integer, if the field is Double, the result will be a floating point raster.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The feature class must be in a projected coordinate system

**ToolBox implementation**

**Command line syntax**

ETS_GPFeaturesToRaster <Input Dataset> <Out Raster> <Value Field> < Cell Size> {Priority Field} {Extents Raster} {Thick Line}

**Parameters**

| Expression | Explanation |
|---|---|

| | |
|---|---|
| <Input Dataset> | A Point, Polyline or Polygon layer or feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Value Field> | A String representing the name of the field which values are going to be used for interpolation. |
| <Cell Size> | A Double representing the cell size of the output raster. |
| {Priority Field} | A String representing the name of the field which values are going to be used define the priority of the features. See explanations above |
| {Extents Raster} | A raster layer or raster dataset to be used to define the extents of the output raster |
| {Thick Line} | For polylines include all cells touched by the polyline. |

**Scripting syntax**

ETS_GPFeaturesToRaster (Input Dataset, Out Raster, Value Field, Cell Size, Priority Field, Extents Raster, Thick Line)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

FeaturesToRaster (inFeatureClass As IFeatureClass, sOutRaster As String, sValueField As String, dCellSize As Double, Optional sPriorityField As String = "", Optional extentRaster As IRaster = Nothing, Optional bThick As Boolean = False) As IRasterDataset2

# ESRI TIN to Raster

ToolBox Implementation .NET Implementation

Converts an ESRI TIN to a raster

**Inputs:**

- An ESRI TIN
- Output raster name and format
- Cell Size of the output raster

**Output**

- A floating point raster.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input TIN must be in a projected coordinate system

**ToolBox implementation**

**Command line syntax**

ETS_GPEsriTINToRaster<Input ESRI TIN> <Out Raster> <Cell Size>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input ESRI TIN> | An ESRI TIN dataset or layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Cell Size> | A Double representing the cell size of the output raster. |

**Scripting syntax**

ETS_GPEsriTINToRaster(Input ESRI TIN, Out Raster, Cell Size)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

EsriTinToRaster (inTin As ITin, sOutRaster As String, dCellSize As Double) As IRasterDataset2

# Features To 3D

ToolBox Implementation .NET Implementation

A set of three functions used to convert 2D feature classes to 3D feature classes by interpolating the Z values from a reference surface - Raster, ESRI TIN, PolygonZ TIN.

If the input feature class is of Polyline or Polygon type, additional vertices can be added to the geometries in order to fit them better to the terrain.

- If the reference surface is a Raster, the user can specify sample distance. New vertices will be added to the geometry at the sample distance specified, then Z values will be interpolated for all vertices. If sample distance is not assigned or set to 0, no new vertices will be added - Z values will be interpolated for the original vertices only
- If the reference surface is a TIN, the user can specify if only the original vertices to be used or new vertices to be inserted.

**Inputs:**

- A Point, Polyline or Polygon feature class
- A reference surface - Raster, ESRI TIN or PolygonZ TIN

**Outputs:**

- New PointZ, PolylineZ or PolygonZ feature class.
- The attributes of the input dataset are preserved.

**ToolBox implementation**

**Command line syntax - Three different toolbox tools available depending on the reference surface. Check the color coding for specifics.**

ETS_GPFeaturesTo3DRaster <Input Dataset> <Reference Raster> <Out Feature Class> {Sample Distance}

ETS_GPFeaturesTo3DEsriTIN <Input Dataset> <Reference TIN> <Out Feature Class> {Vertices Only}

ETS_GPFeaturesTo3DPolygonZTIN <Input Dataset> <Reference TIN> <Out Feature Class> {Vertices Only}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Point, Polyline or Polygon layer or feature class |
| <Reference Raster> | A Raster layer or Raster dataset. |
| <Reference TIN> | An ESRI TIN layer or dataset |
| <Reference TIN> | A PolygonZ TIN (feature class or feature layer) |
| <Out Feature Class> | A String - the full name of the outputfeature class |
| {Sample Distance} | A Double representing the sample distance |
| {Vertices Only} | A Boolean. If TRUE Z values will be interpolated for the existing vertices of the input features. If False new vertices will be added to the input features in the places of intersection with the TIN edges. |
| {Vertices Only} | A Boolean. If TRUE Z values will be interpolated for the existing vertices of the input features. If False new vertices will be added to the input features in the places of intersection with the TIN edges. |

**Scripting syntax**

ETS_GPFeaturesTo3DRaster (Input Dataset, Reference Raster, Out Feature Class, Sample Distance)

ETS_GPFeaturesTo3DEsriTIN (Input Dataset, Reference TIN, Out Feature Class, Vertices Only)

ETS_GPFeaturesTo3DPolygonZTIN (Input Dataset, Reference TIN, Out Feature Class, Vertices Only)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

FeaturesTo3DRaster (inFeatureClass As IFeatureClass, inRasterDataset As IRasterDataset2, sOutFeature As String, dSample As Double) As IFeatureClass

FeaturesTo3DESRITin (inFeatureClass As IFeatureClass, inTin As ITin, sOutFeature As String, Optional bVertex As Boolean = False) As IFeatureClass

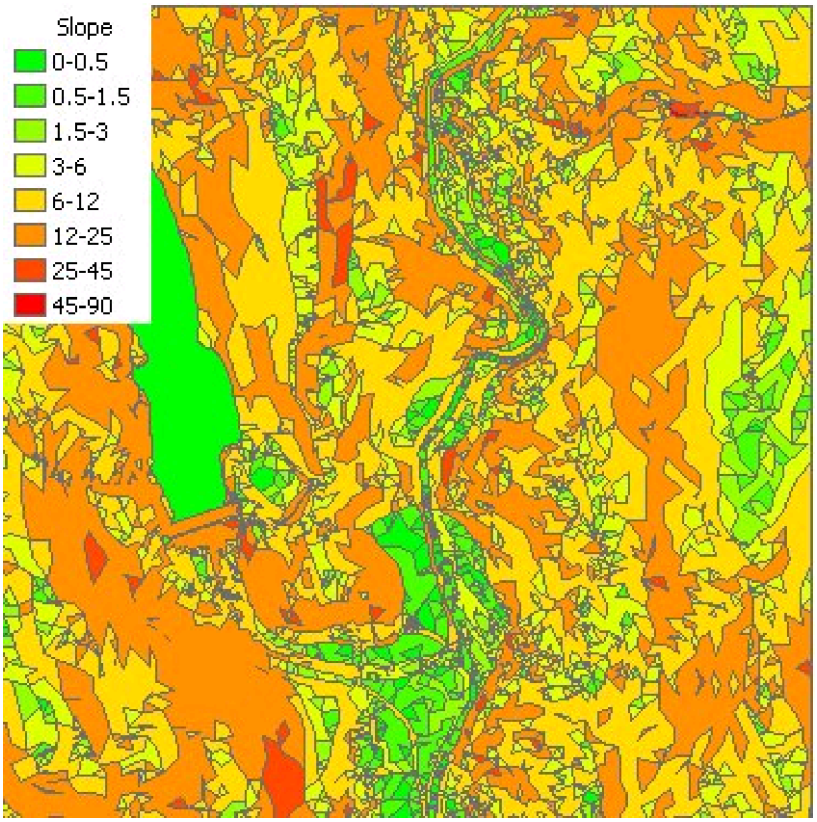
FeaturesTo3DPolygonZTin (inFeatureClass As IFeatureClass, inPolygonZTin As IFeatureClass, sOutFeature As String, Optional bVertex As Boolean = False) As IFeatureClass

# TIN Slope

ToolBox Implementation

.NET Implementation

Calculates the slope of the TIN Triangles of the input ESRI TIN or PolygonZ TIN in percent or degrees (0 to 90). Categorizes the triangles and groups them based on the slope. The user can input a text file to be used for defining the groups to be created. If not such file is provided the default grouping will be used.




**Inputs:**

- A ESRI TIN or PolygonZ TIN
- Slope option
  - Percent
  - Degrees
- Optional: Class Breaks text file

**Outputs:**

- A polygon feature class
- A new field ET_SCode will be added to the polygon attribute table. The values in this field will indicate the slope groups.

**Default breaks**

Slope in degrees

- 0 - 0.5
- 0.5 - 1.5
- 1.5 - 3.0
- 3.0 - 6.0
- 6.0 - 12.0
- 12.0 - 25.0
- 25.0 - 45.0
- 45.0 - 90.0

Slope in percents

- 0 - 1.0
- 1.0 - 2.0
- 2.0 - 5.0
- 5.0 -10.0
- 10.0 -20.0
- 20.0 - 50.0
- 50.0 - 100.0
- Above 100

**Breaks file format**

The file consist of lines with 2 comma delimited values. The first value is the break value and the second the Group ID. The Group ID can be numeric or string, the only requirement is to be unique. The first value will define the first group, if for example

the value is 5, the first group will be 0 to 5. Example text file:

| | |
|---|---|
| Break, Group | The file on the left will produce 6 groups |
| 1, Flat | 0 - 1 Flat |
| 5,Low | 1 - 5 Low |
| 10,Low To Moderate | 5 - 10 Low To Moderate |
| 25,Moderate | 10 - 25 Moderate |
| 35,High | 25 - 35 High |
| | > 35 Above High |

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPSlopeFromEsriTIN <Input ESRI TIN>  <Out Feature Class> <Slope Units> {Class Break File}

ETS_GPSlopeFromPolygonZTIN <Input PolygonZ TIN>  <Out Feature Class>  {Class Break File}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| <Slope Units> | A String indicating the units to be used for slope. Valid values "Degree", "Percent" |
| {Class Break File} | A String - the full name of the class breaks file. |

**Scripting syntax**

ETS_GPSlopeFromEsriTIN (Input ESRI TIN, Out Feature Class, Slope Units, Class Break File)

ETS_GPSlopeFromPolygonZTIN (Input PolygonZ TIN, Out Feature Class, Slope Units, Class Break File)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

SlopeFromEsriTin (inTin As ITin, sOutFName As String, sPercent As String, Optional sClassBreakFile As String = "") As IFeatureClass

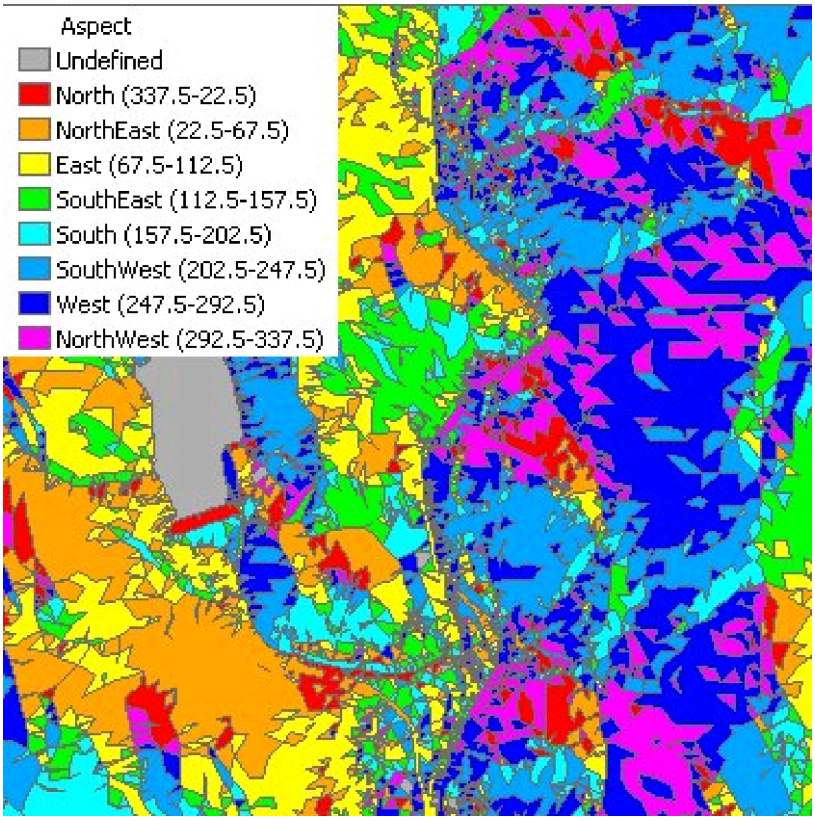
SlopeFromPolygonZTin (inFeatureClass As IFeatureClass, sOutFName As String, sPercent As String, Optional sClassBreakFile As String = "") As IFeatureClass

# TIN Aspect

ToolBox Implementation

.NET Implementation

Calculates the aspect (compass direction of the slope) for each TIN Triangle. Groups the triangles based on their Aspect category and creates a polygon dataset with 9 Aspect categories.




**Inputs:**

- A ESRI TIN or PolygonZ TIN

**Outputs:**

- A polygon feature class
- A new field ET_ACode will be added to the polygon attribute table. The values in this field will indicate the aspect groups.
    - N - North ( 0 to 22.5 and 337.5 to 360)
    - NE - North East (22,5 to 67.5)
    - E - East (67.5 to 112.5)
    - SE - South East (112.5 to 157.5)
    - S - South (67.5 to 112.5)
    - SW - South West (202.5 to 247.5)
    - W - West (247.5 to 292.5)
    - NW - North West (292.5 to 337.5)
    - U - Undefined - Slope = 0

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPAspectFromEsriTIN <Input ESRI TIN>  <Out Feature Class>

ETS_GPAspectFromPolygonZTIN <Input PolygonZ TIN>  <Out Feature Class>

**Parameters**

| Expression | Explanation |
| --- | --- |

| | |
|---|---|
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |

**Scripting syntax**

ETS_GPAspectFromEsriTIN (Input ESRI TIN, Out Feature Class)

ETS_GPAspectFromPolygonZTIN (Input PolygonZ TIN, Out Feature Class)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

AspectFromEsriTin (inTin As ITin, sOutFName As String) As IFeatureClass

AspectFromPolygonZTin (inFeatureClass As IFeatureClass, sOutFName As String) As IFeatureClass

# Interpolate Contours

ToolBox Implementation .NET Implementation

Interpolates contours from an ESRI TIN or PolygonZ TIN. The contours can be created as PolylineZs with constant Z values representing the elevation.

**Inputs:**

- An ESRI TIN or PolygonZ TIN
- Base value - the contour from which to begin generation of contours.
- Contour interval - Z value difference between adjacent contours in map units.
- Option for interpolating 3D contours.

**Outputs:**

- New polyline feature class.
- A new field ET_Height is added to the attribute table. The values of this field contain the Z value for each contour.

**Notes:**

- If flat triangles (see TIN Notes ) are present, some small problems might occur in the contours. These problems are easy to identify using Export Nodes Wizard. Since the contour lines never intersect each other, there should not be Regular Nodes in the contour layer. A regular node in this case will indicate an error ( most probably caused by a flat triangle ). The excess polyline can be deleted.

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPContoursFromEsriTIN <Input ESRI TIN> <Out Feature Class> <Base Contour> <Contour Interval> {3D Contours}

ETS_GPContoursFromPolygonZTIN <Input PolygonZ TIN> <Out Feature Class> <Base Contour> <Contour Interval> {3D Contours}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| <Base Contour> | A Double representing the Z level of the contour from which to begin generation of contours. |
| <Contour Interval> | A Double representing the contour interval |
| {3D Contours} | A Boolean indicating whether the output to be Polyline or PolylineZ feature class |

**Scripting syntax**

ETS_GPContoursFromEsriTIN (Input ESRI TIN, Out Feature Class, Base Contour, Contour Interval, 3D Contours)

ETS_GPContoursFromPolygonZTIN (Input PolygonZ TIN, Out Feature Class, Base Contour, Contour Interval, 3D Contours)

See the explanations above:
<> - required parameter

{} - optional parameter

**.NET implementation**
(Go to TOP)

InterpolateContoursEsriTin (inTin As ITin, sOutFName As String, baseContour As Double, contourInterval As Double, Optional bZ As Boolean = False) As IFeatureClass

InterpolateContoursPolygonZTin (inPolygonZTIN As IFeatureClass, sOutFName As String, baseContour As Double, contourInterval As Double, Optional bZ As Boolean = False) As IFeatureClass

# Volume of TIN

ToolBox Implementation .NET Implementation

Calculates the Volume and 3D Area of an ESRI TIN or PolygonZ TIN above and below a horizontal reference plane defined by the user specified level. The function analyzes the location of each triangle relatively to the reference plane and calculates its volume and area below and above the plane. The result is the accumulated values for the below and above portions of the triangles stored in a new or existing text file.

**Inputs:**

- An ESRI TIN or PolygonZ TIN
- Level of the reference plane.
- Output text file

**Outputs:**

- The output text file will contain
  - The name of the Surface
  - The level of the reference plane
  - Volume above the plane
  - Volume below the plane
  - 3D Area below the plane
  - 3D Area above the plane

**Example:**

Surface: dtm_tin
Level: 187
Volume Above: 701506210.390127
Volume Below: 208341325.899824
Area Above: 9690039.84649869
Area Below: 6662731.4224316

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPVolumeOfEsriTIN <Input ESRI TIN> <Out File> <Level>

ETS_GPVolumeOfPolygonZTIN <Input PolygonZ TIN> <Out File> <Level>

Parameters

| Expression | Explanation |
|---|---|
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out File> | A String - the full name of the output text file. |
| <Level> | A Double indicating the level of the reference plane for which the volume calculation will be performed. |

Scripting syntax

ETS_GPVolumeOfEsriTIN (Input ESRI TIN, Out File , Level)

ETS_GPVolumeOfPolygonZTIN (Input PolygonZ TIN, Out File,Level)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

VolumeOfEsriTin (inTin As ITin, sOutTextFile As String, dLevel As Double) As Boolean

VolumeOfPolygonZTin (inPolygonZTin As IFeatureClass, sOutFName As String, dLevel As Double) As Boolean

# Volume of Polygons

ToolBox Implementation .NET Implementation

Calculates the Volume and 3D Area of the polygons from the input polygon dataset based on user assigned level or different levels for each polygon from a field in the attribute table and the surface defined by an ESRI TIN or PolygonZ TIN. The Volume calculated for each polygon is split in two portions - above and below the surface.

**Inputs:**

- A polygon feature class
- An ESRI TIN or PolygonZ TIN
- Level for all polygons or field from the attribute table containing the level information.

**Outputs:**

- A polygon feature class with several new fields added
  - ET_VAbove - Volume of the polygon above the surface
  - ET_VBelow - Volume of the polygon below the surface
  - ET_Area3D - 3D Area of the polygon
  - ET_Area - 2D Area of the polygon
- All attributes of the original polygons are preserved.

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPVolumeOfPolygonsEsriTIN <Input Dataset> <Input ESRI TIN> <Out Feature Class> {Level Field} {Level}

ETS_GPVolumeOfPolygonsPolygonZTIN <Input Dataset> <Input PolygonZ TIN> <Out Feature Class> {Level Field} {Level}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polygon feature layer or feature class |
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| {Level Field} | A String indicating the field name of the input polygon attribute table to be used for levels of the polygons |
| {Level} | A Double indicating the level for all input polygons |

**Scripting syntax**

ETS_GPVolumeOfPolygonsEsriTIN (Input Dataset, Input ESRI TIN, Out Feature Class, Level Field)

ETS_GPVolumeOfPolygonsPolygonZTIN (Input Dataset, Input PolygonZ TIN, Out Feature Class, Level Field)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**

(Go to TOP)

VolumeOfPolygonsEsriTin (inFeatureClass As IFeatureClass, inTin As ITin, sOutFName As String, dLevel As Double, Optional sLevelField As String = "") As IFeatureClass

VolumeOfPolygonsPolygonZTin (inFeatureClass As IFeatureClass, inPolygonZTin As IFeatureClass, sOutFName As String, dLevel As Double, Optional sLevelField As String = "") As IFeatureClass

# Cut/Fill Analysis

ToolBox Implementation .NET Implementation

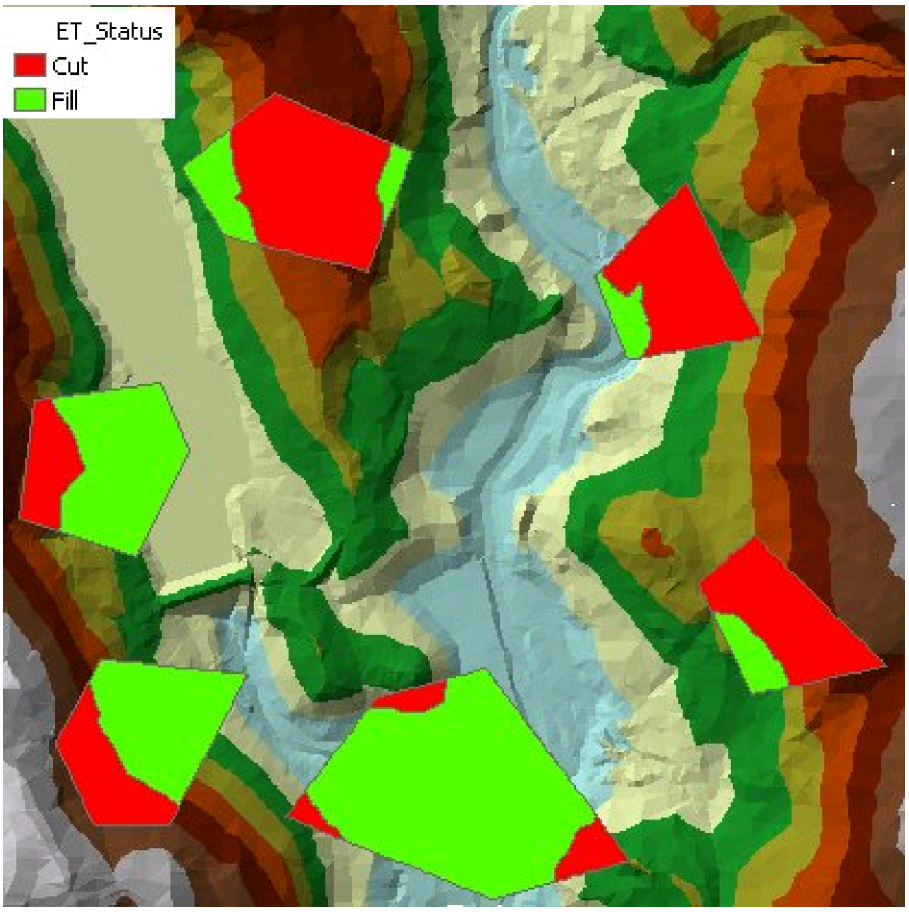
Calculates the cut and fill areas for each polygon from a polygon feature class based on user assigned level or different levels for each polygon from a field in the attribute table and the surface defined by an ESRI TIN or PolygonZ TIN. If a single polygon has both cut and fill areas it is split into one or more polygons. The Volume and 3D area are calculated for each resulting polygon.

**Inputs:**

- A polygon feature class
- An ESRI TIN or PolygonZ TIN
- Level for all polygons or field from the attribute table containing the level information.

**Outputs:**

- A polygon feature class with several new fields added
  - ET_ID - contains the FID of the original polygon
  - ET_Status - the classification of the polygon based on its Cut/Fill status
  - ET_Cut - the volume of the polygon with a Status = Cut
  - ET_Fill - the volume of a polygon with Status = Fill
  - ET_Area3D - the 3D area of the polygon
  - ET_Area - the 2D Area of the polygon
- All attributes of the original polygons are preserved.




**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPCutFillEsriTIN <Input Dataset> <Input TIN>  <Out Feature Class> {Level Field} {Level}

ETS_GPCutFillPolygonZTIN <Input Dataset> <Input TIN>  <Out Feature Class>  {Level Field} {Level}

Parameters

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polygon feature layer or feature class |
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| {Level Field} | A String indicating the field name of the input polygon attribute table to be used for levels of the polygons |
| {Level} | A Double indicating the level for all input polygons |

Scripting syntax

ETS_GPCutFillEsriTIN (Input Dataset, Input ESRI TIN, Out Feature Class, Level Field)

ETS_GPCutFillPolygonZTIN (Input Dataset, Input PolygonZ TIN, Out Feature Class, Level Field)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CutFillEsriTin(inFeatureClass As IFeatureClass, inTin As ITin, sOutFName As String, dLevel As Double, Optional sLevelField As String = "") As IFeatureClass

CutFillPolygonZTin(inFeatureClass As IFeatureClass, inPolygonZTin As IFeatureClass, sOutFName As String, dLevel As Double, Optional sLevelField As String = "") As IFeatureClass

# Visibility Analysis

ToolBox Implementation .NET Implementation

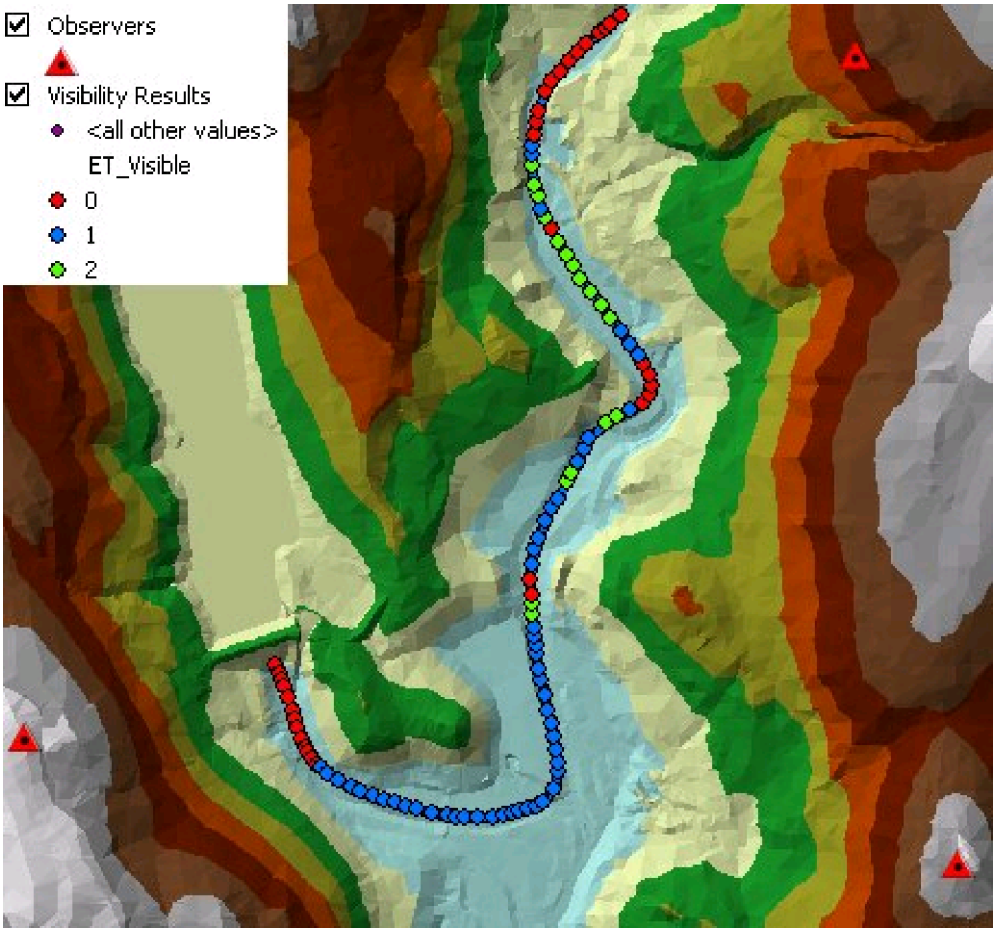
Calculates the visibility from a set of observer points to the features of the target dataset. If the target dataset is of polyline or polygon type, the visibility to the vertices of the geometries is calculated. The result will be a point dataset (vertices of the targets). For each point the number of observers that can see the point are recorded in the attribute table.

**Inputs:**

- A point dataset (Observers). The attribute table can have a numeric field which values will indicate the offset of the observers above the terrain.
- A point, polyline or polygon dataset (Targets). The attribute table can have a numeric field which values will indicate the offset of the targets above the terrain.
- A surface - Raster, ESRI TIN or PolygonZ TIN
- Options (see Line of Sight discussion)
    - Apply Earth curvature correction
    - Apply air refraction corrections
    - Apply radio wave corrections.

**Outputs:**

- A point feature class
    - ET_Visible - the number of observers that can see the point
- All attributes of the original polygons are preserved.




Notes:

- The function is resources hungry and time consuming. Do not use it on large datasets. Recommended is O x T < 500 where

- O = Number Observers
- T = Number Target Points

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPVisibilityEsriTIN <Observers Dataset> <Target Dataset> <Input ESRI TIN> <Out Feature Class> <Observer Offset Field> <Target Offset Field> {Use Earth Curvature} {Refraction Correction} {Radio Waves Correction} {Cutoff Distance}

ETS_GPVisibilityPolygonZTIN <Observers Dataset> <Target Dataset> <Input PolygonZ TIN> <Out Feature Class> <Observer Offset Field> <Target Offset Field> {Use Earth Curvature} {Refraction Correction} {Radio Waves Correction} {Cutoff Distance}

**Parameters**

| Expression | Explanation |
|---|---|
| <Observers Dataset> | A Point feature layer or feature class |
| <Target Dataset> | A Point or Polyline feature layer or feature class |
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| <Observer Offset Field> | A String representing the name of the field which values are going to be used as offset of the observers above the raster. |
| <Target Offset Field> | A String representing the name of the field which values are going to be used as offset of the target above the raster. |
| {Use Earth Curvature} | A Boolean |
| {Refraction Correction} | A Double representing the air refraction coefficient - Default value = 0.13 |
| {Radio Waves Correction} | A Double representing the radio waves correction coefficient - Default value = 1.333333 |
| {Cutoff Distance} | A Double representing the Cutoff distance. |

**Scripting syntax**

ETS_GPVisibilityEsriTIN (Observers Dataset, Target Dataset, Input ESRI TIN, Out Feature Class, Observer Offset Field, Target Offset Field, Use Earth Curvature, Refraction Correction, Radio Waves Correction, Cutoff Distance)

ETS_GPVisibilityPolygonZTIN (Observers Dataset, Target Dataset, Input PolygonZ TIN, Out Feature Class, Observer Offset Field, Target Offset Field, Use Earth Curvature, Refraction Correction, Radio Waves Correction, Cutoff Distance)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

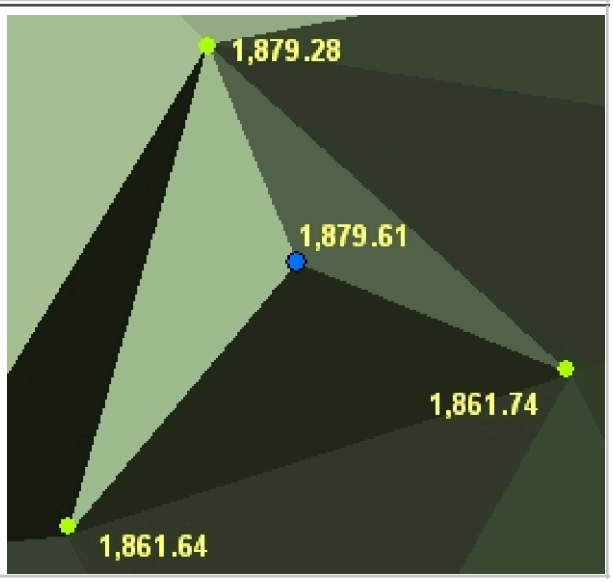
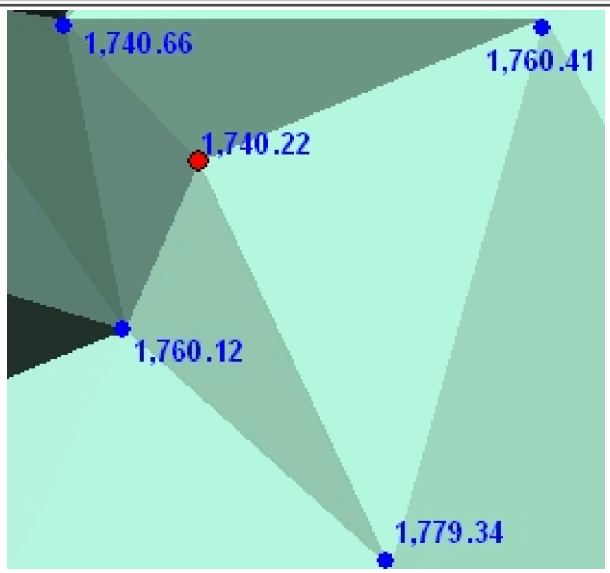
VisibilityEsriTin (inTin As ITin, observerFeatures As IFeatureClass, targetFeatures As IFeatureClass, sOutFName As String, observerOffsetField As String, targetOffsetField As String, bCurvature As Boolean, Optional dRefraction As Double = 0, Optional dRadio As Double = 1) As IFeatureClass

VisibilityPolygonZTin (inPolygonZTin As IFeatureClass, observerFeatures As IFeatureClass, targetFeatures As IFeatureClass, sOutFName As String, observerOffsetField As String, targetOffsetField As String, bCurvature As Boolean, Optional dRefraction As Double = 0, Optional dRadio As Double = 1) As IFeatureClass

# Identify Sinks and Peaks from a TIN

ToolBox Implementation .NET Implementation

Analyses the triangles of the input TIN, finds the nodes that represent sinks (pits) or peaks. A TIN node participates in several triangles, each triangle is created from 3 nodes.

| If a node has maximum value in all triangles in which it participates it represents a Peak. | If a node has minimum value in all triangles in which it participates it represents a Sink or Pit. |
|---|---|
|  |  |



While the peaks in most of the cases represent natural features, the sinks with some exceptions (natural lakes and depression) are caused by incorrect data or interpolation.

**Inputs:**

- An ESRI TIN or PolygonZ TIN
- Output feature class
- Option to include/exclude in the analysis the flat triangles. Since all nodes in a flat triangle have the same Z value, if included in the analysis they in most of the cases will be either peaks or sinks.

**Outputs:**

- New Point or PointZ feature class.
- A new field is added to the point attribute table.
  - ET_Type - the type of the node - Peak or Sink

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPPeaksAndSinksEsriTIN <Input ESRI TIN> <Out Feature Class> {3D Features} {Ignore Flat Triangles}

ETS_GPPeaksAndSinksPolygonZTIN <Input PolygonZ TIN> <Out Feature Class> {3D Features} {Ignore Flat Triangles}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input ESRI TIN> | An ESRI TIN layer or dataset |

| | |
|---|---|
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| {3D Features} | A Boolean indicating whether the result will be PointZ or Point |
| {Ignore Flat Triangles} | A Boolean indicating whether the flat triangles to be included in the analysis. |

**Scripting syntax**

ETS_GPPeaksAndSinksEsriTIN (Input ESRI TIN, Out Feature Class, 3D Features,  Ignore Flat Triangles)

ETS_GPPeaksAndSinksPolygonZTIN (Input PolygonZ TIN, Out Feature Class, 3D Features  Ignore Flat Triangles)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

PeaksAndSinksFromESRITin (inTin As ITin, sOutFName As String, Optional bZ As Boolean = False, Optional bNoFlat As Boolean = False) As IFeatureClass

PeaksAndSinksFromPolygonZTin (polygonZTin As IFeatureClass, sOutFName As String, Optional bZ As Boolean = False, _
Optional bNoFlat As Boolean = False) As IFeatureClass

# Raster Slope

ToolBox Implementation

.NET Implementation

Calculates the slope of each cell of the input raster in percent or degrees. The result added to ArcMap is categorized in 8 groups.

Read more about Slope and its importance here

**Inputs:**

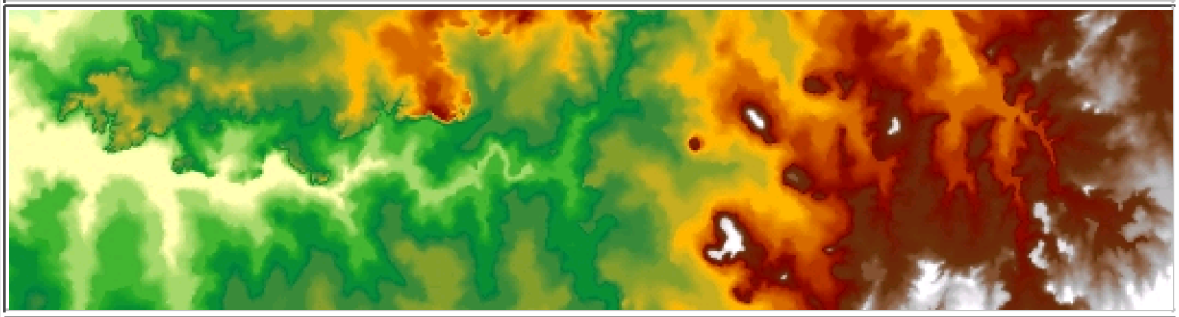
- Input raster dataset
- Output raster name and format
- The unit of the output slope - Degrees or Percentage (rise/run)

**Output:**

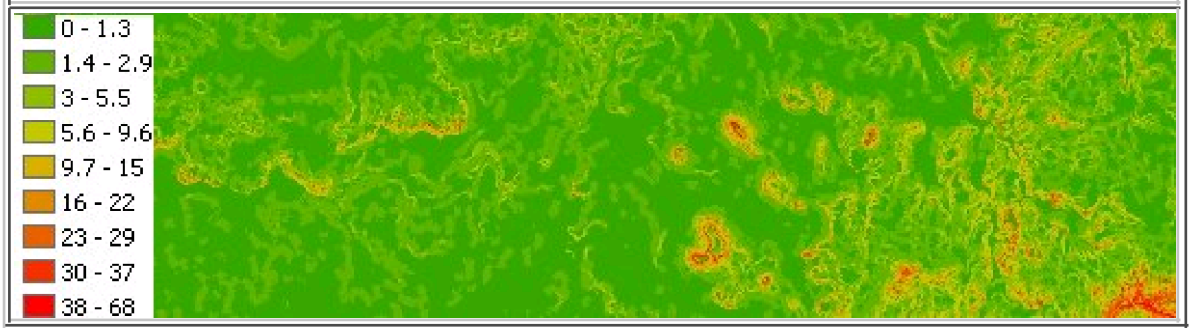
- A floating point raster.

**Example:**

Source Elevation raster



Result Slope raster (Degrees option used)




**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPRasterSlope <Input Raster> <Out Raster> <Slope Units>

**Parameters**

| Expression | Explanation |
| --- | --- |

| | |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Slope Units> | A String indicating the slope units. Valid inputs "Degrees" and "Percent" |

**Scripting syntax**

ETS_GPRasterSlope (Input Raster,  Out_Raster, Slope Units)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

RasterSlope (inRasterDataset As IRasterDataset2, sOutRaster As String, bDegrees As Boolean) As IRasterDataset2

# Raster Aspect

ToolBox Implementation .NET Implementation

Calculates the aspect (the direction in which slope faces) of each cell of the input raster in degrees. The result added to ArcMap is categorized in 9 groups.

- N - North ( 0 to 22.5 and 337.5 to 360)
- NE - North East (22,5 to 67.5)
- E - East (67.5 to 112.5)
- SE - South East (112.5 to 157.5)
- S - South (67.5 to 112.5)
- SW - South West (202.5 to 247.5)
- W - West (247.5 to 292.5)
- NW - North West (292.5 to 337.5)
- U - Undefined - Slope = 0

Read more about Aspect and its importance here.

**Inputs:**

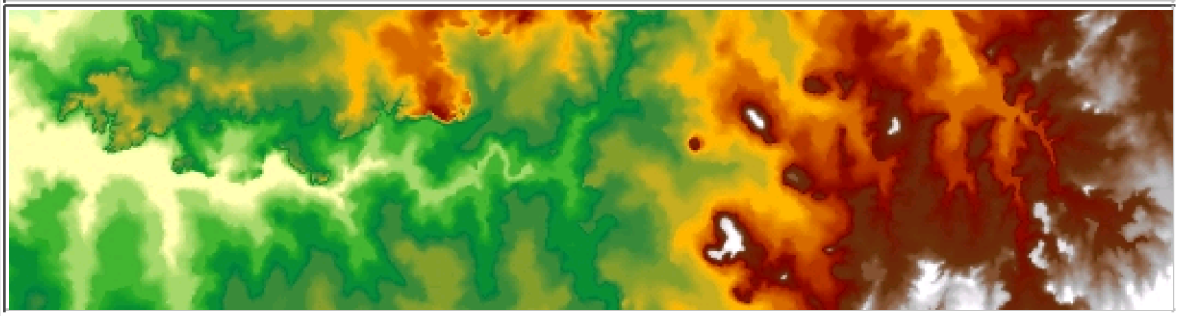
- Input raster dataset
- Output raster name and format

**Output:**

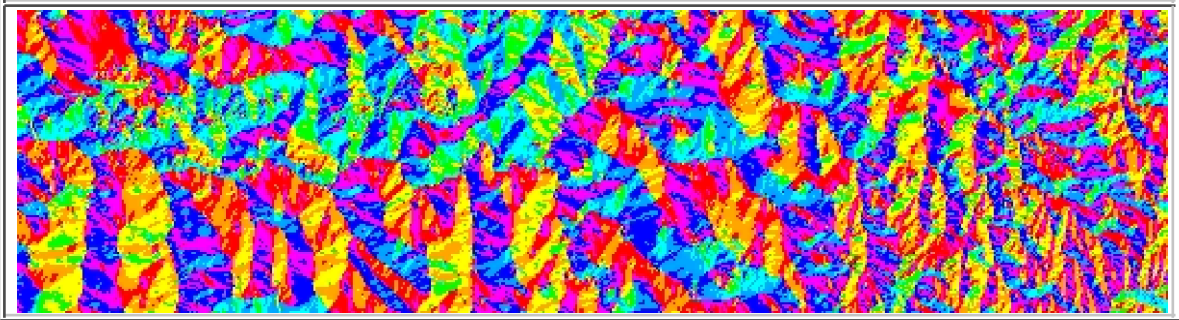
- A floating point raster.

**Example:**

Source Elevation raster



Result Aspect raster



**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.

    - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
    - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPRasterAspect <Input Raster>  <Out Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Scripting syntax**

ETS_GPRasterAspect (Input Raster,  Out Raster)

**.NET implementation**
(Go to TOP)

RasterAspect (inRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Raster Hillshade

ToolBox Implementation .NET Implementation

Calculates the shading of a terrain raster based on user defined position of the light source (The Sun). The position of the light source is defined with Azimuth (0 to 360) and Altitude (0 to 90). The combination of the Hillshade overlaid with a semi-transparent elevation raster gives more realistic look of the terrain (see example below).

Read more about Hillshade and its importance here.

**Inputs:**

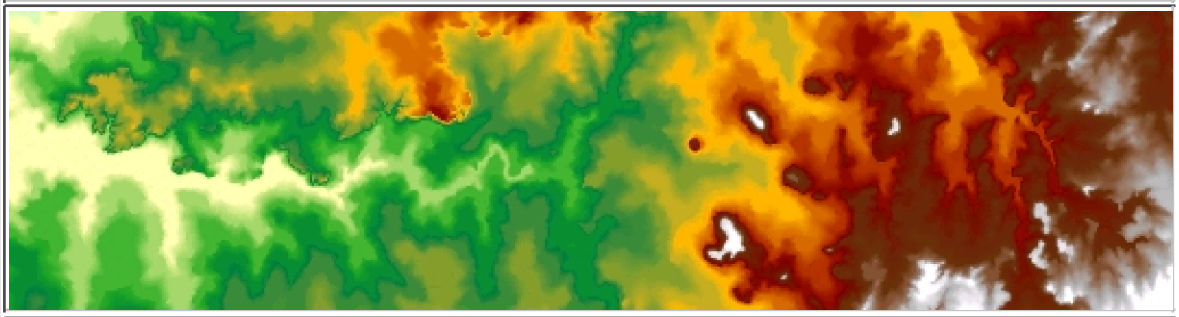
- Input raster dataset
- Output raster name and format
- Light azimuth (0 to 360)
- Light altitude(0 to 90)

**Output:**

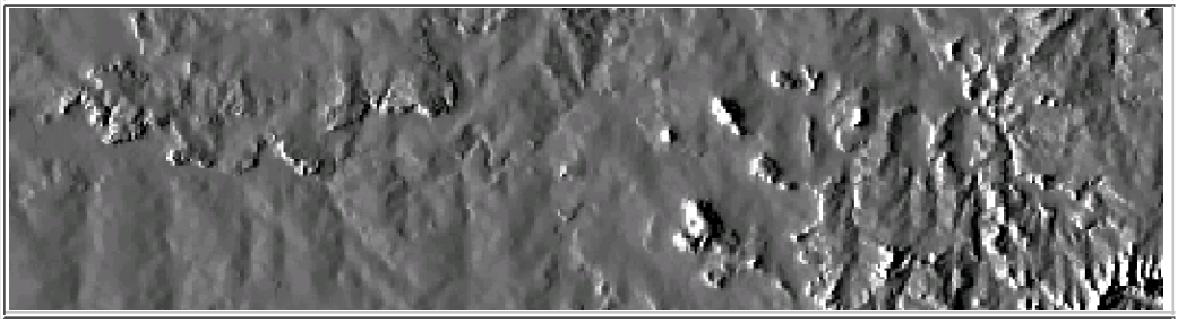
- A floating point raster.

**Example:**

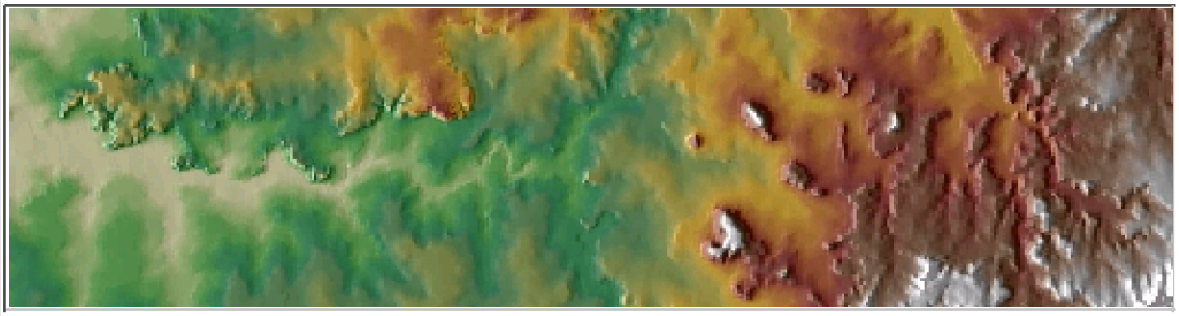
Source elevation raster



Result Hillshade raster



Elevation raster (50% transparency) over the Hillshade raster



**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats

(ESRI GRID, Erdas Imagine and TIFF).

- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPRasterHillshade <Input Raster>  <Out Raster> {Light Azimuth} {Light Altitude}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| {Light Azimuth} | A Double representing azimuth of the light source (0 to 360). 0 indicates North, 90 - East, 180 - South, 270 - West (default is 315) |
| {Light Altitude} | A Double representing the altitude of the light source  in degrees (0 to 90) (default is 45) |

**Scripting syntax**

ETS_GPRasterHillshade  (Input Raster,  Out_Raster, Light Azimuth, Light Altitude)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

RasterHillshade (inRasterDataset As IRasterDataset2, sOutRaster As String, Optional dAzimuth As Double = 315, Optional dAltitude As Double = 45) As IRasterDataset2

# Contours from Raster

ToolBox Implementation .NET Implementation

Interpolates contours from a raster. The contours can be created as PolylineZs with constant Z values representing the elevation.

**Inputs**:

- A raster dataset.
- Base value - the contour from which to begin generation of contours.
- Contour interval - Z value difference between adjacent contours in map units.
- Option for interpolating 3D contours.
- Option for smoothing the contours.

**Outputs:**

- New polyline feature class. If the option for creating 3D contours is used the output will be PolylineZ feature class.
- A new field ET_Height is added to the attribute table. The values of this field contain the Z value for each contour.

**Notes:**

- The smooth polylines option might be time consuming.
- The input raster must have a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPContoursFromRaster <Input Raster>  <Out Feature Class> <Base Contour> <Contour Interval> {3D Contours}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Feature Class> | A String - the full name of the output feature class. |
| <Base Contour> | A Double representing the Z level of the contour from which to begin generation of contours. |
| <Contour Interval> | A Double representing the contour interval |
| {3D Contours} | A Boolean indicating whether the output to be Polyline or PolylineZ feature class |

**Scripting syntax**

ETS_GPContoursFromRaster (Input Raster, Out Feature Class , Base Contour, Contour Interval, 3D Contours)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

InterpolateContoursRaster (inRasterDataset As IRasterDataset2, sOutFName As String, baseC As Double, contInt As Double, Optional bZ As Boolean = False) As IFeatureClass

# Raster Viewshed

ToolBox Implementation .NET Implementation

Calculates the visibility from a set of observer points to each of the cells of the output raster based on the input surface raster.

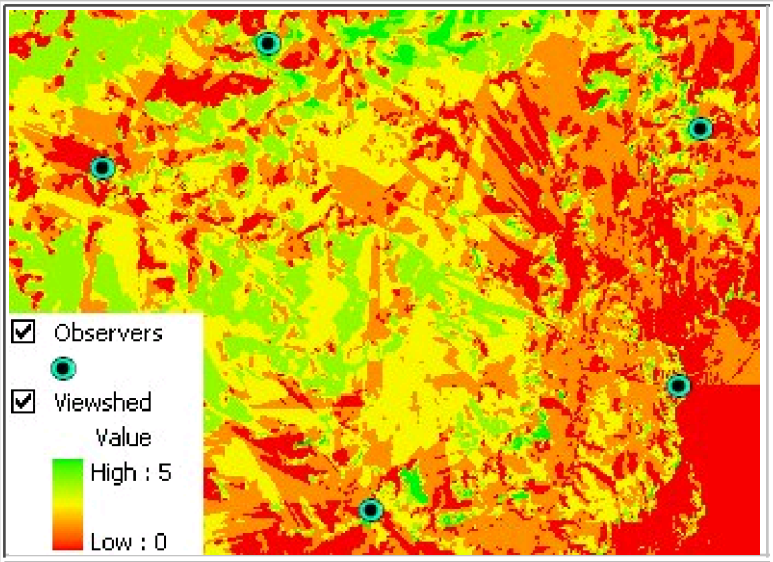
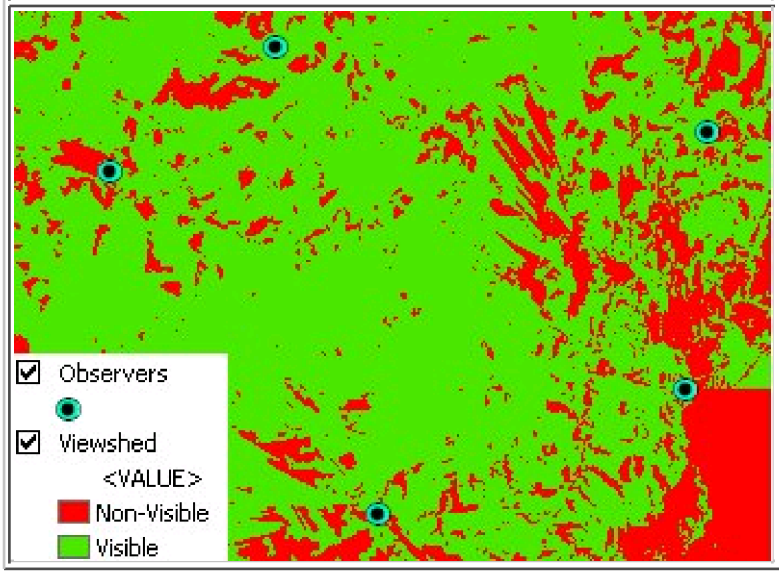
**Inputs:**

- A point dataset (Observers). The attribute table can have a numeric field which values will indicate the offset of the observers above the terrain.
- A raster dataset representing the surface for which the analysis will be performed.
- A field in the point feature class that will indicate the offset of the observer above the surface.
- A number that will indicate the offset of the target above the surface.
- Cutoff distance (optional) - how far an observer can see
- Options (see Line of Sight discussion)
  - Apply Earth curvature correction
  - Apply air refraction corrections
  - Apply radio wave corrections.

**Outputs:**

- An integer raster. Each cell will have as a value the number of observers from which this cell is visible.

**Example:**

| | |
|---|---|
|  | Viewshed from 5 observers symbolized with unique value - the number of observers that can see each cell. |
|  | Viewshed from 5 observers classified in 2 classes:<br>• 0 - Non-visible.<br>• >0 - Visible |

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input point feature class and raster must be in the same projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPViewshed <Observers> <Input Raster> <Out Raster> <Observer Offset Field> <Target Offset> {Use Earth Curvature} {Refraction Correction} {Radio Waves Correction} {Cutoff Distance}

**Parameters**

| Expression | Explanation |
|---|---|
| <Observers> | A Point layer or feature class |
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Observer Offset Field> | A String representing the name of the field which values are going to be used as offset of the observers above the raster. |
| <Target Offset> | A Double indicating the offset of the target above the surface. |
| {Use Earth Curvature} | A Boolean |
| {Refraction Correction} | A Double representing the air refraction coefficient - Default value = 0.13 |
| {Radio Waves Correction} | A Double representing the radio waves correction coefficient - Default value = 1.333333 |
| {Cutoff Distance} | A Double representing the Cutoff distance. |

**Scripting syntax**

ETS_GPViewshed (Observers, Input Raster, Out Raster, Observer Offset Field, Target Offset, Use Earth Curvature, Refraction Correction, Radio Waves Correction, Cutoff Distance)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ViewshedRaster (inFeatureClass As IFeatureClass, inRasterDataset As IRasterDataset2, sOutRaster As String, observerOffsetField As String, targetOffset As Double, Optional bCurv As Boolean = False, Optional dRefraction As Double = 0.13, Optional dRadio As Double = 1.333333, Optional dCutOff As Double = 0) As IRasterDataset2

# Cut/Fill Analysis

ToolBox Implementation .NET Implementation

Calculates the cut and fill areas for each polygon from a polygon feature class based on levels for each polygon from a field in the attribute table and the surface defined by a raster.

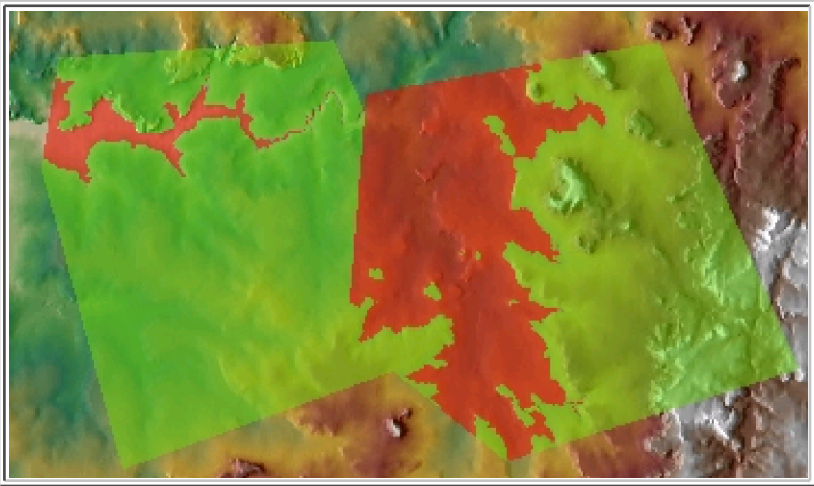
**Inputs:**

- A polygon feature class
- A raster dataset representing the surface for which the analysis will be performed.
- A field from the attribute table containing the level information for each polygon.

**Outputs:**

- An integer raster with values assigned to each zone derived from the input surface raster and the polygons at their levels
- The raster attribute table will contain the following fields
  - Value - the zone ID
  - ET_Volume - the volume of the zone in the units of the spatial reference of the input raster.
    - Negative values of the volume indicate areas where the Z values of the raster surface is above the level of the polygons - CUT.
    - Positive values of the volume indicate areas where the Z values of the surface is below the levels of the polygons. - FILL
  - ET_Area - the area of the zone in the units of the spatial reference of the input raster
  - ET_Status - Cut or Fill

**Example:**



The result of the Cut/Fill function overlaid with the elevation raster.

- red areas indicate FILL
- green areas indicate CUT

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class and raster must have the same projected coordinate system.
- If the input polygon feature class has overlapping polygons, the maximum level in the area of overlap will be used.
- The areas not covered by polygons will have assigned NODATA values
- The polygons or parts of them that are outside of the extents of the input raster will be ignored.
- The cells of the input raster that have the same value as the levels of the polygons will have assigned NODATA values

- The result raster can be easily converted to a polygon feature class using the standard ArcGIS Raster To Polygon tool
- The attributes can be transferred to the polygons by joining the Raster Attribute Table to the polygons using GRID_CODE field of the feature class and the Value field of the raster attribute table.

**ToolBox implementation**

**Command line syntax**

ETS_GPRasterCutFill <Input Polygons> <Input Raster>  <Out Raster> <Level Field>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Polygons> | A Polygon layer or feature class |
| <Input Raster> | A  Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Level Field> | A String representing the name of the field which values are going to be used as levels of the polygons. |

**Scripting syntax**

ETS_GPRasterCutFill (Input Polygons, Input Raster,  Out Raster, Level Field)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

RasterCutFill (inFeatureClass As IFeatureClass, inRasterDataset As IRasterDataset2, sOutRaster As String, sElField As String) As IRasterDataset2

# Raster Volume

ToolBox Implementation .NET Implementation

Calculates the Volume and Area of a raster surface above and below a horizontal reference plane defined by the user specified level.

**Inputs:**

- A raster dataset representing the surface for which the analysis will be performed.
- The level (Z value) of the reference plane or Overlay raster which will be used for the volume calculations
- Volume Option
  - Below
  - Above
  - Both

**Outputs:**

- An integer raster with values assigned to each zone derived from the input surface raster and the level of the reference plane.
- The raster attribute table will contain the following fields
  - Value - the zone ID
  - ET_Volume - the volume of the zone in the units of the spatial reference of the input raster.
    - Negative values of the volume indicate areas where the Z values of the raster surface is above the level of the polygons - CUT.
    - Positive values of the volume indicate areas where the Z values of the surface is below the levels of the polygons. - FILL
  - ET_Area - the area of the zone in the units of the spatial reference of the input raster
  - ET_Status - Cut or Fill

**Example:**

Volume for a dam calculated for three different levels - 1350, 1355 and 1360 meters. The option to calculate the volume only below the reference plane used. The result rasters converted to polygons. The polygons downstream from the dam wall removed.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.
- If an Overlay Raster is used it must have the same spatial reference as the Base Raster
- The cells of the input raster that have the same value as the level of the plane will have assigned NODATA values
- The result raster can be easily converted to a polygon feature class using the standard ArcGIS Raster To Polygon tool
- The attributes can be transferred to the polygons by joining the Raster Attribute Table to the polygons using GRID_CODE field of the feature class and the Value field of the raster attribute table.

**ToolBox implementation**

**Command line syntax**

ETS_GPRasterVolume <Input Raster>  <Out Raster> <Level> <Volume Option>

ETS_GPVolumeBetweenRasters <Input Raster>  <Out Raster> <Second Raster> <Volume Option>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Second Raster> | A Raster Dataset for the overlay raster for volume calculation |
| <Level> | A Double representing the Z level of the reference plane |
| <Volume Option> | A String representing the volume option. Valid values<br>- Below<br>- Above<br>- Below & Above |

**Scripting syntax**

ETS_GPRasterVolume (Input Raster,  Out Raster, Level, Volume Option)

ETS_GPVolumeBetweenRasters (Input Raster,  Out Raster, Second Raster, Volume Option)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

VolumeOfRaster (inRasterDataset As IRasterDataset2, sOutRaster As String, dLevel As Double, sOption As String) As IRasterDataset2

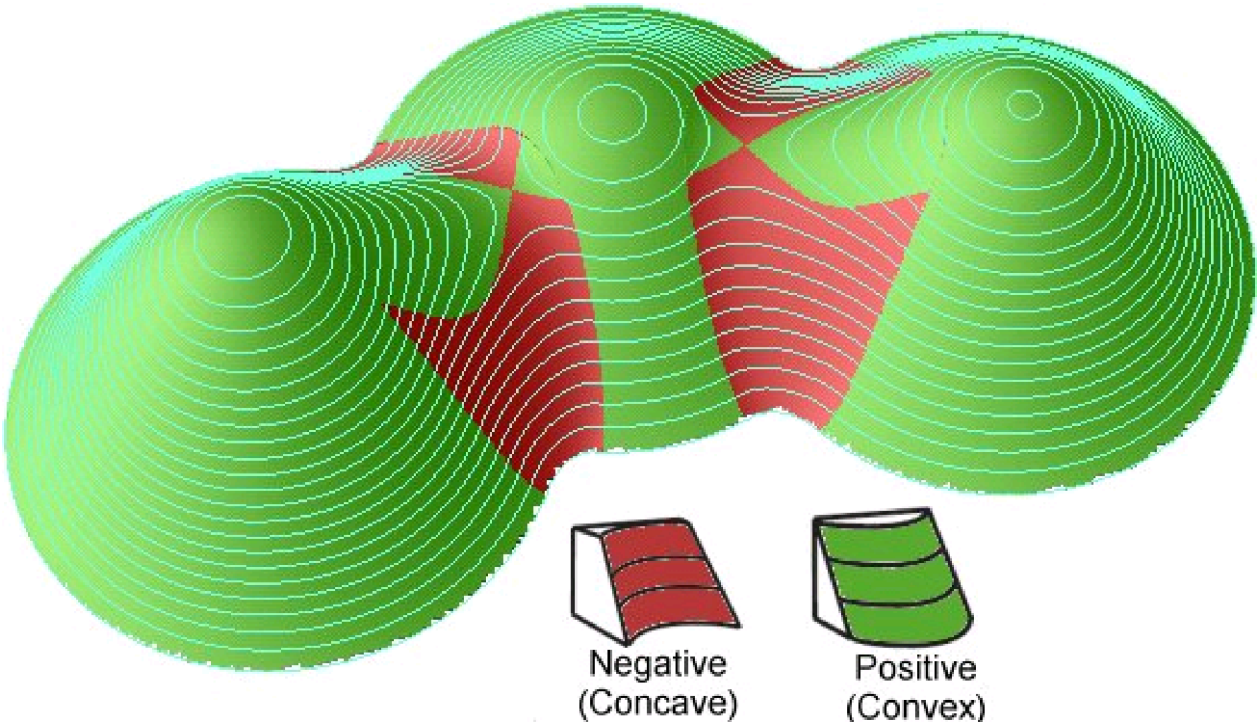
# Raster Curvature

ToolBox Implementation .NET Implementation

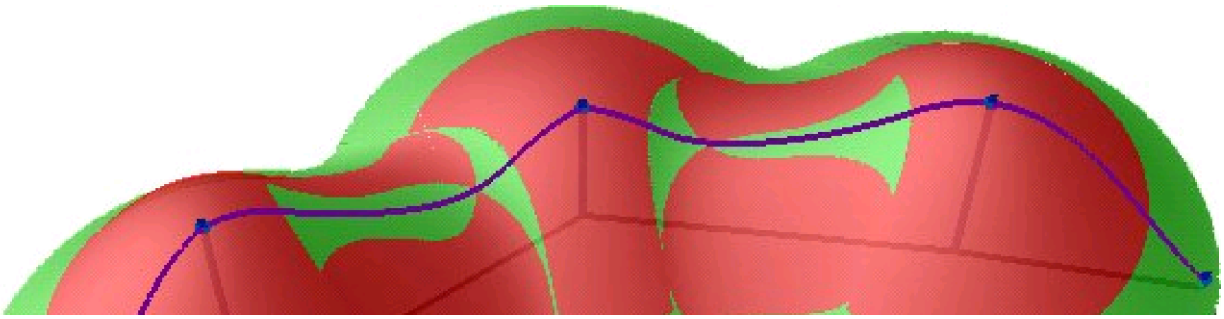
Surface curvature at a point is the curvature of a line formed by the intersection of the surface with a plane with a specific orientation passing through this point. The value of the curvature is reciprocal of the radius of the curve - the larger the radius, the smaller the curvature value (a gentle curve has small curvature and a tight curve has large curvature value). The units of the curvature are radians per linear unit (the unit of the spatial reference of the raster).
Because the values of the curvature are typically small, the results of the curvature functions of ET Surface are the actual curvature multiplied by 100. So the value of the curvature can be described as the change of the orientation resulting from travelling one hundred linear units along the respective line.
The sign of the curvature is assigned differently by different authors. To keep consistency with ArcGIS the functions of ET Surface assign the sign as the corresponding functions of ArcGIS. If you want to change the sign you can use the Raster Calculator to multiply the result by -1.

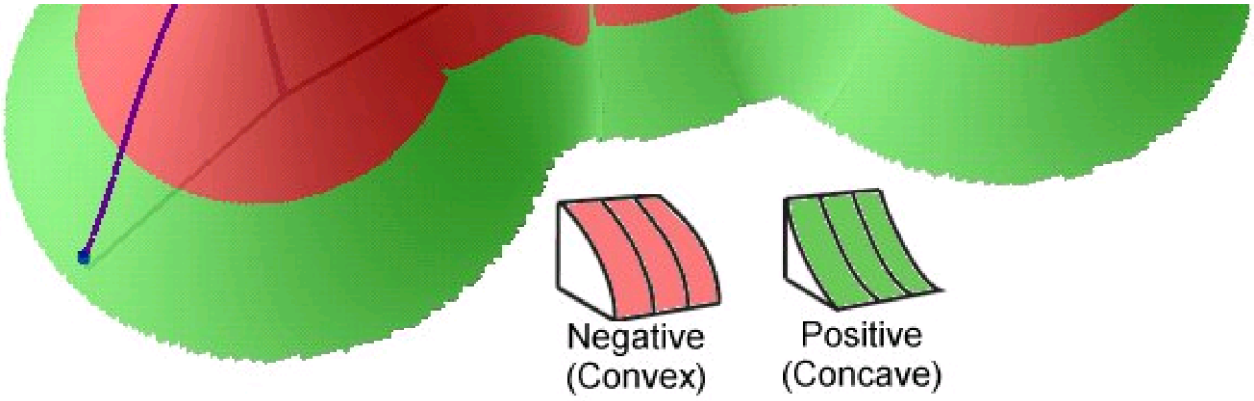
The functions of ET Surface calculate the curvatures for each cell of a raster dataset using its immediate neighbors as devised by Zevenbergen and Thorne (1987).

**Plan Curvature** is the curvature in a horizontal plane. It can be also described as the curvature of the hypothetical contour line that passes through a specific cell. The Plan Curvature is positive for cells with concave contours and negative for cells with convex contours. Plan curvature can be used to differentiate between ridges and valleys.

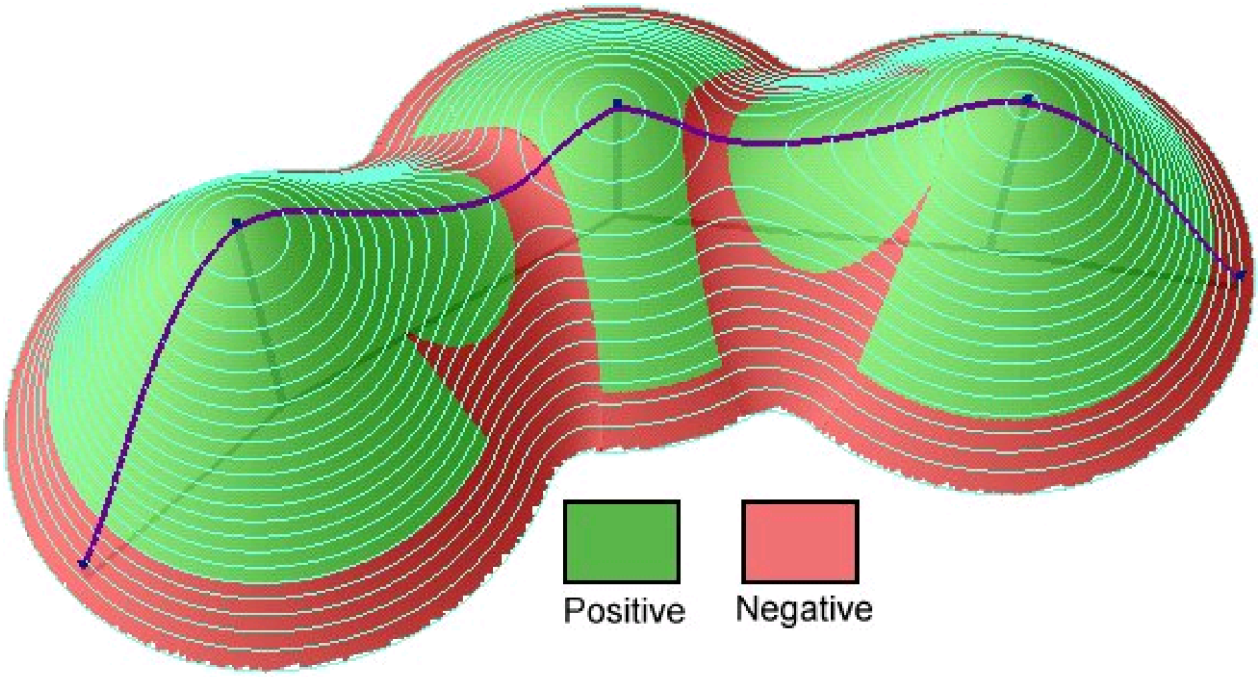



**Profile Curvature** is the curvature of the surface in the direction of the steepest slope (in the vertical plane of a flow line). The Profile Curvature affects the flow velocity of water draining the surface and influences erosion and deposition. In locations with convex(negative) Profile Curvature the erosion will prevail and in locations with concave (positive) curvature the deposition.






**General Curvature** (also called Total) is the curvature of the surface itself (not the curvature of a line formed by the intersection of the surface with a plane). The General Curvature can be positive or convex (indicating peaks), negative or concave (indicating valleys) or zero (indicating flat surface or a saddle).




**Inputs:**

- Input raster dataset
- Output raster name and format

**Output:**

- A floating point raster.

**Example:**

DEM (50% transparency) over hillshade

DEM (50% transparency) over Plan Curvature (20% transparency) over hillshade. Pronounced valleys and ridges.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPPlanCurvature <Input Raster> <Out Raster>

ETS_GPProfileCurvature <Input Raster> <Out Raster>

ETS_GPGeneralCurvature <Input Raster> <Out Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Scripting syntax**

ETS_GPPlanCurvature (Input Raster, Out_Raster)

ETS_GPProfileCurvature (Input Raster, Out_Raster)

ETS_GPGeneralCurvature (Input Raster, Out_Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

PlanCurvature (inRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

ProfileCurvature (inRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

GeneralCurvature (inRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

References:

Zevenbergen, L.W. and Thorne, C.R. (1987) Quantitative analysis of land surface topography. Earth Surface Processes and Landforms.

Wilson, J.P., and Gallant, J.C. editors, 2000, Terrain Analysis: Principles and Applications (Chichester: Wiley).

# Euclidean Distance

ToolBox Implementation .NET Implementation

Calculates for each cell of the output raster the Euclidean (straight line) distance to the closest point (source) of the input feature class. If the input feature class is of Polyline or Polygon type, the vertices will be used as sources.

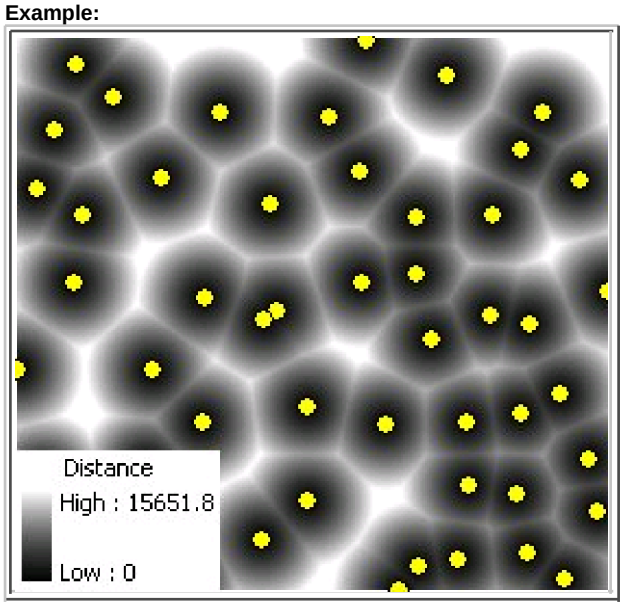
**Inputs:**

- A point, polyline or polygon dataset (Sources).
- The cell size of the output raster.

**Outputs:**

- A floating point raster. Each cell will have as a value the distance to the closest input point (Source). The extent of the output is equal to the extent of the input feature class.

**Example:**




**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input point feature class must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPEUCDistance <Input Features> <Out Raster> < Cell Size>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Features> | A Point, Polyline or Polygon feature layer or feature class |

| | |
|---|---|
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Cell Size> | A Double representing the cell size of the output raster. |

**Scripting syntax**

ETS_GPEUCDistance (Input Features, Out Raster, ID Field, Weight Field, Cell Size)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

EucledeanDistance (inFeatureClass As IFeatureClass, sOutRaster As String, dCellSize As Double) As IRasterDataset2

# Euclidean Direction

ToolBox Implementation

.NET Implementation

Calculates for each cell of the output raster the direction to the closest point (source) of the input feature class. If the input feature class is of Polyline or Polygon type, the vertices will be used as sources. The result added to ArcMap is categorized in 9 groups.

- N - North ( 0 to 22.5 and 337.5 to 360)
- NE - North East (22,5 to 67.5)
- E - East (67.5 to 112.5)
- SE - South East (112.5 to 157.5)
- S - South (67.5 to 112.5)
- SW - South West (202.5 to 247.5)
- W - West (247.5 to 292.5)
- NW - North West (292.5 to 337.5)
- U - Undefined - Slope = 0

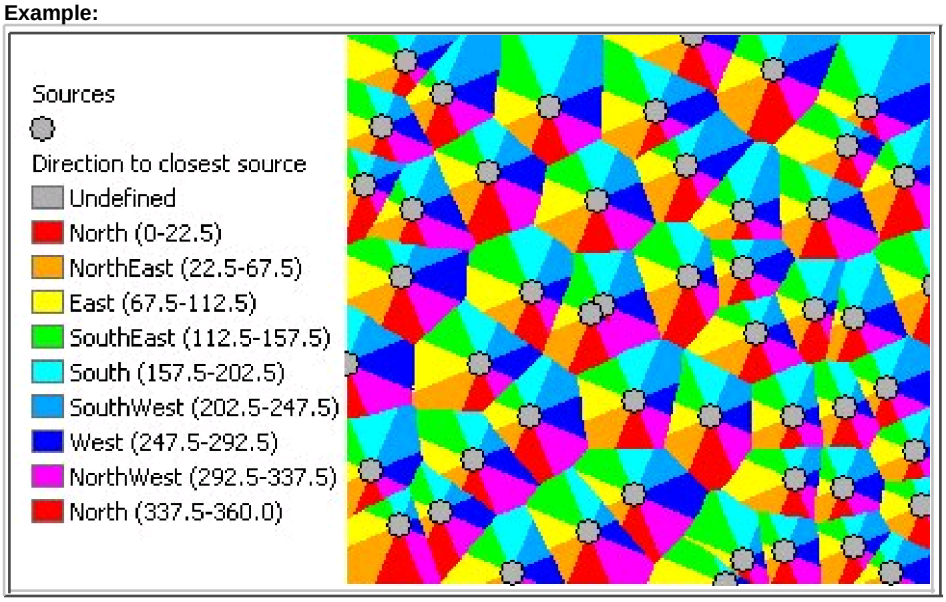
**Inputs:**

- A point, polyline or polygon dataset (Sources).
- The cell size of the output raster.

**Outputs:**

- A floating point raster. Each cell will have as a value the direction in degrees to the closest input point (Source). The extent of the output is equal to the extent of the input feature class.

**Example:**




**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input point feature class must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPEUCDirection <Input Features> <Out Raster> < Cell Size>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Features> | A Point, Polyline or Polygon feature layer or feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Cell Size> | A Double representing the cell size of the output raster. |

**Scripting syntax**

ETS_GPEUCDirection (Input Features, Out Raster, ID Field, Weight Field, Cell Size)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

EucledeanDirection (inFeatureClass As IFeatureClass, sOutRaster As String, dCellSize As Double) As IRasterDataset2

# Euclidean Allocation / Voronoi Diagram / Thiessen Polygons

ToolBox Implementation .NET Implementation

Derives the catchment areas of the input features (Sources). Each cell of the output raster will have the value of the closest (based on straight line distance) Source.

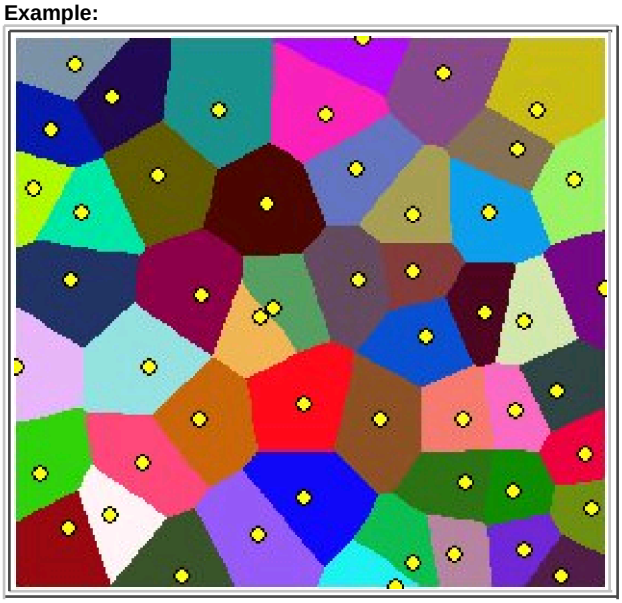
**Inputs:**

- A point, polyline or polygon dataset (Sources).
- The cell size of the output raster.
- Source ID field. The values from this field will be allocated to the cells of the output raster.

**Outputs:**

- An integer raster. Each cell will have as a value the ID of the closest input point (Source). The extent of the output is equal to the extent of the input feature class.

**Example:**



**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class must be in a projected coordinate system.
- The result raster can be easily converted to a polygon feature class using the standard ArcGIS Raster To Polygon tool
- The attributes can be transferred to the polygons by joining the Raster Attribute Table to the polygons using GRID_CODE field of the feature class and the Value field of the raster attribute table.

**ToolBox implementation**

**Command line syntax**

ETS_GPVoronoi <Input Features> <Out Raster> < ID Field> < Cell Size>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Features> | A Point, Polyline or Polygon feature layer or feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| < ID Field> | A String representing the name of the field in the input point feature class to be used as point ID. |
| <Cell Size> | A Double representing the cell size of the output raster. |

**Scripting syntax**

ETS_GPVoronoi (Input Features, Out Raster, ID Field, Weight Field, Cell Size)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

VoronoiAllocation (inFeatureClass As IFeatureClass, sOutRaster As String, sIDField As String, dCellSize As Double) As IRasterDataset2

# Weighted Voronoi Diagram

ToolBox Implementation .NET Implementation

Derives the catchment areas of the input features (Sources) using their spatial location and the weight of each source. The smaller the weight value of a source is, the greater the influence of the source is.

**Inputs:**

- A point feature class (Sources).
- The cell size of the output raster.
- Source ID field. The values from this field will be allocated to the cells of the output raster.
- Weight field. The values of this field will be used to define the weight of the sources. Only integer fields can be used for weight of the sources
- Cutoff distance - a cell farther than this distance of a source cannot be allocated to this source - the maximum radius of influence of a source.

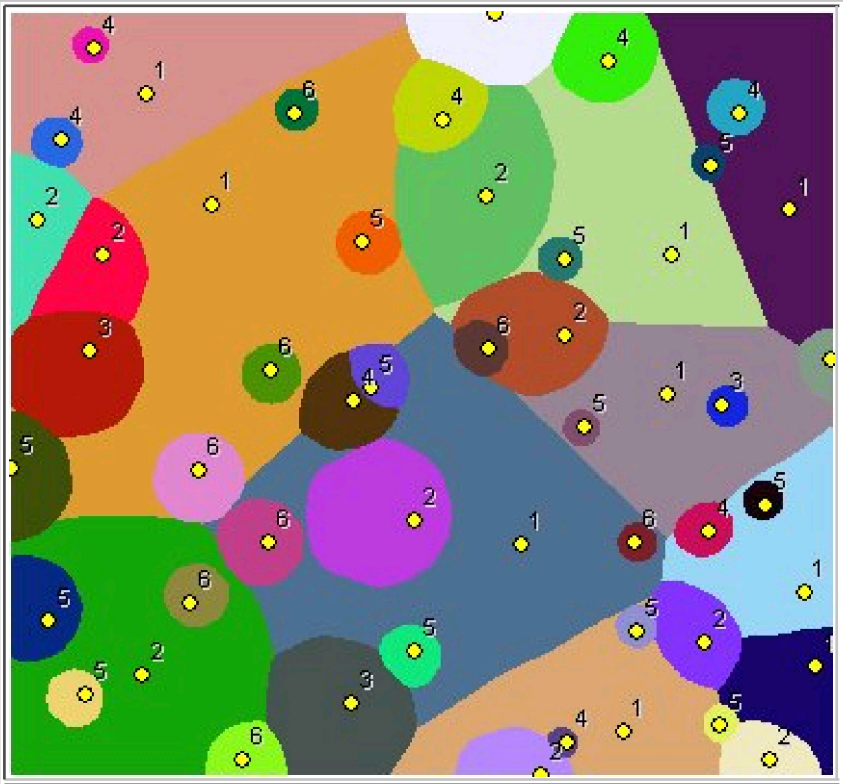
**Algorithm:**

All sources start growing at the same time. A source with weight = 1 increases its area every cycle, a source with weight = 2 every second cycle, a source with weight = 10 every tenth cycle and so on. The process finishes when all cells of the output raster are occupied. If cutoff distance is specified a source will stop increasing its area when it reaches the maximum radius of influence.

**Outputs:**

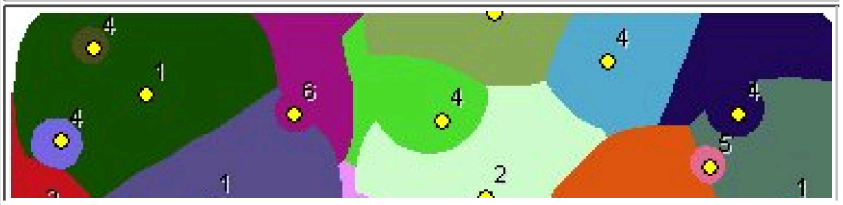
- An integer raster. Each cell will have as a value the ID of the closest input point (Source). The extent of the output is equal to the extent of the input feature class.

**Examples:**



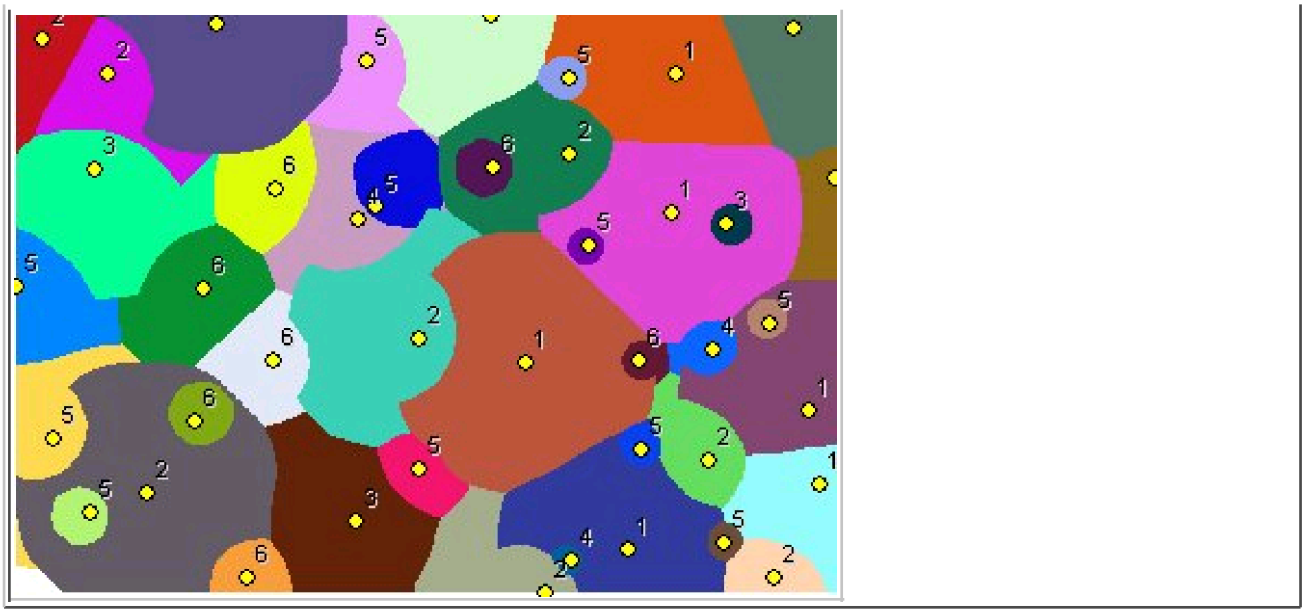
Points labeled with their weights.

No cutoff distance specified



Points labeled with their weights.

Cutoff distance specified



**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class must be in a projected coordinate system.
- The result raster can be easily converted to a polygon feature class using the standard ArcGIS Raster To Polygon tool
- The attributes can be transferred to the polygons by joining the Raster Attribute Table to the polygons using GRID_CODE field of the feature class and the Value field of the raster attribute table.

**ToolBox implementation**

**Command line syntax**

ETS_GPWeightedVoronoi <Input Points> <Out Raster> < ID Field> <Weight Field> < Cell Size> {Cut Off Cost}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Points> | A Point layer feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| < ID Field> | A String representing the name of the field in the input point feature class to be used as point ID. |
| <Weight Field> | A String representing the name of the field in the input point feature class that are going to be used as weights. |
| <Cell Size> | A Double representing the cell size of the output raster. |
| {Cut Off Cost} | A Double representing the cut off cost - the value of the cells with larger than this cost (distance x weight) to reach will be set to NODATA |

**Scripting syntax**

ETS_GPWeightedVoronoi (Input Points, Out Raster, ID Field, Weight Field, Cell Size, Cut Off Cost)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

WeightedVoronoiAllocation (inFeatureClass As IFeatureClass, sOutRaster As String, sIDField As String, sWeightField As String, dCellSize As Double, Optional dCutOff As Double = 0) As IRasterDataset2

# Cost Allocation - Costs from Source

ToolBox Implementation .NET Implementation

Derives the catchment areas of the input features (Sources) using their spatial location and the weight of each source. The function allocates the cells of the output raster to the sources based on minimum cost to reach a source from the cell. The cost is calculated as distance from the cell to the source multiplied by the weight of the source. The smaller the weight value of a source is, the greater the influence of the source is.

This function produces results similar to the results from the Weighted Voronoi Diagram function. The algorithm is different and non-integer weights can be used. Since the Weighted Voronoi Diagram is much faster, the better option is to use it instead of this function unless you need to have double values for the weights.

The performance of the function depends very much on the difference between the smallest and largest weights. The larger this difference is, the slower the function will be.

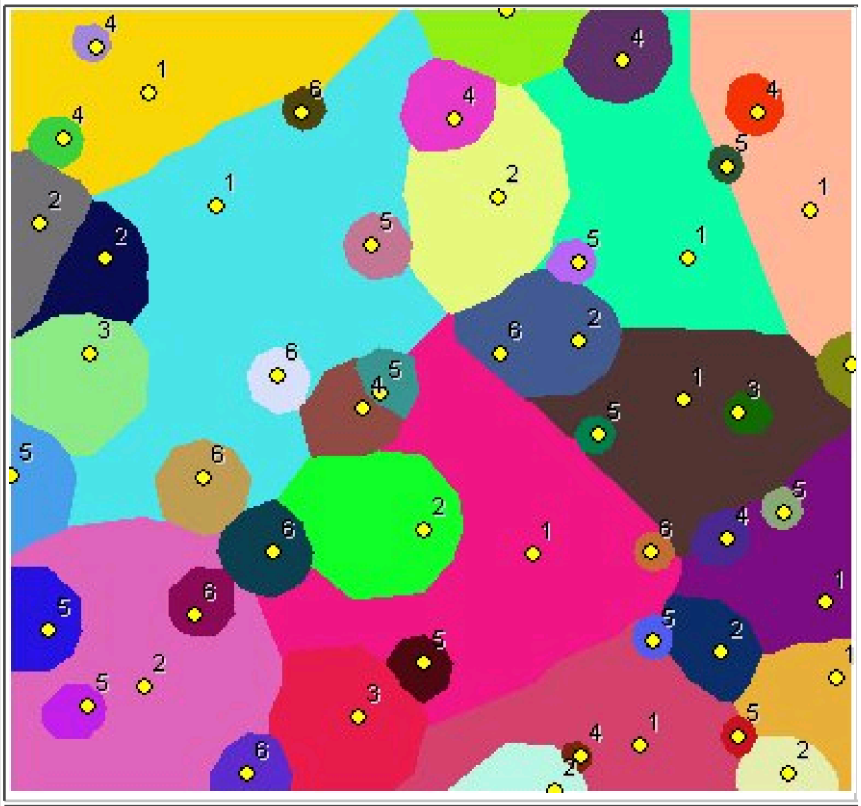
**Inputs:**

- A point feature class(Sources).
- The cell size of the output raster.
- Source ID field. The values from this field will be allocated to the cells of the output raster.
- Weight field. The values of this field will be used to define the weight of the sources. Any numeric field (integer or double) can be used as a weight field.
- Cutoff cost - cells with larger than this cost (distance x weight) to reach will not be allocated to any source.

**Outputs:**

- An integer raster. Each cell will have as a value the ID of the closest input point (Source). The extent of the output is equal to the extent of the input feature class.

**Examples:**

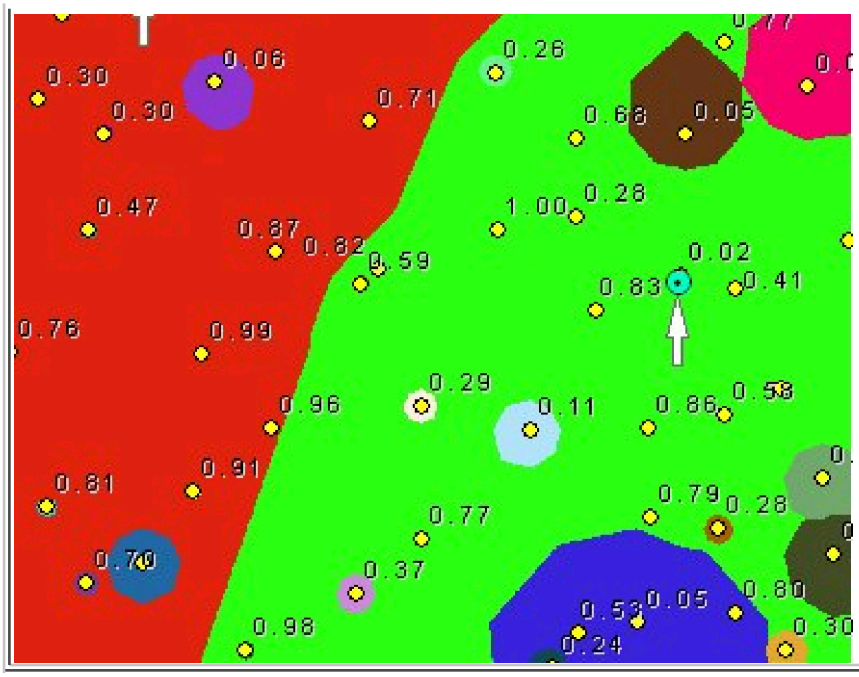


Points labeled with their weights.

Integer weights used - the results very similar to the results produced by Weighted Voronoi Diagram . In this case the use of Weighted Voronoi Diagram function is recommended.


Points labeled with their weights.

Double values for the weights used. The




difference between the smallest weights (indicated on the image) and the largest ones close to 100 times.

Expect long processing time.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class must be in a projected coordinate system.
- The result raster can be easily converted to a polygon feature class using the standard ArcGIS Raster To Polygon tool
- The attributes can be transferred to the polygons by joining the Raster Attribute Table to the polygons using GRID_CODE field of the feature class and the Value field of the raster attribute table.

**ToolBox implementation**

**Command line syntax**

ETS_GPCostAllocationSource <Input Points> <Out Raster> < ID Field> <Weight Field> < Cell Size> {Cut Off Cost}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Points> | A Point layer feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| < ID Field> | A String representing the name of the field in the input point feature class to be used as point ID. |
| <Weight Field> | A String representing the name of the field in the input point feature class that are going to be used as weights. |
| <Cell Size> | A Double representing the cell size of the output raster. |
| {Cut Off Cost} | A Double representing the cut off cost - the value of the cells with larger than this cost (distance x weight) to reach will be set to NODATA |

**Scripting syntax**

ETS_GPCostAllocationSource (Input Points, Out Raster, ID Field, Weight Field, Cell Size, Cut Off Cost)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CostAllocationSource (inFeatureClass As IFeatureClass, sOutRaster As String, sIDField As String, sWeightField As String, dCellSize As Double, Optional dCutOff As Double = 0) As IRasterDataset2

# Cost Distance - Costs from Source

ToolBox Implementation

.NET Implementation

Calculates for each cell of the output raster the least cost to reach one of the sources of the input point feature class. The cost is calculated as distance from the cell to the source multiplied by the weight of the source. The smaller the weight value of a source is, the greater the influence of the source is.

The performance of the function depends very much on the difference between the smallest and largest weights. The larger this difference is, the slower the function will be.

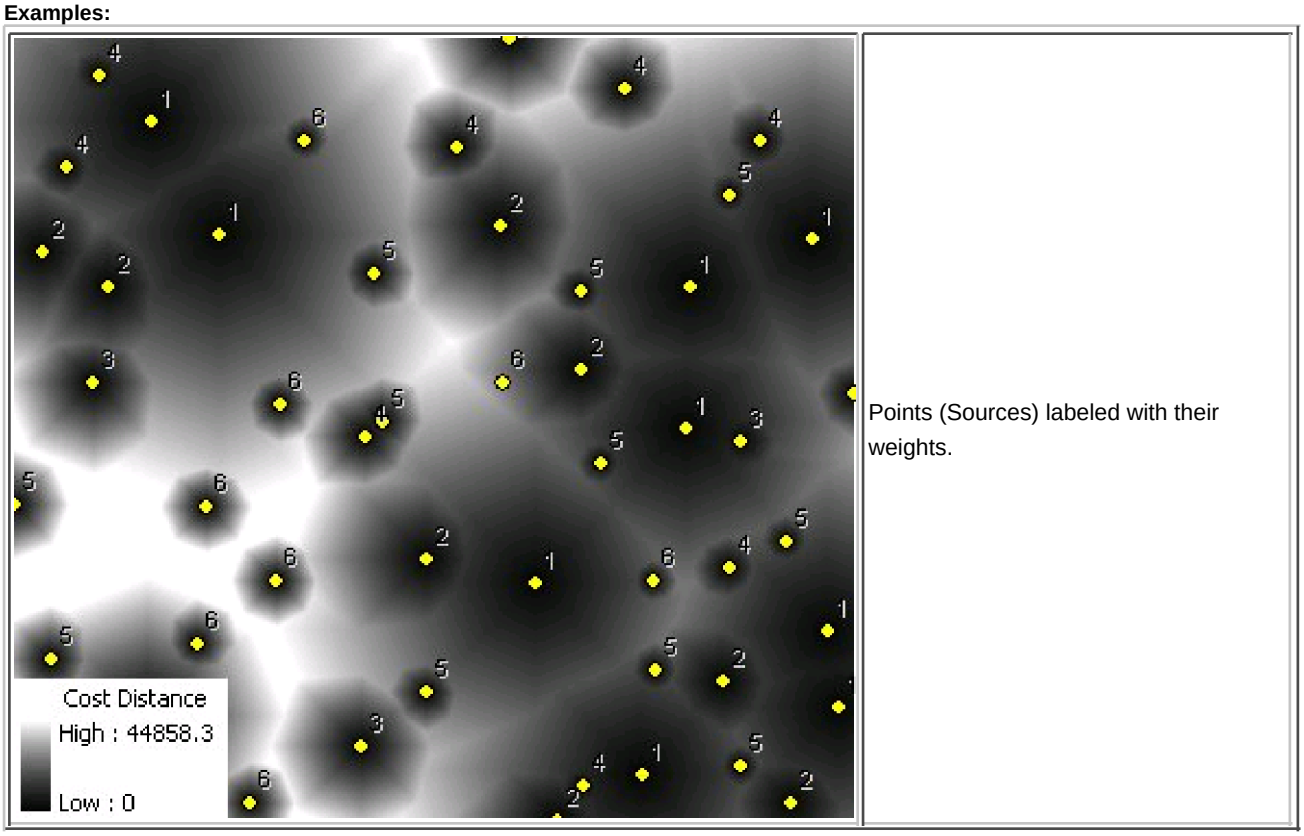
**Inputs:**

- A point feature class (Sources).
- The cell size of the output raster.
- Weight field. The values of this field will be used to define the weight of the sources. Any numeric field (integer or double) can be used as a weight field.
- Cutoff cost - the value of the cells with larger than this cost (distance x weight) to reach will be set to NODATA

**Outputs:**

- A floating point raster. Each cell will have as a value the minimum cost to be reached from one of the sources.

**Examples:**




Points (Sources) labeled with their weights.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPCostDistanceSource <Input Points>  <Out Raster> <Weight Field> < Cell Size> {Cut Off Cost}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Points> | A Point layer feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Weight Field> | A String representing the name of the field in the input point feature class that are going to be used as weights. |
| <Cell Size> | A Double representing the cell size of the output raster. |
| {Cut Off Cost} | A Double representing the cut off cost - the value of the cells with larger than this cost (distance x weight) to reach will be set to NODATA |

**Scripting syntax**

ETS_GPCostDistanceSource (Input Points, Out Raster, Weight Field, Cell Size, Cut Off Cost)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CostDistanceSource (inFeatureClass As IFeatureClass, sOutRaster As String, sWeightField As String, dCellSize As Double, Optional dCutOff As Double = 0) As IRasterDataset2

# Cost Allocation - Costs from Raster

ToolBox Implementation .NET Implementation

Derives the catchment areas of the input features (Sources) using their spatial location and the weights from a weight raster. The function allocates the cells of the output raster to the sources based on minimum cost to reach a source from the cell. The cost is calculated as distance from the cell to the source multiplied by the weight of the cells of the cost raster. The NODATA values in the cost raster are considered prohibitive cost.

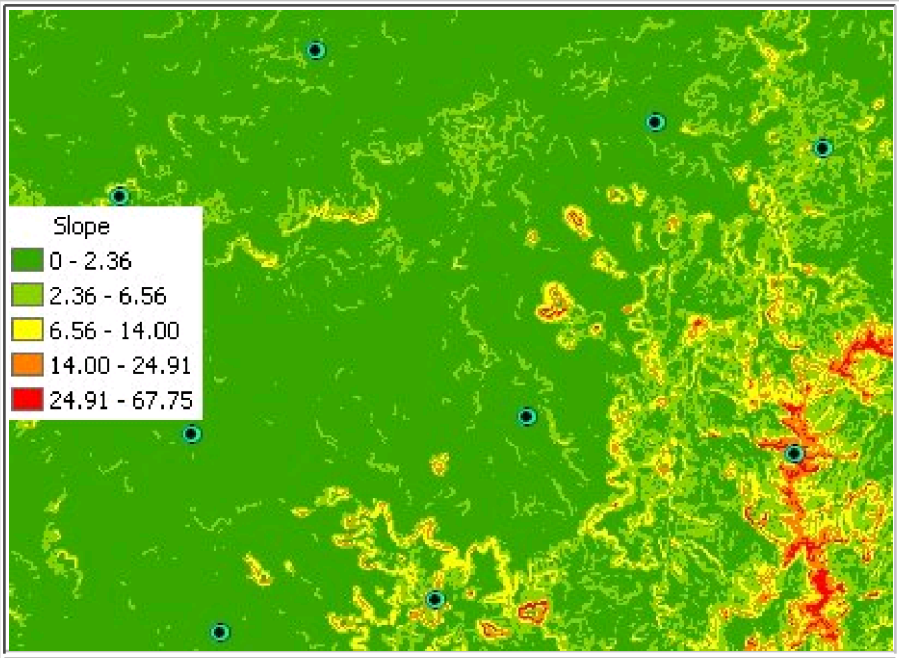
**Inputs:**

- A point feature class (Sources).
- A Cost raster
- Source ID field. The values from this field will be allocated to the cells of the output raster.
- Cutoff cost - cells with larger than this cost (distance x weight) to reach will not be allocated to any source.

**Outputs:**

- An integer raster. Each cell will have as a value the ID of the closest input point (Source).
  - The extent of the output is equal to the extent of the input cost raster.
  - The cell size of the output is equal to the cell size of the input cost raster.

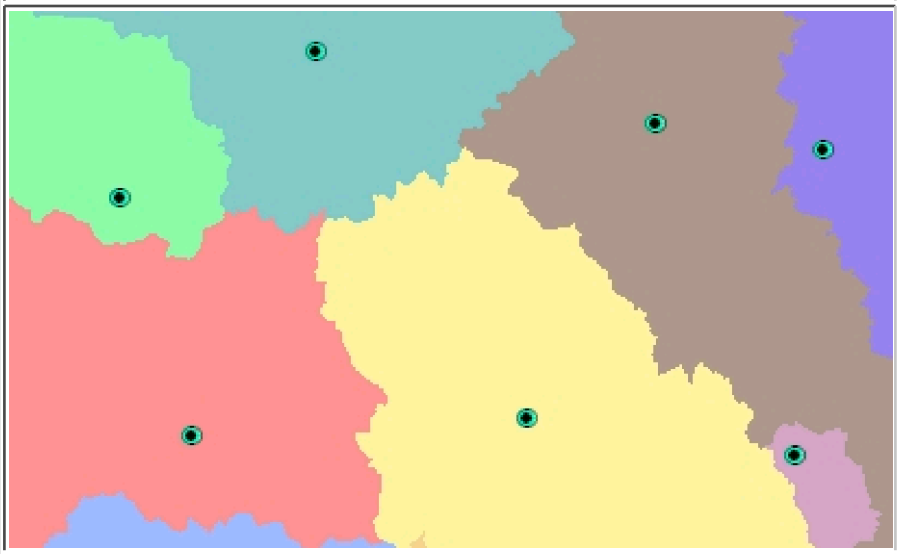
**Examples:**



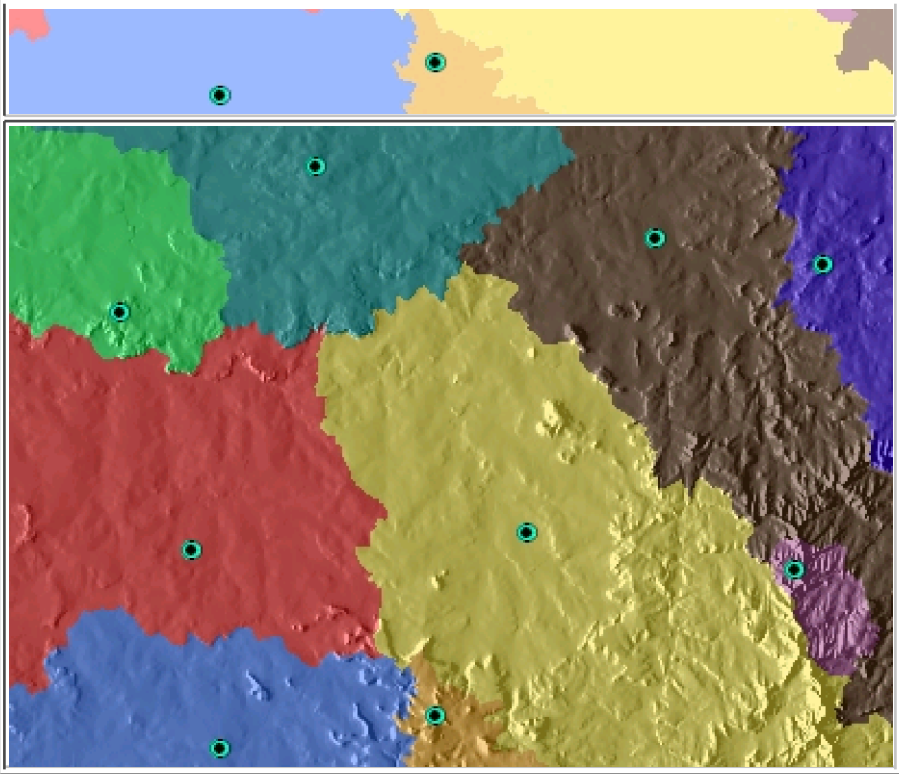

Source points

Cost Raster - Slope raster of digital terrain model used in the example.

The scenario might be to allocate emergency response areas to centers in a mountain - the larger the slope - the lower the accessibility.



The resulting allocation raster.



The resulting allocation raster over the Hillshade of the terrain. Illustrates that the boundaries of the zones allocated to the Centers are on very steep terrain (slope is big - the cost is high).

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class and cost raster must be in the same projected coordinate system.
- The result raster can be easily converted to a polygon feature class using the standard ArcGIS Raster To Polygon tool
- The attributes can be transferred to the polygons by joining the Raster Attribute Table to the polygons using GRID_CODE field of the feature class and the Value field of the raster attribute table.

**ToolBox implementation**

**Command line syntax**

ETS_GPCostAllocationRaster <Input Points> <Cost Raster> <Out Raster> <ID Field> {Cut Off Cost}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Points> | A Point layer feature class |
| <Cost Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <ID Field> | A String representing the name of the field in the input point feature class to be used as point ID. |
| {Cut Off Cost} | A Double representing the cut off cost - the value of the cells with larger than this cost (distance x weight) to reach will be set to NODATA |

**Scripting syntax**

ETS_GPCostAllocationRaster (Input Points, Cost Raster, Out Raster, ID Field, Cut Off Cost)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CostAllocationRaster (inFeatureClass As IFeatureClass, costRasterDataset As IRasterDataset2, sOutRaster As String, sIDField As String, Optional dCutOff As Double = 0) As IRasterDataset2

# Cost Distance - Costs from Raster

ToolBox Implementation .NET Implementation

Calculates for each cell of the output raster the least cost to reach one of the sources of the input point feature class. The cost is calculated as distance from the cell to the source multiplied by the weight of the cells of the cost raster. The NODATA values in the cost raster are considered prohibitive cost.

The performance of the function depends very much on the difference between the smallest and largest weights. The larger this difference is, the slower the function will be.

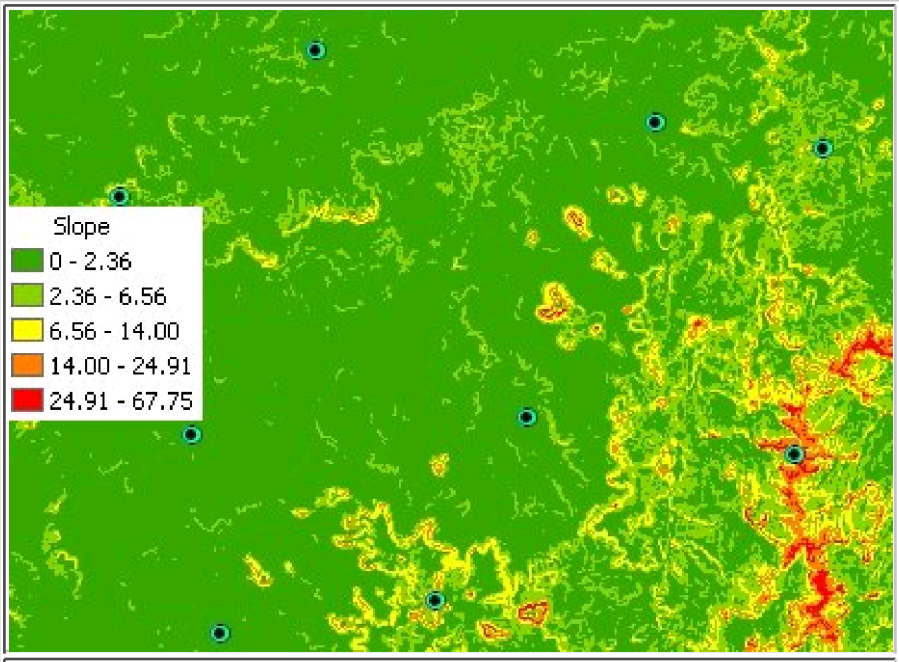
**Inputs:**

- A point feature class (Sources).
- A Cost raster
- Cutoff cost - the value of the cells with larger than this cost (distance x weight) to reach will be set to NODATA

**Outputs:**

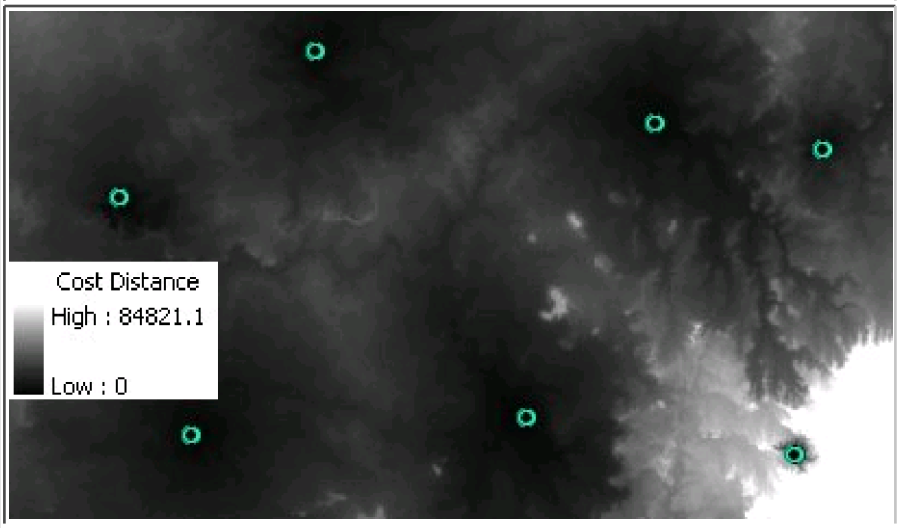
- A floating point raster. Each cell will have as a value the minimum cost to be reached from one of the sources.
  - The extent of the output is equal to the extent of the input cost raster.
  - The cell size of the output is equal to the cell size of the input cost raster.

**Examples:**

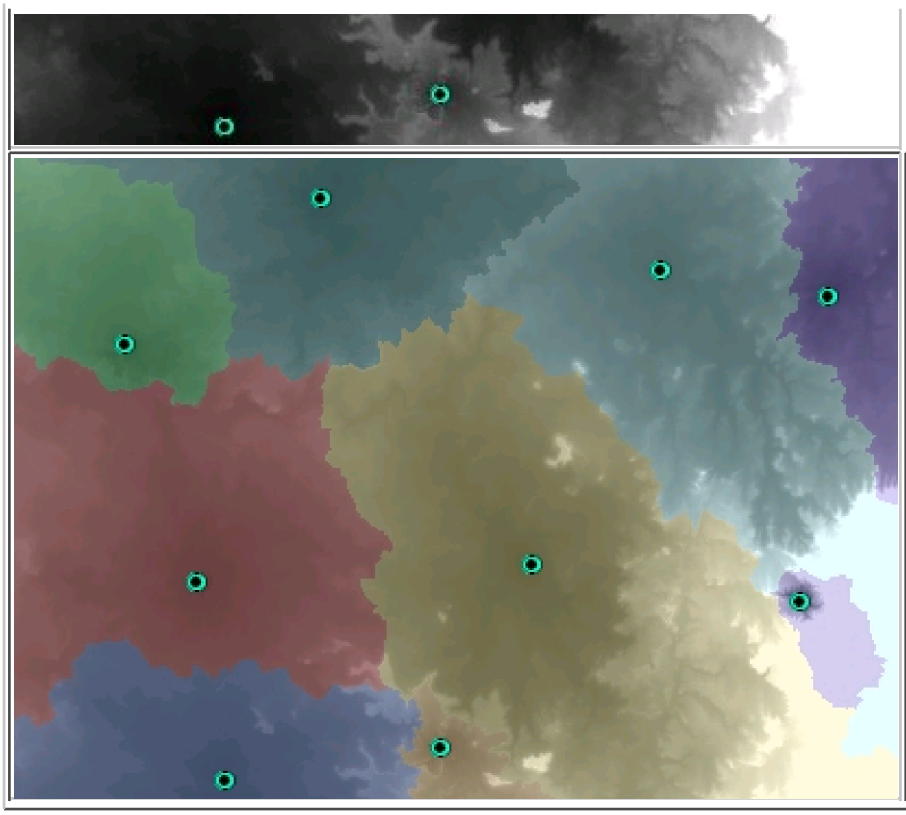



Source points

Cost Raster - Slope raster of digital terrain model used in the example.




The result of the Cost Distance function



The result of the Cost Distance function over the zones derived with the Cost Allocation function

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class and cost raster must be in the same projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPCostDistanceRaster <Input Points> <Cost Raster> <Out Raster> {Cut Off Cost}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Points> | A Point layer feature class |
| <Cost Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| {Cut Off Cost} | A Double representing the cut off cost - the value of the cells with larger than this cost (distance x weight) to reach will be set to NODATA |

**Scripting syntax**

ETS_GPCostDistanceRaster (Input Points, Cost Raster, Out Raster, Cut Off Cost)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CostDistanceRaster (inFeatureClass As IFeatureClass, costRasterDataset As IRasterDataset2, sOutRaster As String, Optional dCutOff As Double = 0) As IRasterDataset2

# Clip Raster with Envelope

ToolBox Implementation .NET Implementation

Clips a raster with user defined envelope. The extent can be imported from an existing feature class or raster.

**Inputs:**

- A raster dataset.

**Outputs:**

- A raster with cell values equal to the cell values of the input raster, but only in the extent of the clip envelope.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- If the extents of the clip envelope are outside of the extents of the input raster, the result will be a copy of the input raster.

**ToolBox implementation**

**Command line syntax**

ETS_GPClipRasterWithEnvelope <Input Raster> <Out Raster> {Extent from Existing} <Extent>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | Raster layer or raster dataset |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| {Extent from Existing} | A string representing the full path to an existing dataset. The extent of the output raster will have the extent of this dataset |
| <Extent> | A String representing the extent of the output raster. Example "0, 0, 500, 250" |

**Examples:**

- *ETS_GPClipRasterWithEnvelope c:\test\r1.img c:\test\clipped1.img c:\test\extent_source.img* # will clip "r1.img" with the extents of the existing raster extent_source.img
- ETS_*GPClipRasterWithEnvelope c:\test\r1.img c:\test\clipped1.img # "0, 0, 500, 250"* will clip raster "r1.img" with envelope with X Min = 0, Y Min = 0, X Max = 500 and Y Max = 250

**Scripting syntax**

ETS_GPClipRasterWithEnvelope (Input Raster Out Raster, Extent from Existing, Extent)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ClipRasterWithEnvelope (inRasterDataset As IRasterDataset2, sOutRaster As String, pExtent As IEnvelope) As IRasterDataset2

# Clip Raster with Polygons

ToolBox Implementation .NET Implementation

Clips a raster dataset with the polygons of the input polygon feature class

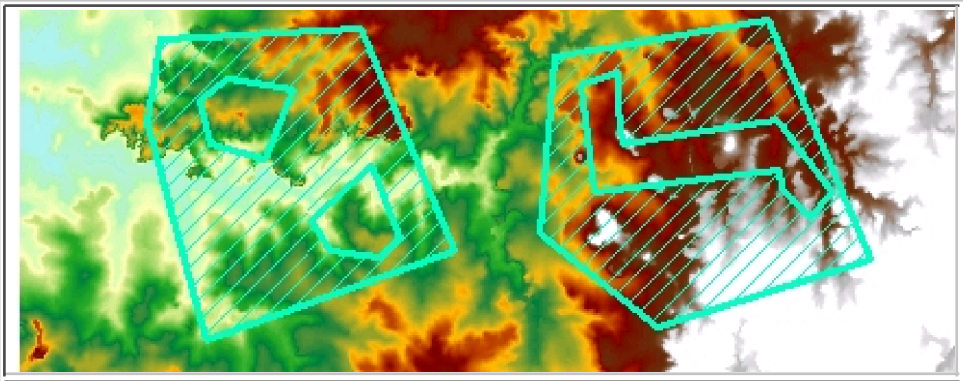
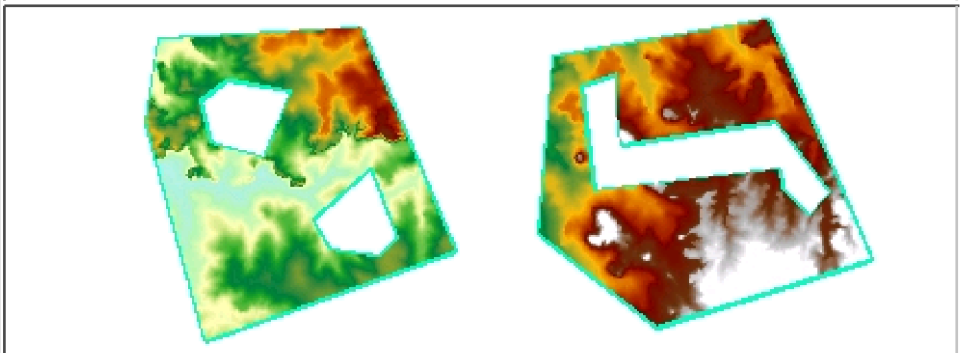
**Inputs:**

- A polygon feature class
- A raster dataset.

**Outputs:**

- A raster with cell values equal to the cell values of the input raster, but only in the locations overlapping the clip polygons.

**Example:**

| | |
|---|---|
|  | Input raster and clip polygons |
|  | Result raster |

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class and raster must have the same projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPClipRasterWithPolygons <Input Raster> <Clip Polygons> <Out Raster>

**Parameters**

| Expression | Explanation |
| --- | --- |
| <Input Raster> | A Raster dataset or Raster layer |
| <Clip Polygons> | A Polygon layer or feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Scripting syntax**

ETS_GPClipRasterWithPolygons (Input Raster, Clip Polygons, Out Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ClipRasterWithPolygons (inRasterDataset As IRasterDataset2, inFeatureClass As IFeatureClass, sOutRaster As String) As IRasterDataset2

# Erase Raster with Polygons

ToolBox Implementation .NET Implementation

Erases a raster dataset with the polygons of the input polygon feature class.

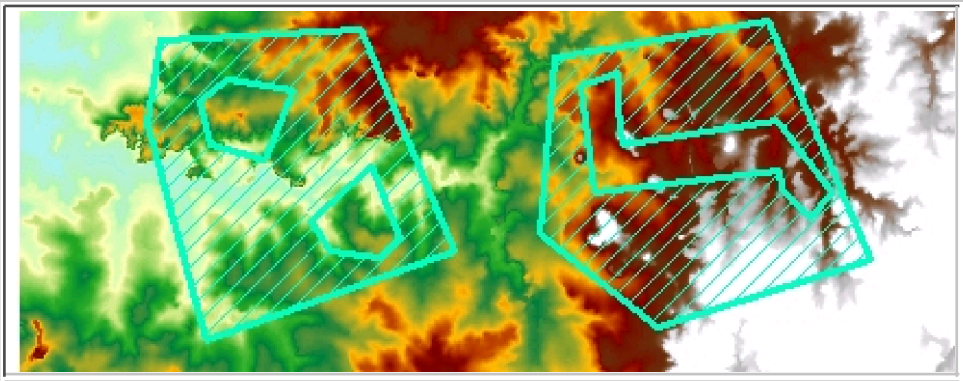
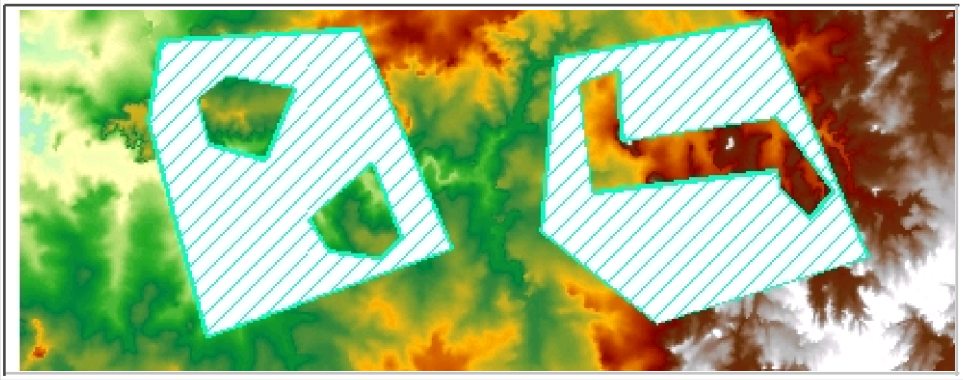
**Inputs:**

- A polygon feature class
- A raster dataset representing.

**Outputs:**

- A raster with NoData values for the cells overlapping the erase polygons.

**Example:**

| | |
|---|---|
|  | Input raster and erase polygons |
|  | Result raster |

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input feature class and raster must have the same projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPEraseRasterWithPolygons <Input Raster> <Erase Polygons> <Out Raster>

**Parameters**

| Expression | Explanation |
|---|---|

| | |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Erase Polygons> | A Polygon layer or feature class |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Scripting syntax**

ETS_GPEraseRasterWithPolygons (Input Raster, Erase Polygons, Out Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

EraseRasterWithPolygons (inRasterDataset As IRasterDataset2, inFeatureClass As IFeatureClass, sOutRaster As String) As IRasterDataset2

# Smooth Raster

ToolBox Implementation .NET Implementation

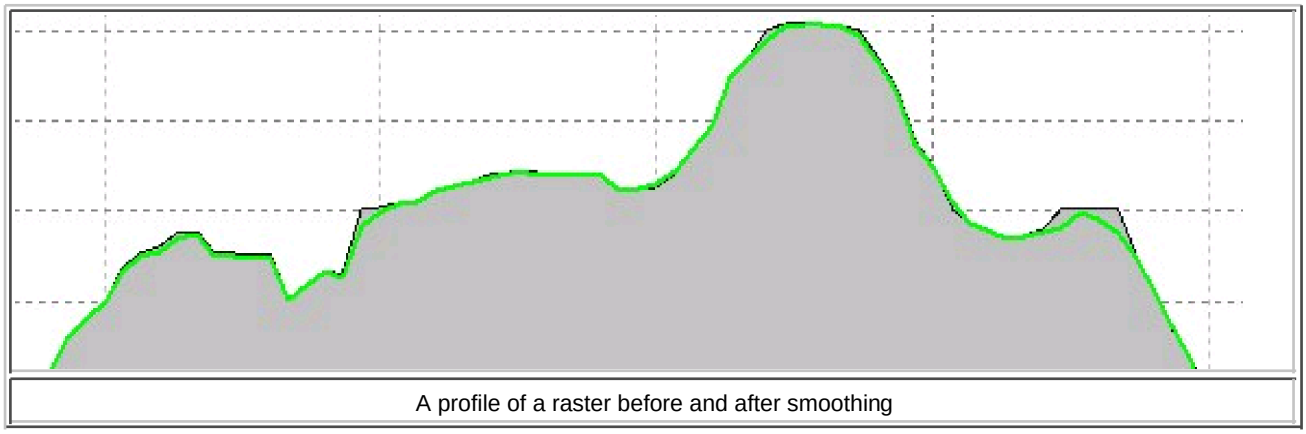
Smooth a raster using Gaussian function on 3 x 3 or 5 x 5 neighborhood.

Read more about Gaussian smoothing here

**Inputs:**

- Input floating point raster dataset
- Output raster name and format
- Neighborhood size - 3 x 3 or 5 x 5

**Example:**



A profile of a raster before and after smoothing

**Output:**

- A floating point raster.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPSmoothRaster <Input Raster> <Out Raster> <Neighborhood>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

| | |
|---|---|
| <Neighborhood> | A String indicating the neighborhood to be used. Valid inputs "3x3" and "5x5" |

**Scripting syntax**

ETS_GPSmoothRaster (Input Raster,  Out_Raster, Neighborhood)

**.NET implementation**
(Go to TOP)

SmoothRaster (inRasterDataset As IRasterDataset2, sOutRaster As String, sNeighbourhood As String) As IRasterDataset2

# Clean Boundaries

ToolBox Implementation .NET Implementation

Cleans the noise around the boundaries between zones of an integer grid. 3 x 3 neighborhood is analyzed for each cell and the value of the cell is set to the value of the majority of the neighbors.

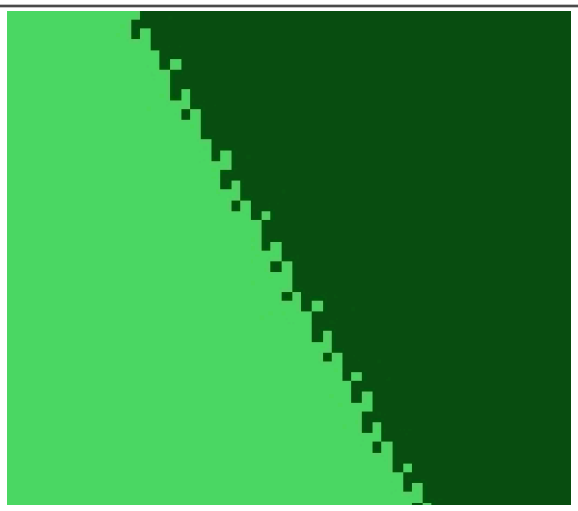
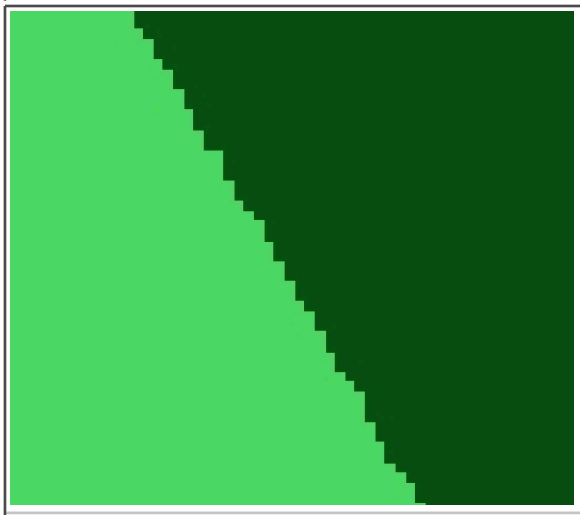
**Inputs:**

- Input integer raster dataset
- Output raster name and format

**Output:**

- An integer raster.

**Example:**

| Before Clean Boundaries | After Clean Boundaries |
| --- | --- |
|  |  |

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPCleanBoundaries <Input Raster> <Out Raster>

**Parameters**

| Expression | Explanation |
| --- | --- |
| <Input Raster> | A Raster dataset or Raster layer |

| | |
|---|---|
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Scripting syntax**

ETS_GPCleanBoundaries (Input Raster,  Out Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CleanBoundaries (inRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Create Constant Raster

ToolBox Implementation .NET Implementation

Creates a constant raster with user defined data type, cell size and extent. The extent can be imported from an existing feature class or raster.

**Inputs:**

- Raster data type - integer or floating point. If Integer data type is specified, depending on the constant value specified, the output might be:
  - SHORT - signed 16 bit integer (values between -32768 and 32768)
  - LONG - signed 32 bit integer (values out of the range above)
- Output raster name and format
- Constant value
- Cell Size of the output raster
- Extents of the output raster (can be copied from an existing feature class or raster). The default extents are the current extents of the active view.
- Output Spatial Reference. The default spatial reference is the one assigned to the data frame. If the extents are copied from an existing dataset, the default spatial reference is changed to the projection of this dataset. The selected spatial reference must be projected coordinate system.

**Output:**

- A raster with constant value for all cells.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

**ToolBox implementation**

**Command line syntax**

ETS_GPConstantRaster <Out Raster> {Out Spatial Reference} <Out Type> {Extent from Existing} <Extent><Cell Size><Raster Value>

**Parameters**

| Expression | Explanation |
|---|---|
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| {Out Spatial Reference} | The spatial reference of the output raster. The dialog allows the user to select a predefined spatial reference or import spatial reference from an existing dataset. If used from the Command Line, the user can just specify the full name of an existing dataset. |
| <Out Type> | A String indicating the type of the output raster. Valid values - "Float" and "Integer" |
| {Extent from Existing} | A string representing the full path to an existing dataset. The extent of the output raster will have the extent of this dataset |

| | |
|---|---|
| <Extent> | A String representing the extent of the output raster. Example "0, 0, 500, 250" |
| <Cell Size> | A Double representing the cell size |
| <Raster Value> | A Number representing the minimum value of the output raster |

**Examples:**

- *ETS_GPConstantRaster c:\test\r1.img c:\test\source1.img Float c:\test\source1.img # 10 50.51* - will create a floating raster "r1.img" with Cell Size = 10 Value = 50.51 in c:\test folder. The spatial reference and the extent will be the same as the ones of the existing raster source1.img
- *ETS_GPConstantRaster c:\test\r1.img c:\test\source1.img Integer # "0, 0, 500, 250" 10 50* - will create an integer raster "r1.img" with Cell Size = 10 Value = 50 in c:\test folder. The spatial reference will be the same as the one of source1.img (must exist) and the extents will be defined by the envelope with X Min = 0, Y Min = 0, X Max = 500 and Y Max = 250

**Scripting syntax**

ETS_GPConstantRaster (Out Raster, Out Spatial Reference, Out Type, Extent from Existing, Extent, Cell Size, Raster Value,)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CreateConstantRaster (sOutRaster As String, spatialReference As ISpatialReference2, sDataType As String, pExtent As IEnvelope, dCellSize As Double, constValue As Single) As IRasterDataset2

# Create Random Raster

ToolBox Implementation .NET Implementation

Creates an integer raster with random values in user defined range, cell size and extent. The extent can be imported from an existing feature class or raster.

**Inputs:**

- Output raster name and format
- Minimum and Maximum values that indicate the range of random values to be assigned to the cells of the output raster.
- Cell Size of the output raster
- Extents of the output raster (can be copied from an existing feature class or raster). The default extents are the current extents of the active view.
- Output Spatial Reference. The default spatial reference is the one assigned to the data frame. If the extents are copied from an existing dataset, the default spatial reference is changed to the projection of this dataset. The selected spatial reference must be projected coordinate system.

**Output:**

- An integer raster. Depending on the range specified, the output might be:
  - SHORT - signed 16 bit integer (values between -32768 and 32768)
  - LONG - signed 32 bit integer (values out of the range above)

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

**ToolBox implementation**

**Command line syntax**

ETS_GPRandomRaster <Out Raster> {Out Spatial Reference} {Extent from Existing} <Extent><Cell Size><Min Value> <Max Value>

**Parameters**

| Expression | Explanation |
|---|---|
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| {Out Spatial Reference} | The spatial reference of the output raster. The dialog allows the user to select a predefined spatial reference or import spatial reference from an existing dataset. If used from the Command Line, the user can just specify the full name of an existing dataset. |
| {Extent from Existing} | A string representing the full path to an existing dataset. The extent of the output raster will have the extent of this dataset |
| <Extent> | A String representing the extent of the output raster. Example "0, 0, 500, 250" |

| | |
|---|---|
| <Cell Size> | A Double representing the cell size |
| <Min Value> | A Number representing the minimum value of the output raster |
| <Max Value> | A  Number representing the maximum value of the output raster. |

**Examples:**

- *ETS_GPRandomRaster c:\test\r1.img c:\test\source1.img c:\test\source1.img # 10 50 100* - will create raster "r1.img" with Cell Size = 10 Min Value = 50 and Max Value = 100 in "c:\test" folder. The spatial reference and the extent will be the same as the ones of the existing raster source1.img
- ETS_GPRandomRaster c:\test\r1.img c:\test\source1.img  # "0, 0, 500, 250" 10 50 100 - will create raster "r1.img" with Cell Size = 10 Min Value = 50 and Max Value = 100 in "c:\test" folder. The spatial reference will be the same as the one of source1.img (must exist) and the extents will be defined by the envelope with X Min = 0, Y Min = 0, X Max = 500 and Y Max = 250

**Scripting syntax**

ETS_GPRandomRaster (Out Raster, Out Spatial Reference, Extent from Existing, Extent, Cell Size, Min Value, Max Value)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CreateRandomRaster (sOutRaster As String, spatialReference As ISpatialReference2, pExtent As IEnvelope, dCellSize As Double, minValue As Integer, maxValue As Integer) As IRasterDataset2

# Change Raster Data Type

ToolBox Implementation .NET Implementation

Changes the data type of the input raster to the output type specified by the user..

**Inputs:**

- Input floating point or integer raster
- Output raster name and format.
- Output type.

**Output:**

- An raster of the specified data type
  - The NODATA of the output will be
    - The same as the NODATA value of the input if it is within the range of the selected output type
    - Assigned automatically based on the selected output type if the input NODATA value is out of the range of the output data type selected.
  - If the input raster has values that are out of the range of the output data type, these values will be stored as NODATA.

Example: If the input raster is of floating point type and the output type selected is "Byte"

- The output raster will have NODATA = 255
- The output raster will have values only in the cells where the original raster has values between 0 and 254 and these values will be rounded to the closest integer. The rest of the cells will have NODATA value.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

**ToolBox implementation**

**Command line syntax**

ETS_GPChangeRasterType <Input Raster> <Out Raster> <Out Type>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Out Type> | A String indicating the slope units. Valid inputs "Byte", "Double", "Long", "Short", "Single" |

**Scripting syntax**

ETS_GPChangeRasterType (Input Raster, Out Raster, Out Type)

See the explanations above:

<> - required parameter
{} - optional parameter

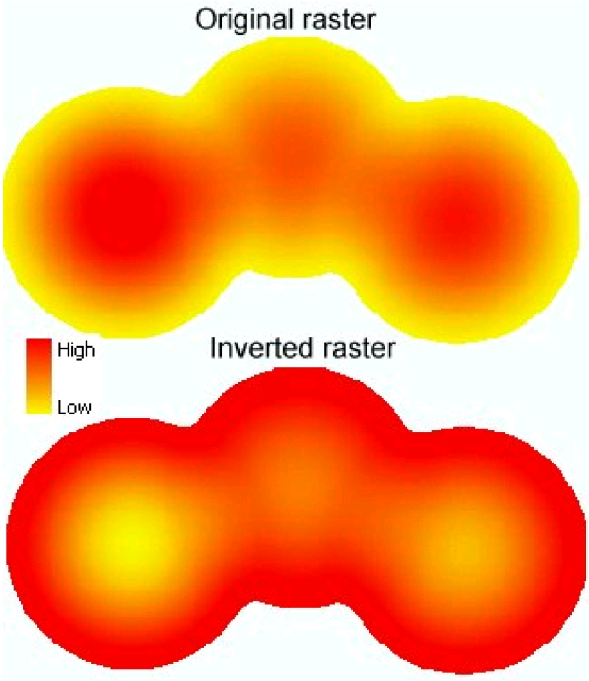
**.NET implementation**
(Go to TOP)

ChangeRasterType (inRasterDataset As IRasterDataset2, sOutRaster As String, sType As String) As IRasterDataset2

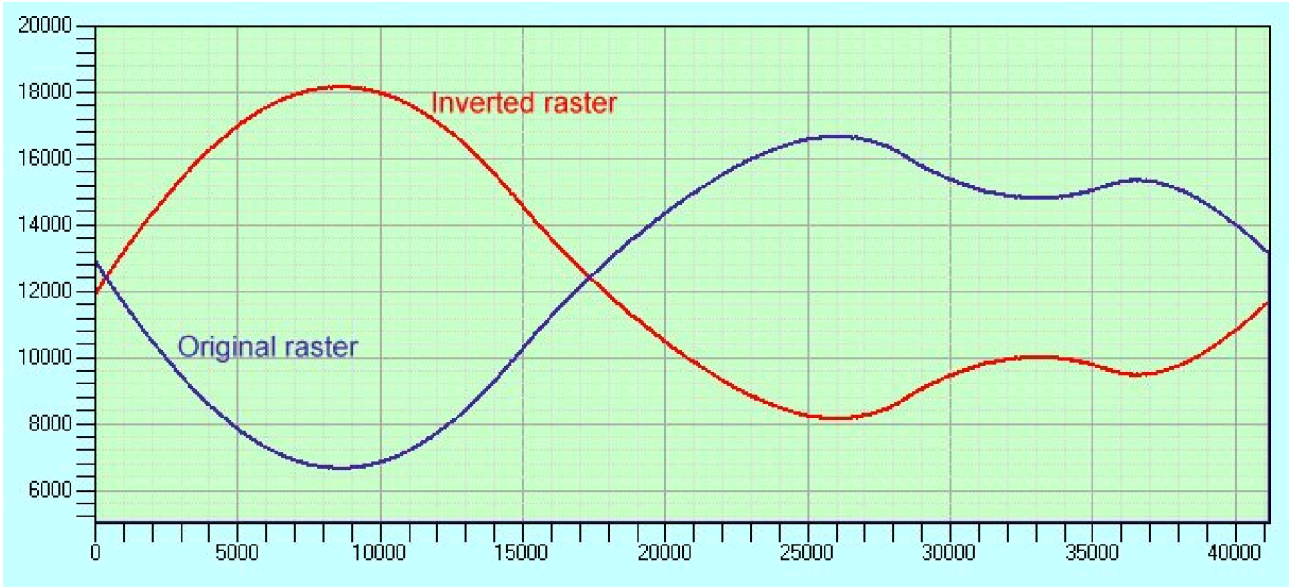
# Invert Raster

ToolBox Implementation .NET Implementation

Recalculates the values of the cells of the input raster in such a way that the cell with the lowest value in the input receives the value of the highest cell and vice versa.
If the raster represents a terrain, the valleys become ridges and the ridges - valleys




Cross sectional profile of the input and output rasters




**Inputs:**

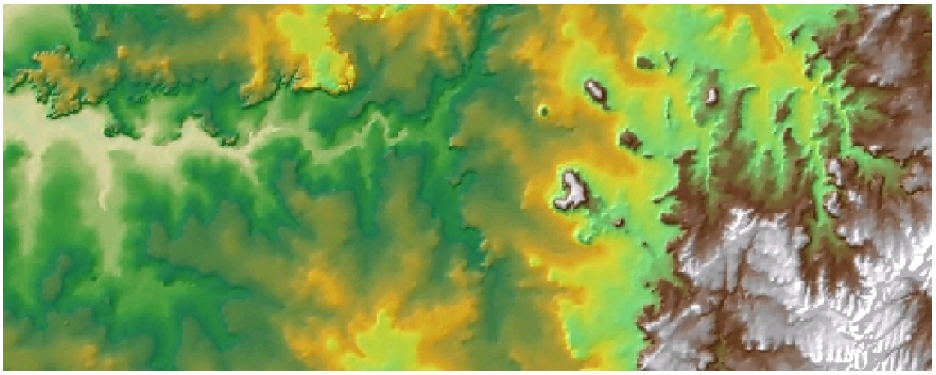
- Input raster dataset
- Output raster name and format

**Output:**

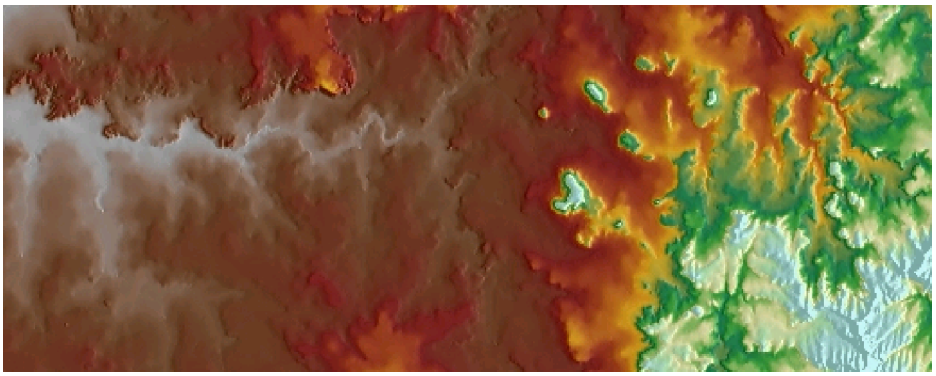
- A floating point raster.

**Example:**

Input raster



Inverted raster



**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPInvertRaster <Input Raster> <Out Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Scripting syntax**

ETS_GPInvertRaster (Input Raster, Out_Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

InvertRaster (inRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Resample Raster

ToolBox Implementation .NET Implementation

Alters a raster by changing the cell size and/or extent. The extent can be imported from an existing feature class or raster.

**Inputs:**

- A raster dataset.
- New cell size value
- New extent (optional)

**Outputs:**

- A raster with cell size and extent according to user input. The resampling method is "Nearest Neighbour" meaning that the value of the cell will be according to the nearest cell center of the source raster. The new extent will be honoured and all values outside of the source raster extent will be assigned to NoData.

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

**ToolBox implementation**

**Command line syntax**

ETS_GPResampleRaster <Input Raster> <Out Raster><Cell Size> {Extent from Existing} {Extent}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | Raster layer or raster dataset |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Cell Size> | A Double representing the cell size |
| {Extent from Existing} | A string representing the full path to an existing dataset. The extent of the output raster will have the extent of this dataset |
| {Extent} | A String representing the extent of the output raster. Example "0, 0, 500, 250" |

**Examples:**

- *ETS_GPResampleRaster c:\test\r1.img c:\test\esampled1.img15 # #* will resample "r1.img" with the same extent and a new cell size of 15.
- *ETS_GPResampleRaster c:\test\r1.img c:\test\resampled1.img 15 # "0, 0, 500, 250"*will resample raster "r1.img" with envelope with X Min = 0, Y Min = 0, X Max = 500 and Y Max = 250 and new cell size of 15

**Scripting syntax**

ETS_GPResampleRaster (Input Raster, Out Raster, Cell Size, Extent from Existing, Extent)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ResampleRaster (inRasterDataset As IRasterDataset2, sOutRaster As String, dCellsize As Double, Optional pExtent As IEnvelope = Nothing) As IRasterDataset2

# Find NoFlow Areas

ToolBox Implementation

.NET Implementation

Finds cells in DEM rasters with undefined flow based on the D8 flow direction algorithm. These cells are either sinks (lower than all neighboring cells) or in the interior of a flat area. The flat area cells are marked with value of 1 and the sink cells with the value 2.

**Inputs:**

- A DEM raster dataset.

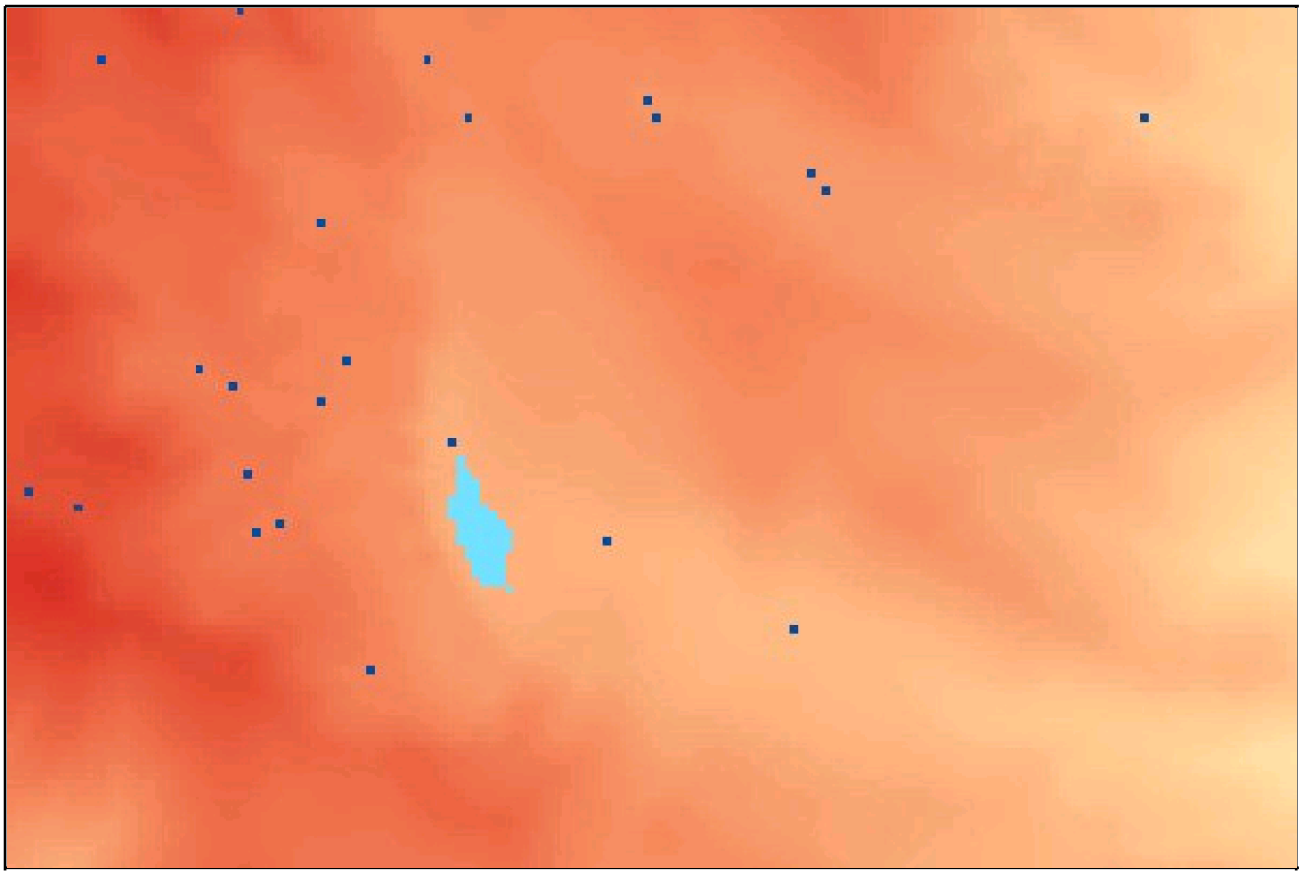
**Outputs:**

- A raster indicating the cells without flow in the DEM on a background of NoData.

**Notes:**

- An output value of 1 indicates that the cell belongs to a flat area - there is no lower neighboring cell and the lowest neighbor has the same elevation.
- An output value of 2 indicates a sink - meaning that all neighboring cells are with higher elevation.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
    - no extension specified - ESRI binary GRID
    - .img extension (for example raster1.img) - ERDAS IMAGINE image.
    - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
    - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of the NoFlow Areas output - Flat areas with light blue and Sinks with dark blue.



**ToolBox implementation**

**Command line syntax**

ETS_GPNoFlowAreas <Input Raster> <Out Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | DEM Raster layer or raster dataset |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Examples:**

- *ETS_GPNoFlowAreas c:\test\dem.img c:\test\NoFlow.img*

**Scripting syntax**

ETS_GPNoFlowAreas (Input Raster, Out Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

NoFlowAreas (demRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Fill Depressions

ToolBox Implementation .NET Implementation

Fills all depressions in a DEM raster and removes flat areas producing a "Depressionless" DEM.

**Inputs:**

- A DEM raster dataset.

**Outputs:**

- A raster representing a Depressionless DEM in which the flow can be determined for every inner cell and all outlets are at the DEM boundary (or next to NoData areas).

**Notes:**

- The Fill Depressions function is based on the algorithm proposed by Planchon and Darboux (2001).
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

**ToolBox implementation**

**Command line syntax**

ETS_GPFillDepressions <Input DEM Raster> <Out Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input DEM Raster> | Raster layer or raster dataset |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Examples:**

- *ETS_GPFillDepressions c:\test\dem.img c:\test\filldem.img*

**Scripting syntax**

ETS_GPFillDepressions (Input DEM Raster, Out Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**Reference:**

Planchon, O. and F. Darboux, (2001), "A fast, simple and versatile algorithm to fill the depressions of digital elevation models", Catena, 46: 159-176.

**.NET implementation**

(Go to TOP)

FillDepressions (demRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Flow Direction D8

ToolBox Implementation .NET Implementation

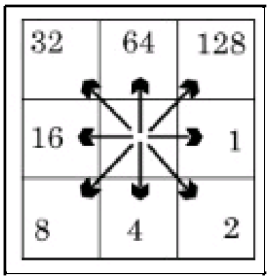
Generates a Flow Direction raster in which each cells represents the direction of flow for this cell in the DEM using the D8 method.

**Inputs:**

- A DEM raster dataset.

**Outputs:**

- An integer raster representing the Flow Direction according to the D8 (Deterministic 8) method. The 8 flow directions are coded with values 1,2,4,8,16,32,64,128 and value of 0 indicates an outlet. The values are coded using the following convention:

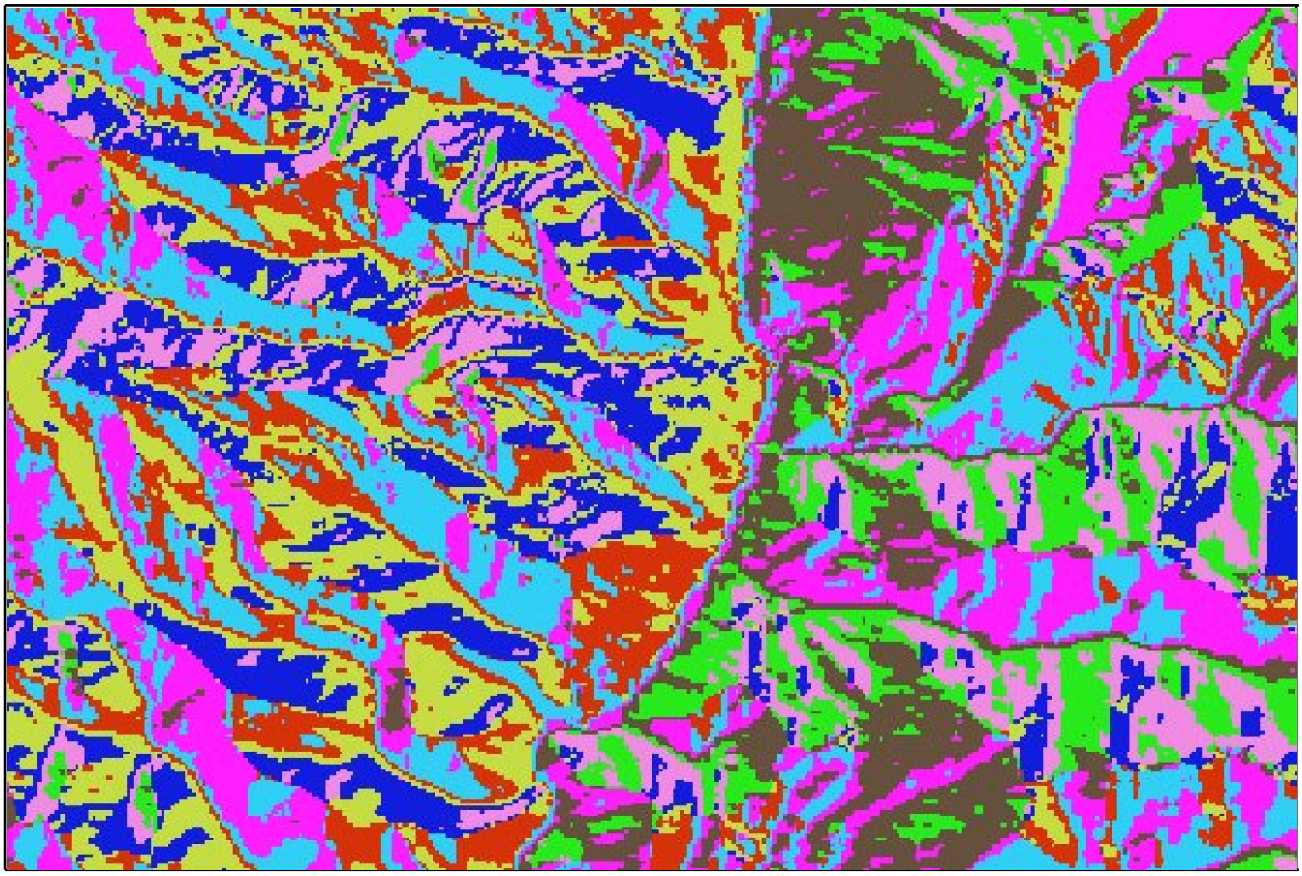



For example if the steepest descent of a cell is to the left, its Flow direction will be coded as 1, if it is to upper right, it will be coded as 32. The steepest descent is calculated by dividing the elevation difference by the distance between the cell centers.

**Notes:**

- The Flow direction values are determined based on the steepest downslope neighbor for each cell.
- If a cell is lower than its 8 neighbors it is assigned a value of 0 meaning it is an outlet or sink.
- If a cell has the same slope in more than one direction, the first encountered direction is assigned.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Flow Direction D8 output.



**ToolBox implementation**

**Command line syntax**

ETS_GPFlowDirectionD8 <Input DEM Raster> <Out Flow Direction Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input DEM Raster> | DEM raster layer or raster dataset |
| <Out Flow Direction Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Examples:**

- *ETS_GPFlowDirectionD8 c:\test\dem.img c:\test\flowdir8.img*

**Scripting syntax**

ETS_GPFlowDirectionD8 (Input DEM Raster, Out Flow Direction Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

FlowDirectionD8 (demRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Flow Direction D-infinity

ToolBox Implementation .NET Implementation

Generates a Flow Direction raster in which each cells represents the direction of flow for this cell in the DEM using the D-infinity method.

**Inputs:**

- A DEM raster dataset.

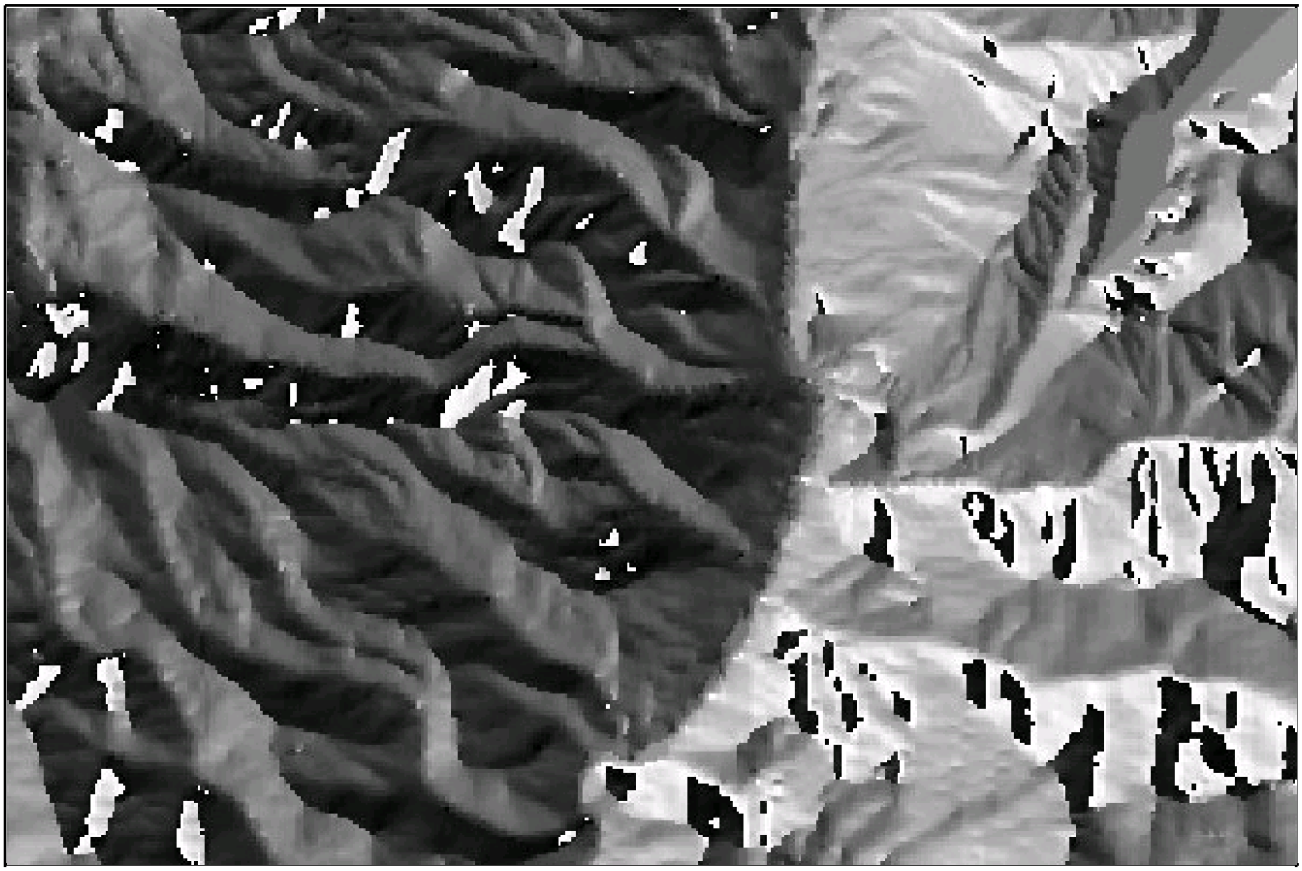
**Outputs:**

- A Floating raster representing the Flow Direction according to the D-infinity (Deterministic Infinity) method.

**Notes:**

- The D-infinity method was suggested by Tarboton (1997). In it Flow Direction is defined as the angle of the steepest descent determined by analysis of 8 triangular facets formed by the 3x3 cell neighborhood.
- The flow direction values are in Decimal Degrees from 0 to 360.0. The value of 0.0 indicates an outlet and all other values represent the direction of flow starting from North in clockwise direction.
- For example a value of 90 indicates flow to the East and a value of 225 - flow to South West.
- The D-infinity method allows flow divergence - the flow from a cell will either go to one or two of the neighboring cells. This is a better representation of water flow on divergent slopes.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Flow Direction D-infinity output.



**ToolBox implementation**

**Command line syntax**

ETS_GPFlowDirectionDinf <Input DEM Raster> <Out Flow Direction Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input DEM Raster> | DEM raster layer or raster dataset |
| <Out Flow Direction Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Examples:**

- *ETS_GPFlowDirectionDinf c:\test\dem.img c:\test\flowdirInf.img* will create "flowdirInf.img".

**Scripting syntax**

ETS_GPFlowDirectionDinf (Input DEM Raster, Out Flow Direction Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

FlowDirectionDinf (demRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

**Reference:**

Tarboton, D. G., (1997), "A New Method for the Determination of Flow Directions and Contributing Areas in Grid Digital Elevation Models", Water Resources Research, 33(2): 309-319

# Flow Accumulation D8

ToolBox Implementation .NET Implementation

Generates a Flow Accumulation raster using the Deterministic 8 (D8) flow direction model.

**Inputs:**

- A D8 Flow Direction raster dataset.

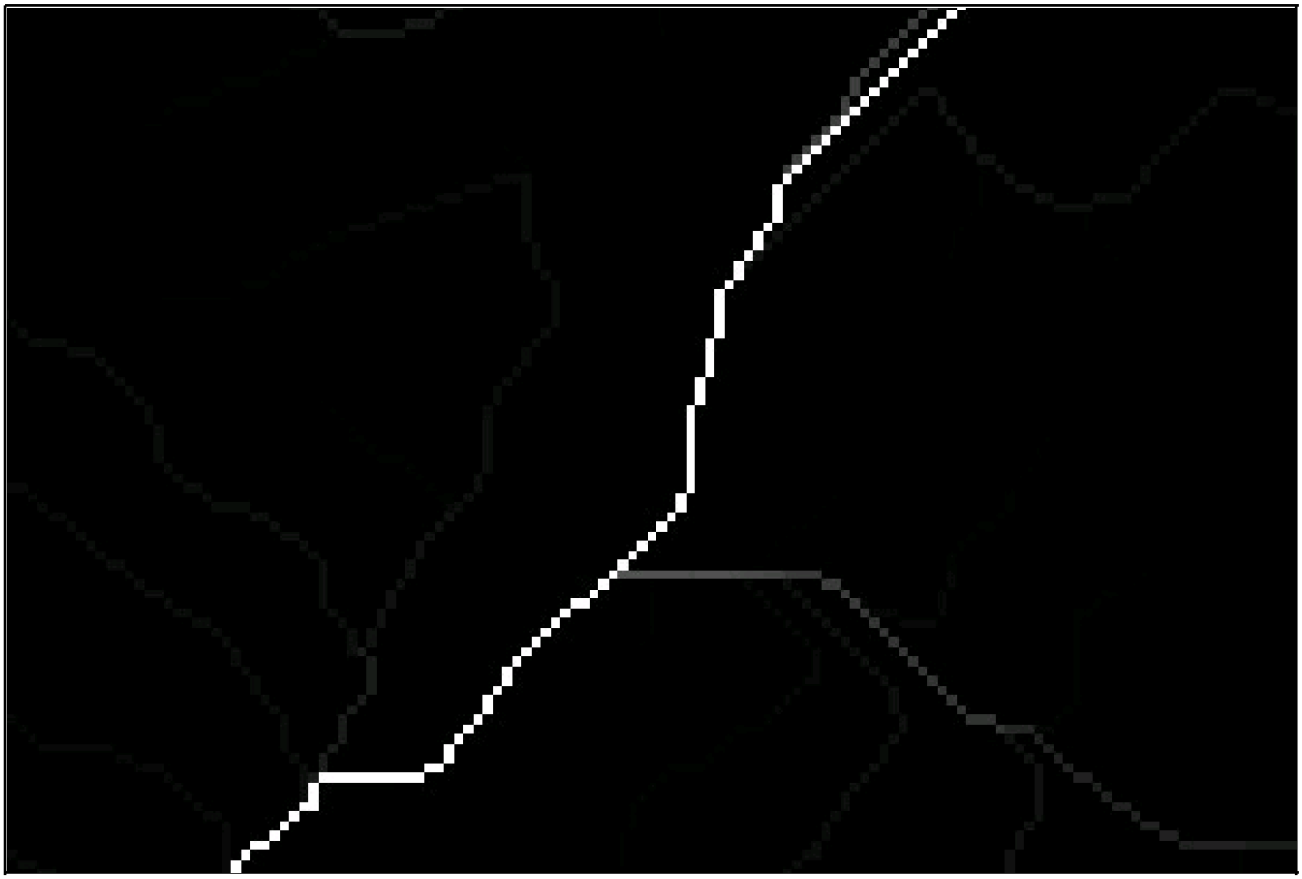
**Outputs:**

- A Flow Accumulation raster .

**Notes:**

- The value for each cell in the Flow Accumulation raster is the total number of cells which flow into it including the cell itself. So for a cell with no inflowing cells the value will be 1.
- The value in the Flow Accumulation raster represents the number of upstream cells from which the water flowing downstream will pass through the current cell.
- The actual contributing area for the cells can be determined by multiplying the accumulation value by the area represented by a cell.
- Thus for a raster with cell size 1 x 1 meter the values are directly square meters. For a cell size of 10 x 10 meters, the accumulated value must be multiplied by 100 to obtain the contributing area.
- Cells with a high value in the Flow Accumulation raster indicate high concentration of water and can be used for identification of streams.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
    - no extension specified - ESRI binary GRID
    - .img extension (for example raster1.img) - ERDAS IMAGINE image.
    - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
    - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Flow Accumulation D8 result.



**ToolBox implementation**

**Command line syntax**

ETS_GPFlowAccumulationD8 <Input Flow Direction Raster> <Out Flow Accumulation Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Flow Direction Raster> | A D8 Flow Direction Raster layer or Raster dataset |
| <Out Flow Accumulation Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Examples:**

- *ETS_GPFlowAccumulationD8 c:\test\flowDirD8.img c:\test\flowAccD8.img*

**Scripting syntax**

ETS_GPFlowAccumulationD8 (Input Flow Direction Raster, Out Flow Accumulation Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

FlowAccumulationD8 (flowDirRasterDataset As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Flow Accumulation D-infinity

ToolBox Implementation .NET Implementation

Generates a Flow Accumulation raster using the D-infinity algorithm (Tarboton, 1997).

**Inputs:**

- A D-infinity Flow Direction raster dataset.
- Stream Initiation Threshold value (optional)
- A D8 Flow Direction raster dataset (optional)

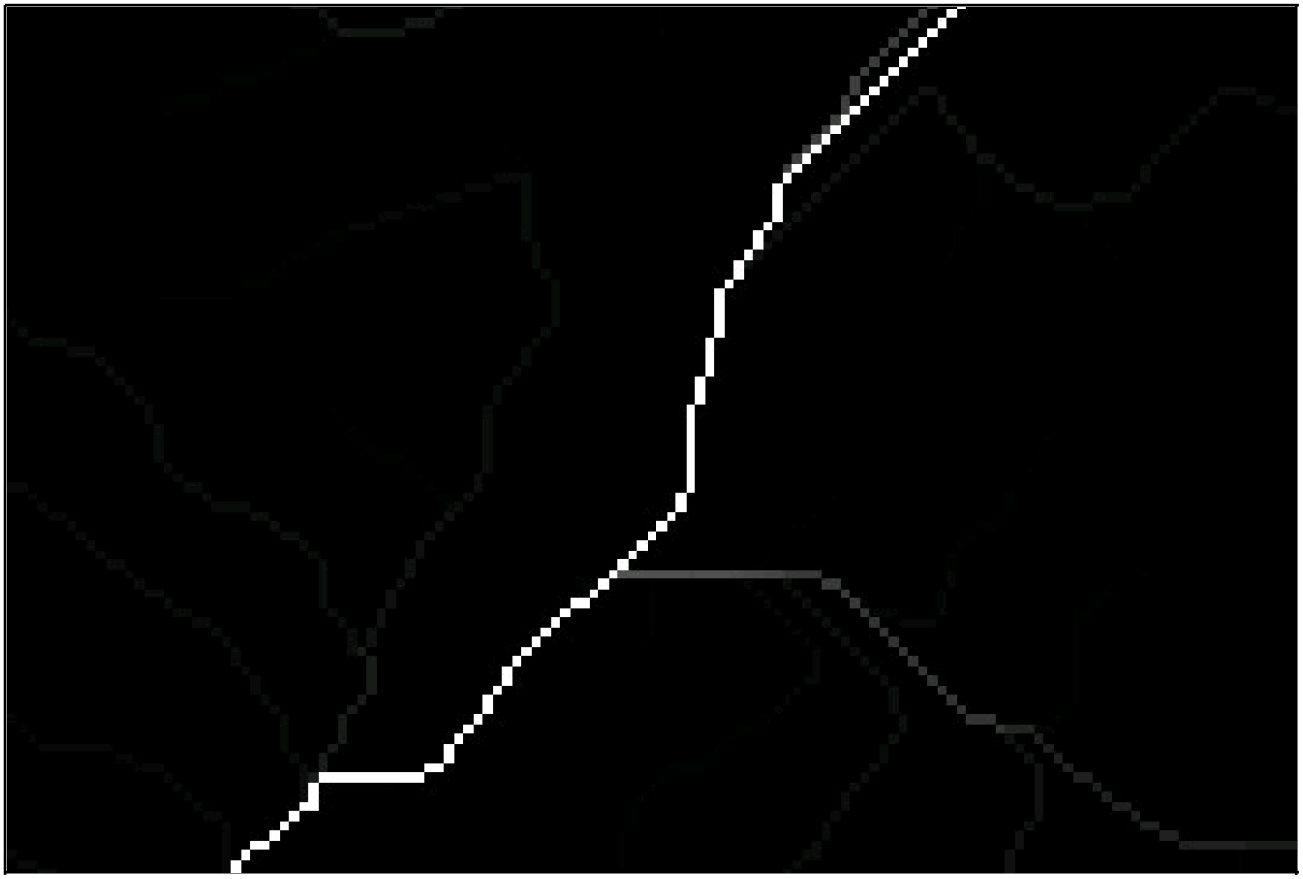
**Outputs:**

- A Flow Accumulation raster .

**Notes:**

- The value for each cell in the Flow Accumulation raster is the total number of cells which flow into it including the cell itself. So for a cell with no inflowing cells the value will be 1.
- The D-infinity flow algorithm allows flow divergence. This means that the value accumulated in a cell can be directed to one or two neighboring cells depending on the flow direction angle. This is a better representation of flow on divergent slopes.
- By applying a Threshold value to the Flow Accumulation raster we can determine the cells which form a Stream network. All cells with Flow Accumulation value above the threshold are considered to be part of a stream.
- The flow direction after Stream Initiation is better represented by the D8 single flow algorithm. The function allows for the specification of a Stream Initiation Threshold value and the relevant D8 Flow Direction raster to be used after the threshold accumulation value is reached.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
    - no extension specified - ESRI binary GRID
    - .img extension (for example raster1.img) - ERDAS IMAGINE image.
    - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
    - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Flow Accumulation D-infinity result (above) and Flow Accumulation D8 (below).
Note the dispersion of flow in the D-infinity result compared with the concentrated flow in D8.



**ToolBox implementation**

**Command line syntax**

ETS_GPFlowAccumulationDinf <Input D-inf Flow Direction Raster> <Out Flow Accumulation Raster> {Stream Initiation Threshold} {Input D8 Flow Direction Raster}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input D-inf Flow Direction Raster> | D-infinity Flow Direction raster layer or raster dataset |
| <Out Flow Accumulation Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| {Stream Initiation Threshold} | A Double representing the Threshold value |
| {Input D8 Flow Direction Raster} | D8 Flow Direction raster layer or raster dataset |

**Examples:**

- *ETS_GPFlowAccumulationDinf c:\test\FlowDirDinf.img c:\test\FlowAcc.img* - all accumulation will be done using the multiple flow D-infinity method
- *ETS_GPFlowAccumulationDinf c:\test\FlowDirDinf.img c:\test\FlowAcc.img* 1000 *c:\test\FlowDirD8.img* - accumulation below value of 1000 will be done using the D-infinity method, above 1000 the D8 Flow Directions will be used

**Scripting syntax**

ETS_GPFlowAccumulationDinf (Input D-inf Flow Direction Raster, Out Flow Accumulation Raster, Stream Initiation Threshold, Input D8 Flow Direction Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**Reference:**

Tarboton, D. G., (1997), "A New Method for the Determination of Flow Directions and Contributing Areas in Grid Digital Elevation Models", Water Resources Research, 33(2): 309-319

**.NET implementation**
(Go to TOP)

FlowAccumulationDinf (dinfFlowDirRaster As IRasterDataset2, sOutRaster As String, Optional iThreshhold As Integer = 0, Optional d8FlowDirRaster As IRasterDataset2 = Nothing) As IRasterDataset2

# Extract Outlets

ToolBox Implementation .NET Implementation

Extract Outlets (pour points) from a Flow Direction raster (D8 or D-infinity).

**Inputs:**

- A Flow Direction raster dataset (D8 or D-infinity).

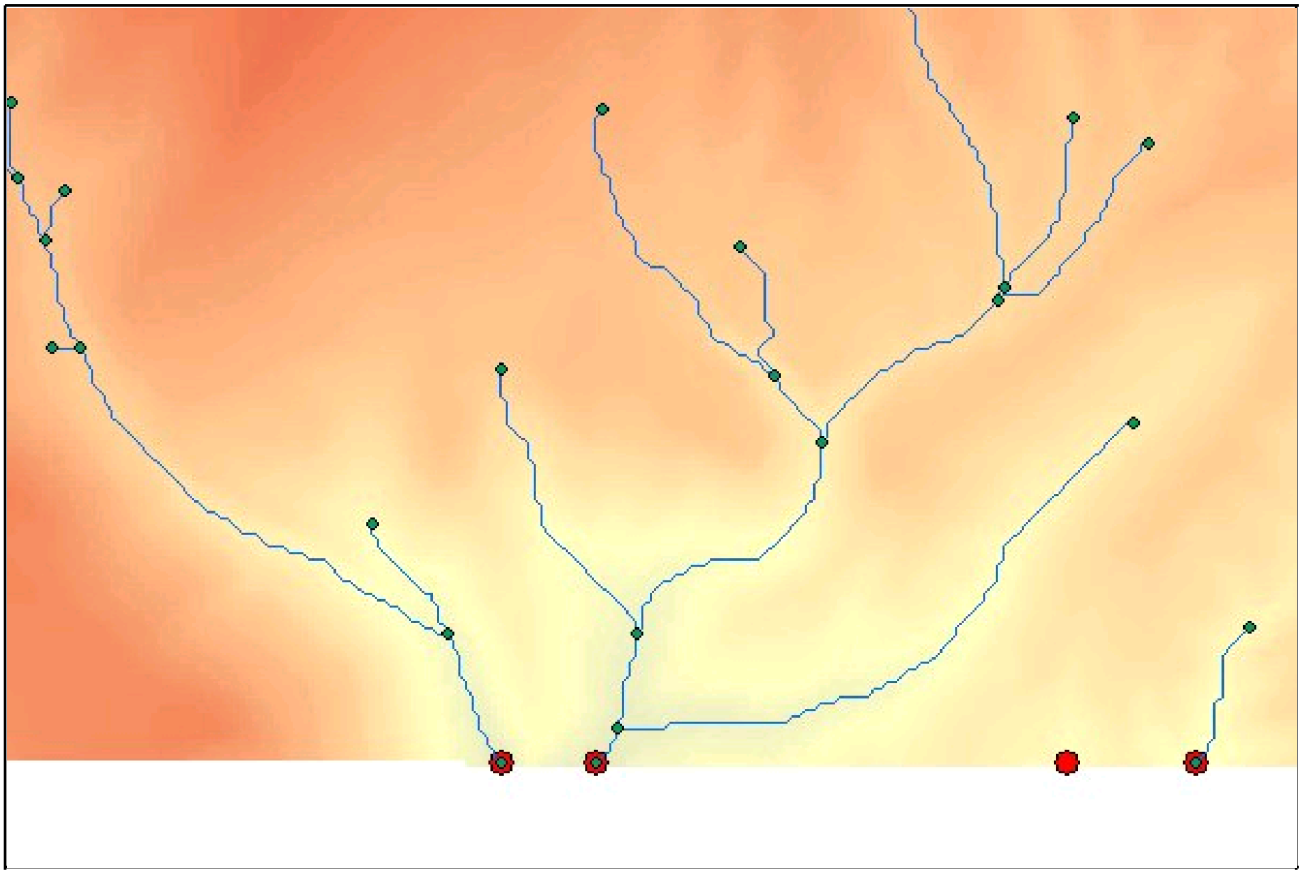
**Outputs:**

- A Point feature class representing the points at which the flow stops or leaves the extent of the DEM.
- Each outlet point is assigned an ID recorded in the "ET_Outlet" field of the attribute table

**Notes:**

- When the Flow Direction raster was created from a depressionless DEM, the outlet points should be at the border of the raster or next to areas of NoData.
- The extracted outlet points can be edited and used later for the delineation of Watersheds.
- See Hydrological functions for more information on the hydrological functionality.

An example of Extract Outlets output (the red points). For some of the extracted points the Flow Accumulation will be too low to delineate a stream.



**ToolBox implementation**

**Command line syntax**

ETS_GPExtractOutlets <Input Flow Direction Raster> <Out Outlet Feature Class>

**Parameters**

| Expression | Explanation |
| --- | --- |
| <Input Flow Direction Raster> | Flow Direction Raster layer or raster dataset |
| <Out Outlet Feature Class> | A String - the full name of the output feature class. |

**Examples:**

- *ETS_GPExtractOutlets c:\test\FlowDirection.img c:\test\Outlets.shp*

**Scripting syntax**

ETS_GPExtractOutlets (Input Flow Direction  Raster, Out Outlet Feature Class)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ExtractOutlets (flowDirRasterDataset As IRasterDataset2, sOutFName As String) As IFeatureClass

# Create Stream Raster

ToolBox Implementation

.NET Implementation

Delineates streams from a Flow Accumulation raster by applying a Stream Initiation threshold.

**Inputs:**

- A Flow Accumulation raster dataset.
- Stream Initiation Threshold - a double

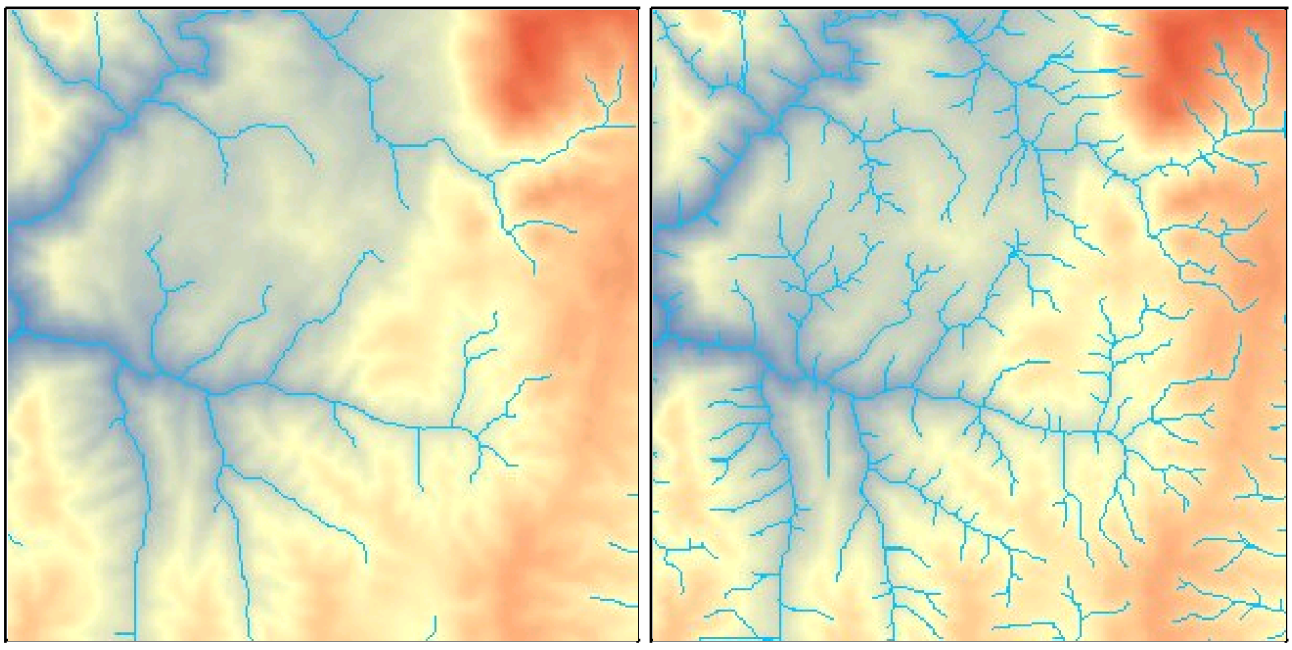
**Outputs:**

- A Stream raster representing the Stream Network.

**Notes:**

- The Stream Initiation Threshold represents the minimum contributing area (number of cells) required to initiate and maintain a Stream.
- The determination of the threshold value will depend on several factors including the type of the terrain, climate, soils and the DEM resolution.
- Reviewing existing data and maps for the area and experimentation with different values are helpful for the determination of the threshold value. Lower values will result in denser Stream Network.
- The Stream Network in the output raster is represented with the value of 1 on a background of NoData.
- If the input Flow Accumulation raster was created using the D8 method, the resulting Stream Network will be properly connected because a cell can only flow to one of its neighbors and the increase in the accumulation value is guaranteed.
- If the input Flow Accumulation raster was created using the D-infinity method it is possible to obtain unconnected streams in the output due to flow divergence. To avoid this when creating a Flow Accumulation raster with the D-infinity model, please use the optional Stream Initiation Threshold above which the accumulation will be determined using the D8 Flow Direction method. In this case a D8 Flow Direction raster is also required.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Create Stream Raster output. The left image shows the output at Threshold value of 1000 cells and the right at Threshold value of 100 cells.



**ToolBox implementation**

**Command line syntax**

ETS_GPCreateStreamRaster <Input Raster> <Out Stream Raster> <Stream Initiation Threshold>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Flow Accumulation raster layer or raster dataset |
| <Out Stream Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Stream Initiation Threshold> | A Double representing the number of Flow Accumulation cells at which a Stream is initiated. |

**Examples:**

- *ETS_GPCreateStreamRaster c:\test\FlowAcc.img c:\test\Stream.img 1000*

**Scripting syntax**

ETS_GPCreateStreamRaster (Input Raster, Out Stream Raster, Stream Initiation Threshold)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

CreateStreamRaster (flowAccumRaster As IRasterDataset2, sOutRaster As String, dThreshhold As Double) As IRasterDataset2

# Strahler Stream Order

ToolBox Implementation .NET Implementation

Assigns an order to each stream segment according to the system proposed by Strahler (1952). The order of the stream section starting at the stream head is assigned to 1. The order increases by 1 only when two sections of the same order intersect. For example if two sections of order 1 intersect, the following section will be assigned an order of 2. When two sections of different order intersect, the following section preserves the higher order.

**Inputs:**

- A Stream raster dataset.
- A D8 Flow Direction raster dataset

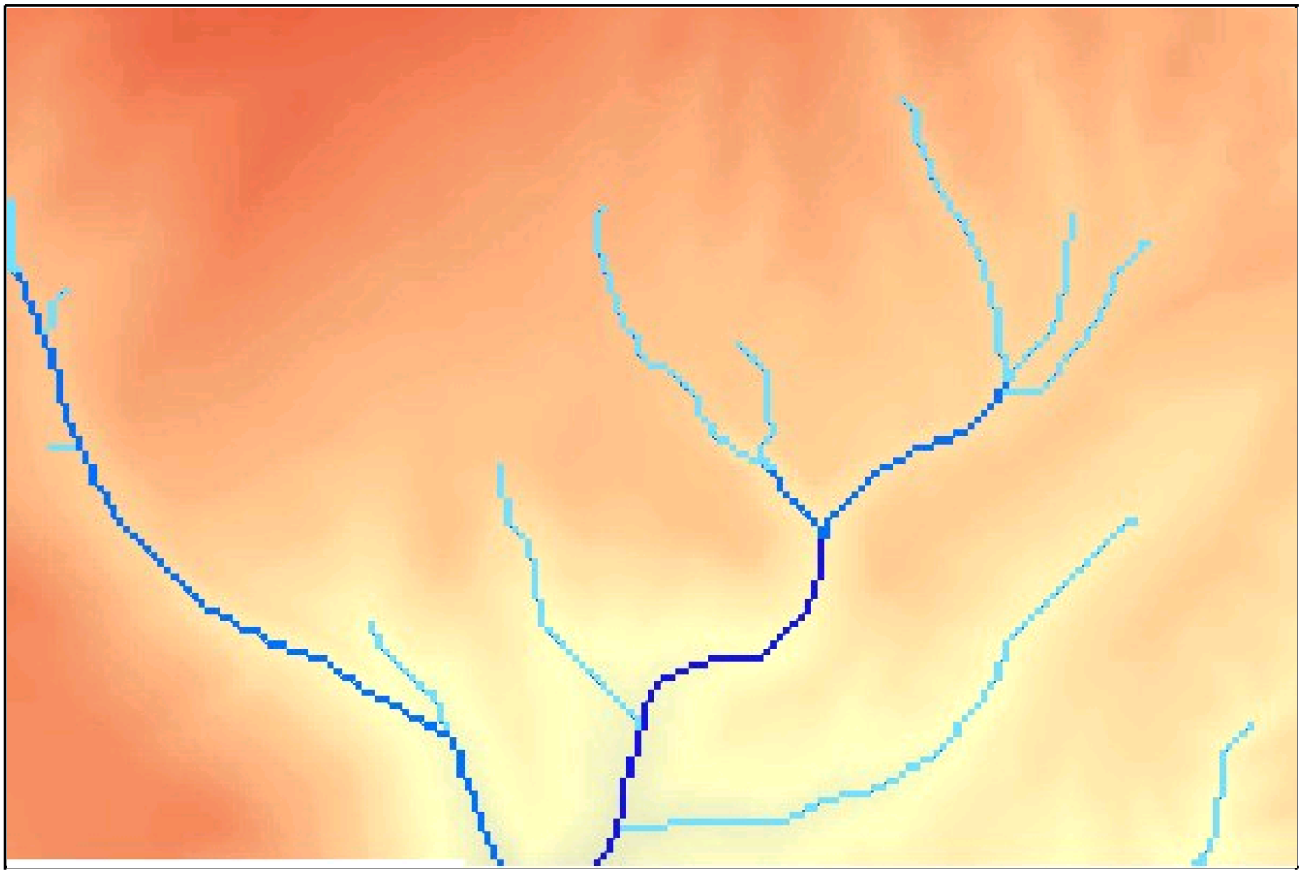
**Outputs:**

- A Strahler Stream Order raster with cell values indicating the assigned order for each stream section

**Notes:**

- The input Stream raster should be represented as values greater than or equal to one on a background of NoData
- The input Stream raster and the input D8 Flow Direction raster must have the same cell size and extent. In fact they should originate from the same DEM raster.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
    - no extension specified - ESRI binary GRID
    - .img extension (for example raster1.img) - ERDAS IMAGINE image.
    - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
    - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Strahler Stream Order result.



**ToolBox implementation**

**Command line syntax**

ETS_GPStrahlerStreamOrder <Input Stream Raster> <Input D8 Flow Direction Raster> <Out Strahler Stream Order Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Stream Raster> | A Stream raster layer or raster dataset |
| <Input D8 Flow Direction Raster> | A D8 Flow Direction raster layer or raster dataset. |
| <Out Strahler Stream Order Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Examples:**

- *ETS_GPStrahlerStreamOrder c:\test\Stream.img c:\test\D8FlowDir.img c:\test\Strahler.img*

**Scripting syntax**

ETS_GPStrahlerStreamOrder (Input Stream Raster, Input D8 Flow Direction Raster, Out Strahler Stream Order Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**

(Go to TOP)

StrahlerStreamOrder (streamRaster As IRasterDataset2, flowDirD8Raster As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Stream Link

ToolBox Implementation

.NET Implementation

Assigns unique values to sections of streams between junctions. A Link is each section of Stream between two junctions, the stream head and a junction or a junction and the outlet.

**Inputs:**

- A Stream raster dataset.
- A D8 Flow Direction raster dataset

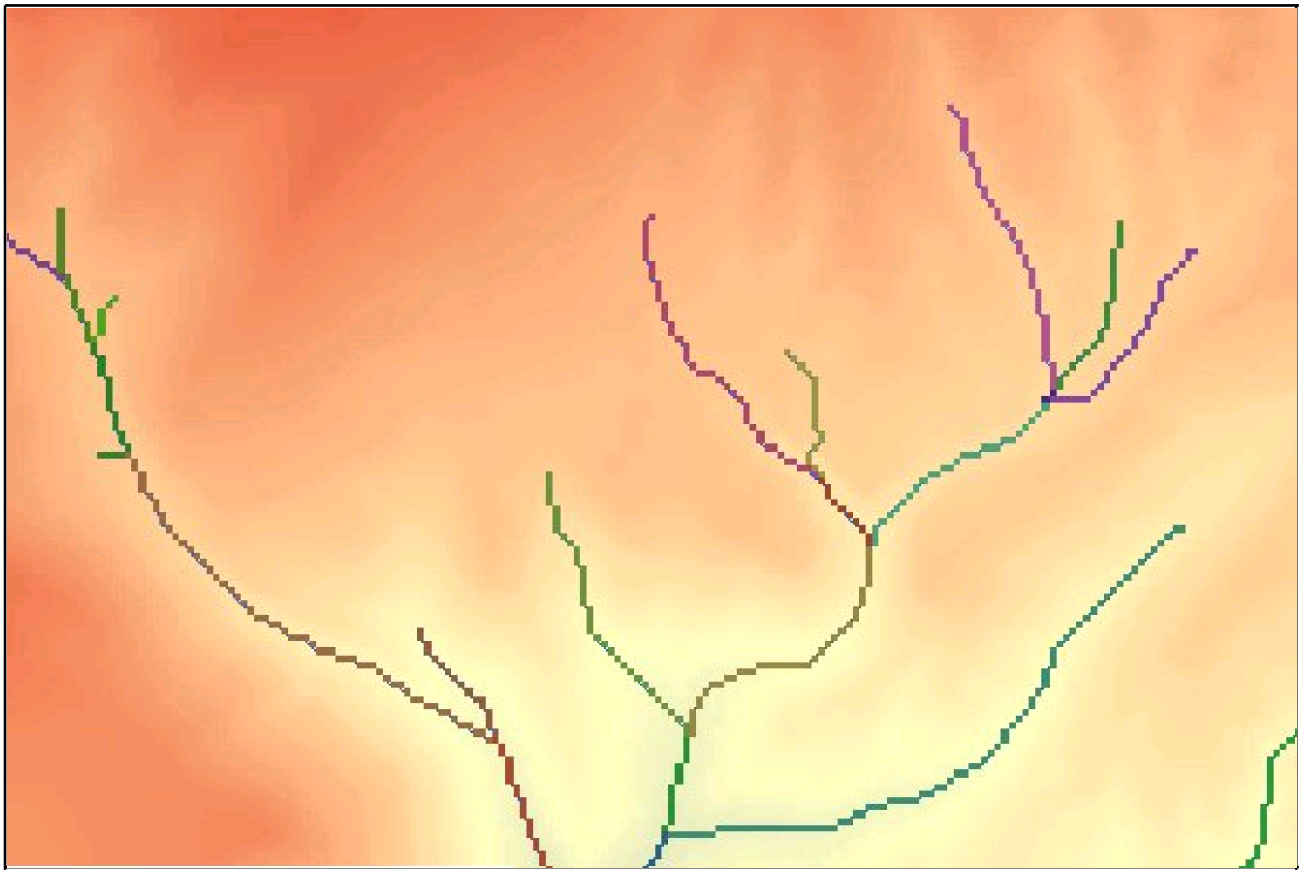
**Outputs:**

- A Stream Link raster. A unique integer value is assigned to each Stream section between junctions.

**Notes:**

- The input Stream raster should be represented as values greater than or equal to one on a background of NoData
- The input Stream raster and the input D8 Flow Direction raster must have the same cell size and extent. In fact they should originate from the same DEM raster.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
    - no extension specified - ESRI binary GRID
    - .img extension (for example raster1.img) - ERDAS IMAGINE image.
    - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
    - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Stream Link output.



**ToolBox implementation**

**Command line syntax**

ETS_GPStreamLink <Input Stream Raster> <Input D8 Flow Direction Raster> <Out Stream Link Raster>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Stream Raster> | Stream raster layer or raster dataset |
| <Input D8 Flow Direction Raster> | D8 Flow Direction raster layer or raster dataset |
| <Out Stream Link Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Examples:**

- *ETS_GPStreamLink c:\test\Stream.img c:\test\FlowDir.img c:\test\StreamLink.img*

**Scripting syntax**

ETS_GPStreamLink (Input Stream Raster, Input Flow Direction Raster, Out Stream Link Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

StreamLink (streamRaster As IRasterDataset2, flowDirD8Raster As IRasterDataset2, sOutRaster As String) As IRasterDataset2

# Stream Raster to Features

ToolBox Implementation .NET Implementation

Converts a Stream raster to a polyline feature class. Optionally creates a point feature class with the Nodes of the stream network.

**Inputs:**

- A Stream raster dataset.
- A D8 Flow Direction raster dataset

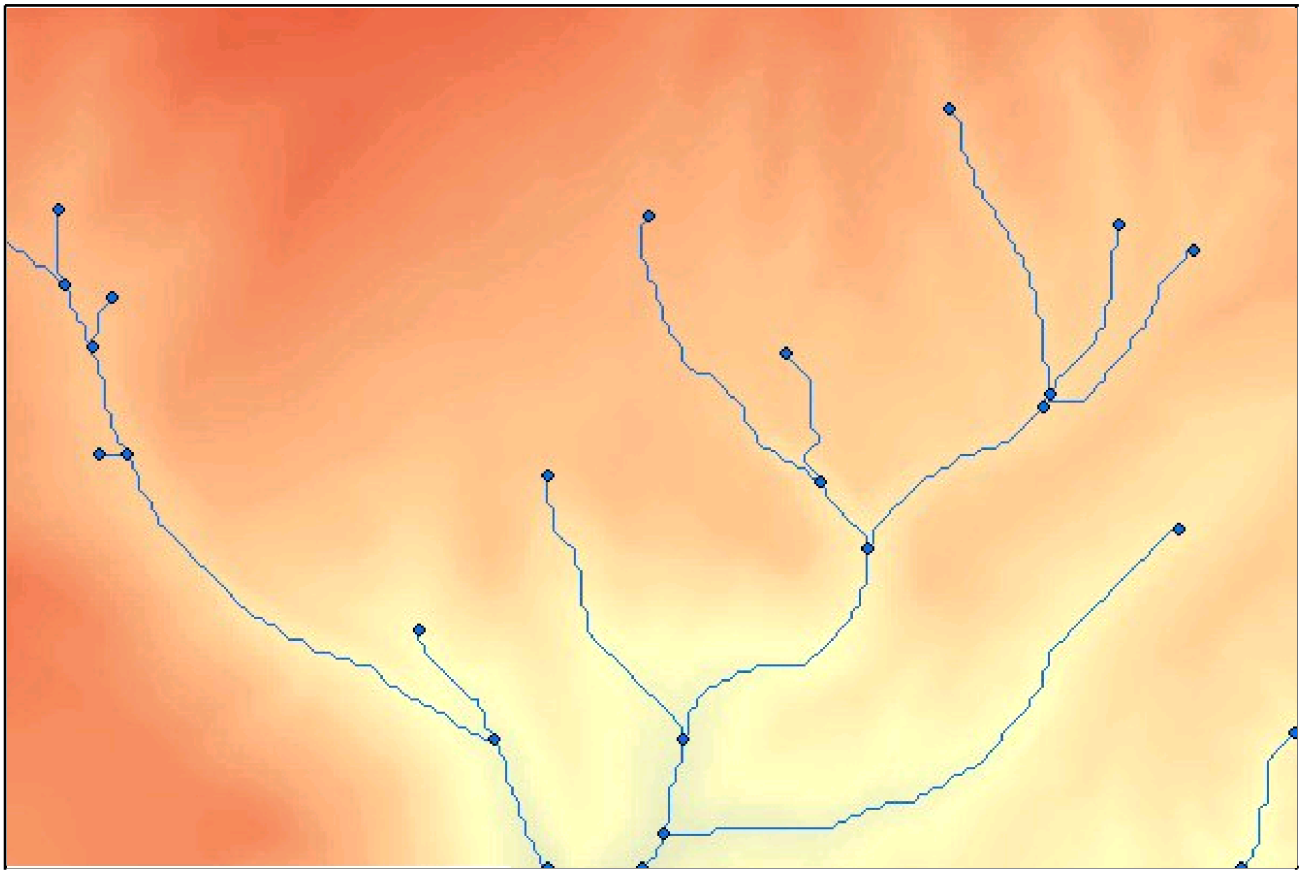
**Outputs:**

- A Polyline feature class. Each polyline represents a section of the Stream network between two junctions or from a junction to an outlet or from stream head to junction.
- Optional Point feature class. Each point represents a node of the Stream network - a junction or an outlet or a stream head.

**Notes:**

- The Stream feature class will have attributes for the unique Stream Link ID, Start Node and End Node corresponding with the Node ID from the Node Point feature class.
- The Node feature class will have attributs for the Unique Node ID.
- The Stream polylines will have a direction pointing downstream.
- The input Stream raster and the input D8 Flow Direction raster must have the same cell size and extent. In fact they should originate from the same DEM raster.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Stream Raster to Features output - Streams and Nodes.



**ToolBox implementation**

**Command line syntax**

ETS_GPStreamRasterToFeatures <Input Stream Raster> <Input D8 Flow Direction Raster> <Out Stream Feature Class> {Out Node Feature Class}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Stream Raster> | Stream raster layer or raster dataset |
| <Input D8 Flow Direction Raster> | D8 Flow Direction raster layer or raster dataset |
| <Out Stream Feature Class> | A String - the full name of the output Stream feature class. |
| {Out Node Feature Class} | A String - the full name of the output Node class. |

**Examples:**

- *ETS_GPStreamRasterToFeatures c:\test\Stream.img c:\test\FlowDir.img c:\test\Streams.shp*
- *ETS_GPStreamRasterToFeatures c:\test\Stream.img c:\test\FlowDir.img c:\test\Streams.shp c:\test\Nodes.shp*

**Scripting syntax**

ETS_GPStreamRasterToFeatures (Input Stream Raster, Input Flow Direction Raster, Out Streams Feature Class, Out Nodes Feature Class)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**

(Go to TOP)

StreamRasterToFeatures (streamRaster As IRasterDataset2, flowDirD8Raster As IRasterDataset2, sOutStream As String, Optional sOutNode As String = "") As IFeatureClass

# Snap Pour Points

ToolBox Implementation .NET Implementation

Snaps Pour points (Outlets) to raster cells within a specified distance depending on the selected Snap option.

**Inputs:**

- A Pour Points feature layer.
- Snap Raster dataset - the raster type depends on the selected Snap Option
- Snap Option - three possible values:
  - Snap to nearest Stream
  - Snap to lowest Elevation
  - Snap to highest Flow Accumulation
- Snap Distance - input points will be moved if a snap value is found within this distance in Map units.

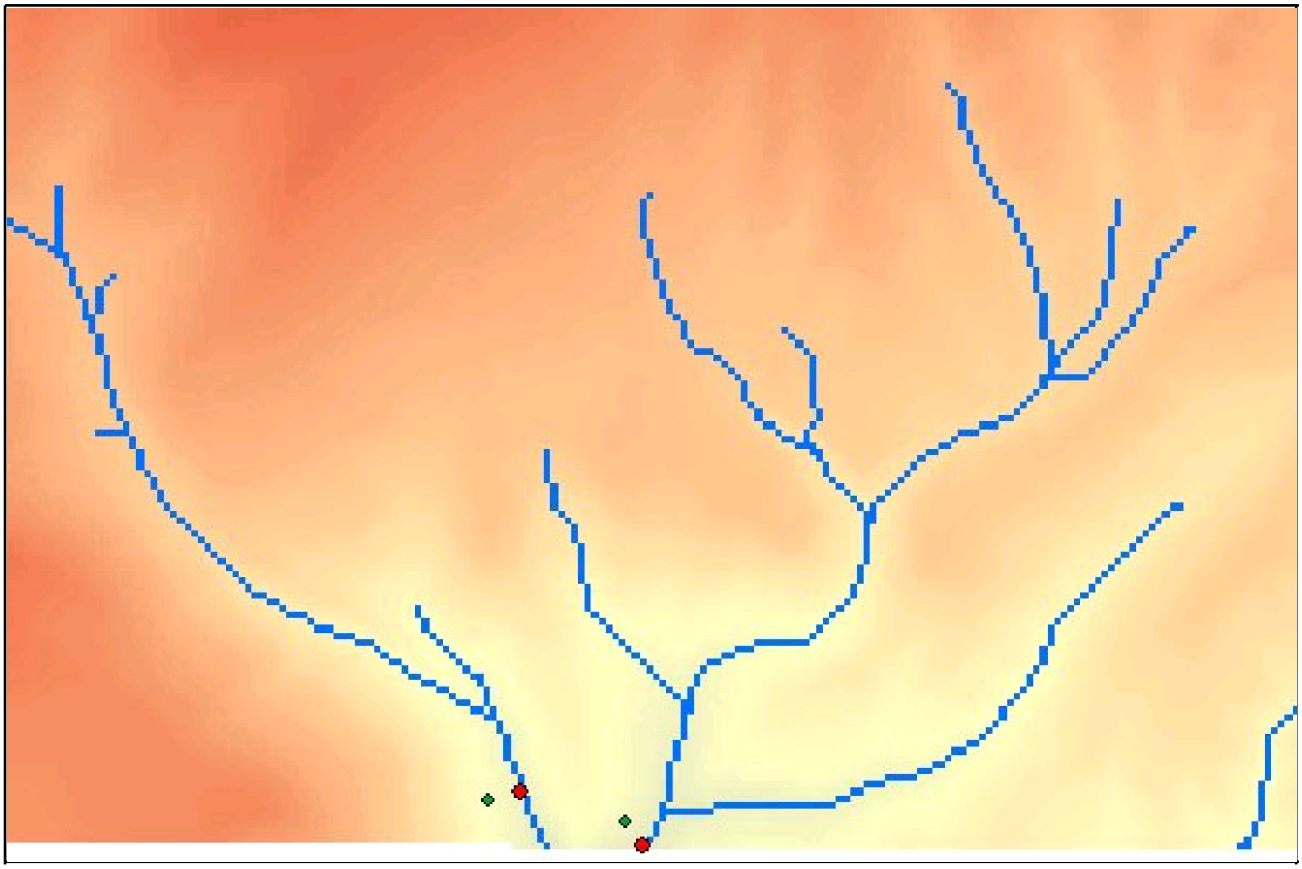
**Outputs:**

- A Point feature class with the moved points. A new field is added "ET_Snap" and will contain the distance with which the point was moved in Map units.

**Notes:**

- If the Snap Option is "Nearest Stream" a point will be moved to the center of the nearest cell in the Stream raster with value above 0 within the Snap Distance. If a point is within a Stream cell, it will not be moved.
- If the Snap Option is "Lowest Elevation" a point will be moved to the cell with the lowest value in the Elevation raster within the Snap Distance.
- If the Snap Option is "Highest Accumulation" a point will be moved to the cell with the highest value in the Flow Accumulation raster within the Snap Distance.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Snap Pour Points using the Stream option - initial points in green and after snapping in red.



**ToolBox implementation**

**Command line syntax**

ETS_GPSnapPourPoints <Input Pour Points Feature Class> <Input Snap Raster> <Out Feature Class> <Snap Option> <Snap Distance>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Pour Points Feature Class> | A Pour Points feature layer. |
| <Input Snap Raster> | A Snap Raster dataset depending on the selected Snap Option |
| <Out Feature Class> | A String - the full name of the output Pour Points feature class. |
| <Snap Option> | A String - possible values are "stream", "elevation" and "accumulation" |
| <Snap Distance> | A Double representing the Snap Distance in Map units |

**Examples:**

- *ETS_GPSnapPourPoints c:\test\Outlet.shp c:\test\Stream.img c:\test\SnapOutlet.shp "stream" 100*
- *ETS_GPSnapPourPoints c:\test\Outlet.shp c:\test\DEM.img c:\test\SnapOutlet.shp "elevation" 200*

**Scripting syntax**

ETS_GPSnapPourPoints (Input Pour Points Feature Class, Input Snap Raster, Out Feature Class, Snap Option, Snap Distance)

See the explanations above:

<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

SnapPourPoints (inFeatureClass As IFeatureClass, snapRasterDataset As IRasterDataset2, sOutFeature As String, snapOption As String, dDistance As Double) As IFeatureClass

# Watershed

ToolBox Implementation .NET Implementation

Delineates Watersheds (Drainage basins) from a D8 Flow Direction raster for a set of Outlet points.

**Inputs:**

- A D8 Flow Direction raster dataset.
- An optional Outlet Point Feature class.
- A Field in the Outlet feature class with integer IDs to be assigned to Watersheds.

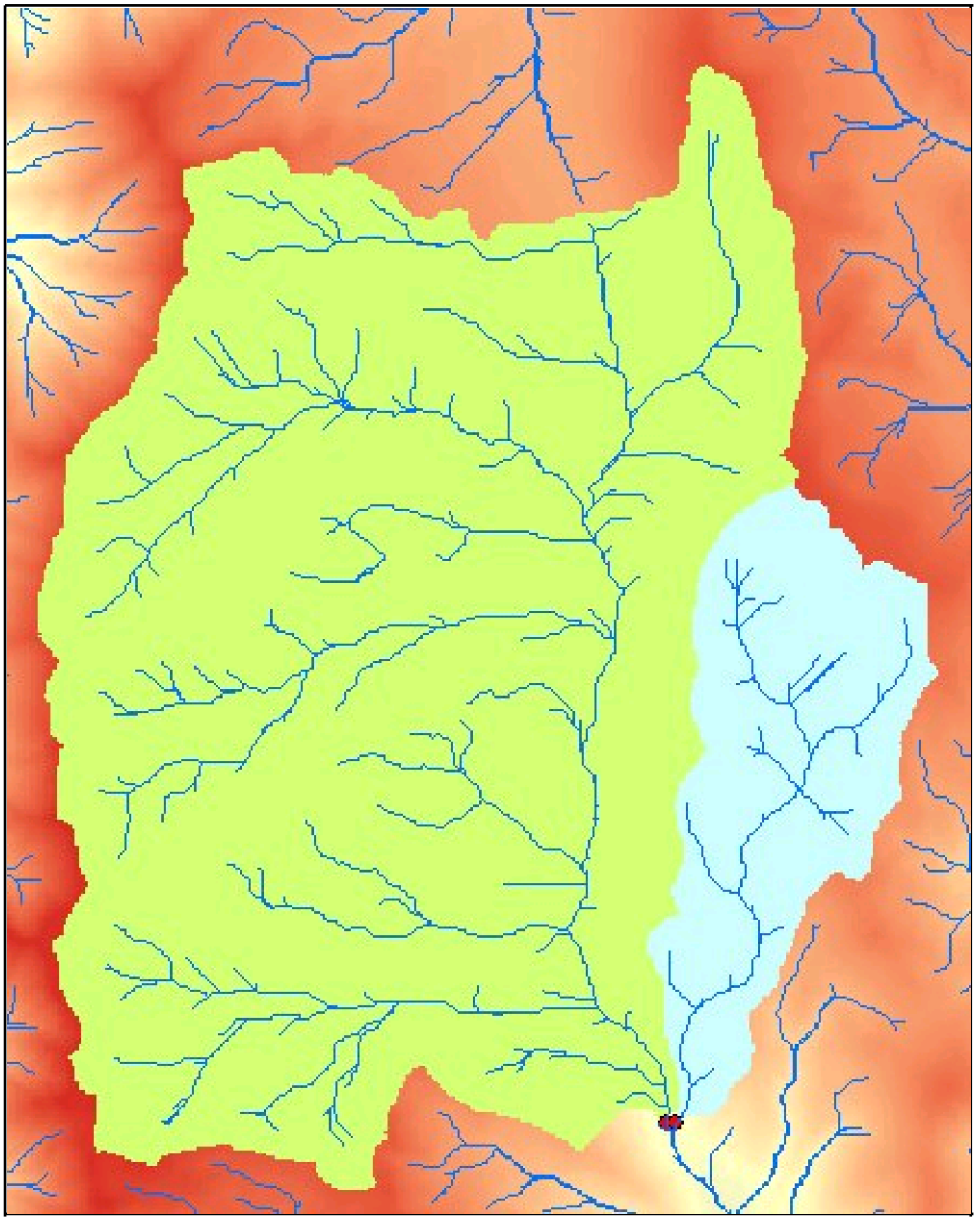
**Outputs:**

- A Watershed Raster representing the contributing areas for each outlet (integer raster).

**Notes:**

- Watersheds are delineated from the D8 Flow Direction raster. An optional Outlets feature class can be specified as input. In this case Watersheds are delineated for the Outlet points. An ID is assigned to each Watershed from a field in the Outlets feature class.
- If no Outlet points are specified, all Outlets for the DEM are determined (the points at which flow leaves the DEM) and Watersheds are delineated for them. In this case the Watersheds are assigned unique generated values starting from 1.
- If Outlet points are specified it is recommended to ensure that they are within cells with sufficient flow to delineate a reasonable Watershed. Otherwise it is possible only a few cells (or even just one) to be delineated.
- The Snap Pour Points function can be used to move the Outlet points to cells with higher flow within a specified distance based on the values of a reference raster.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Watershed output using Outlet points .



**ToolBox implementation**

**Command line syntax**

ETS_GPWatershed <Input D8 Flow Direction Raster> <Out Watershed Raster> {Input Outlet Feature Class} {Outlet ID Field}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input D8 Flow Direction Raster> | A D8 Flow Direction raster layer or raster dataset |
| <Out Watershed Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| {Input Outlet Feature Class} | A Point Feature Class with the Outlet points. |
| {Outlet ID Field} | A String - the name of an Integer Field in the Outlet Point Feature Class with values to be assigned to the delineated Watersheds. Field is required if an Outlet Feature Class is specified. |

**Examples:**

- *ETS_GPWatershed c:\test\FlowDir.img c:\test\Watershed.img*
- *ETS_GPWatershed c:\test\FlowDir.img c:\test\Watershed.img c:\test\Outlets.shp OutletID*

**Scripting syntax**

ETS_GPWatershed (Input D8 Flow Direction Raster, Out Watershed Raster, Input Outlets Feature Class, Input Outlet ID Field)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

Watershed (flowDirD8Raster As IRasterDataset2, sOutRaster As String, Optional outletFC As IFeatureClass = Nothing, Optional sOutletField As String = "") As IRasterDataset2

# Streams and Watershed from DEM

ToolBox Implementation .NET Implementation

Creates Stream features and optionally Nodes features and Watershed raster from a DEM.

**Inputs:**

- A DEM raster dataset.
- Flow Direction Method - D8 or D-infinity
- Stream Initiation Threshold

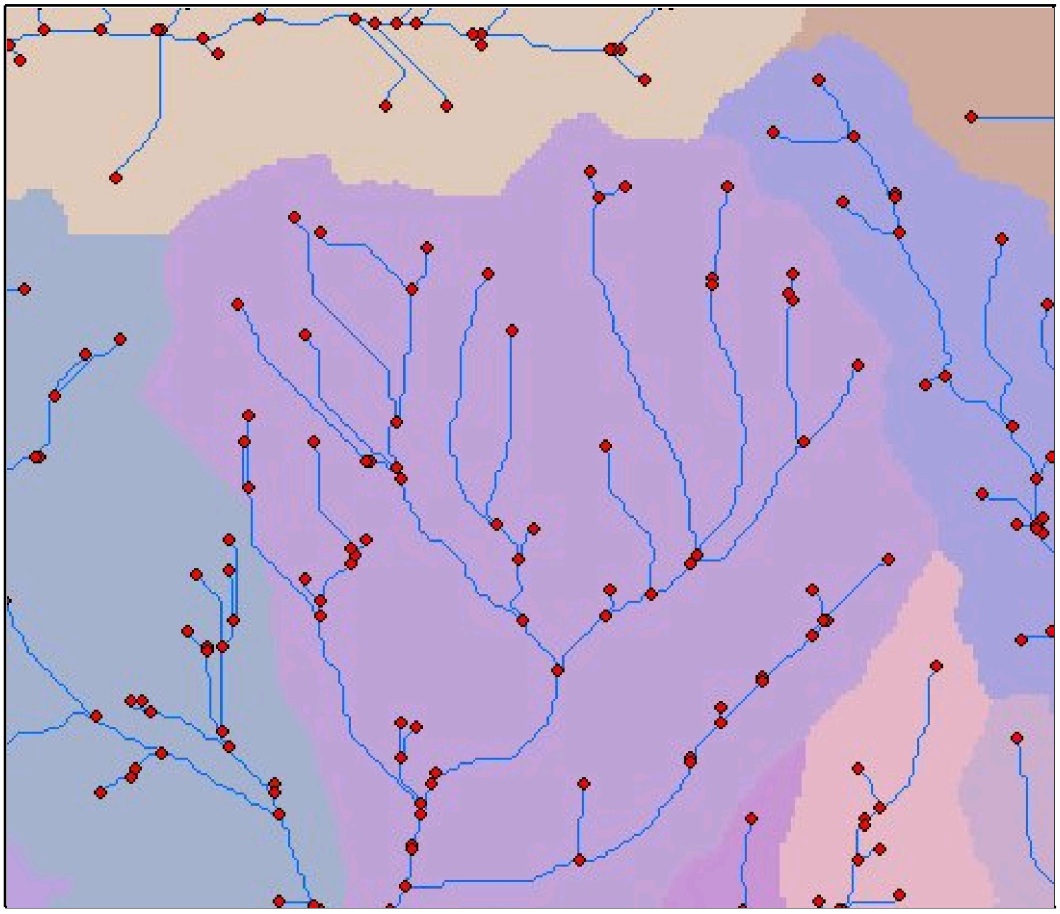
**Outputs:**

- A Polyline feature class. Each polyline represents a section of the Stream network between two junctions or from a junction to an outlet or from stream head to junction.
- Optional Point feature class. Each point represents a node of the Stream network - a junction or an outlet or a stream head.
- Optional Watershed Raster representing the contributing areas for each outlet (integer raster).

**Notes:**

- This function provides a straightforward way to create a Stream Network features and optionally Node features and Watershed raster from DEM. It calculates in the background the necessary Flow Direction and Flow Accumulation rasters depending on the selection of the Flow Direction method. The Watershed Raster represents Watersheds for all Outlet points determined from the Flow Direction raster and Watershed IDs are automatically generated.
- See Hydrological functions for more information on the hydrological functionality.
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.

An example of Streams And Watershed output - Streams and Nodes Features and Watershed raster.



**ToolBox implementation**

**Command line syntax**

ETS_GPStreamsAndWatershed <Input DEM Raster> <Flow Direction Method> <Stream Initiation Threshold> <Out Streams Feature Class> {Out Nodes Feature Class} {Out Watershed Raster}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input DEM Raster> | A DEM raster layer or raster dataset |
| <Flow Direction Method> | A String - possible values are "d8" and "d-inf" |
| <Stream Initiation Threshold> | A Double representing the number of Flow Accumulation cells at which a Stream is initiated. |
| <Out Streams Feature Class> | A String - the full name of the output Streams feature class. |
| {Out Node Feature Class} | A String - the full name of the output Node class. |
| {Out Watershed Raster] | A String - the full name of the output Watershed raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |

**Examples:**

- *ETS_GPStreamsAndWatershed c:\test\DEM.img "d8" 500 c:\test\Streams.shp*

  *ETS_GPStreamsAndWatershed c:\test\DEM.img "d-inf" 1000 c:\test\Streams.shp c:\test\Nodes.shp c:\test\Watershed.img*

**Scripting syntax**

ETS_GPStreamsAndWatershed (Input DEM Raster, Flow Direction Method, Stream Initiation Threshold, Out Streams Feature Class, Out Nodes Feature Class, Out Watershed Raster)

See the explanations above:
<> - required parameter
{} - optional parameter

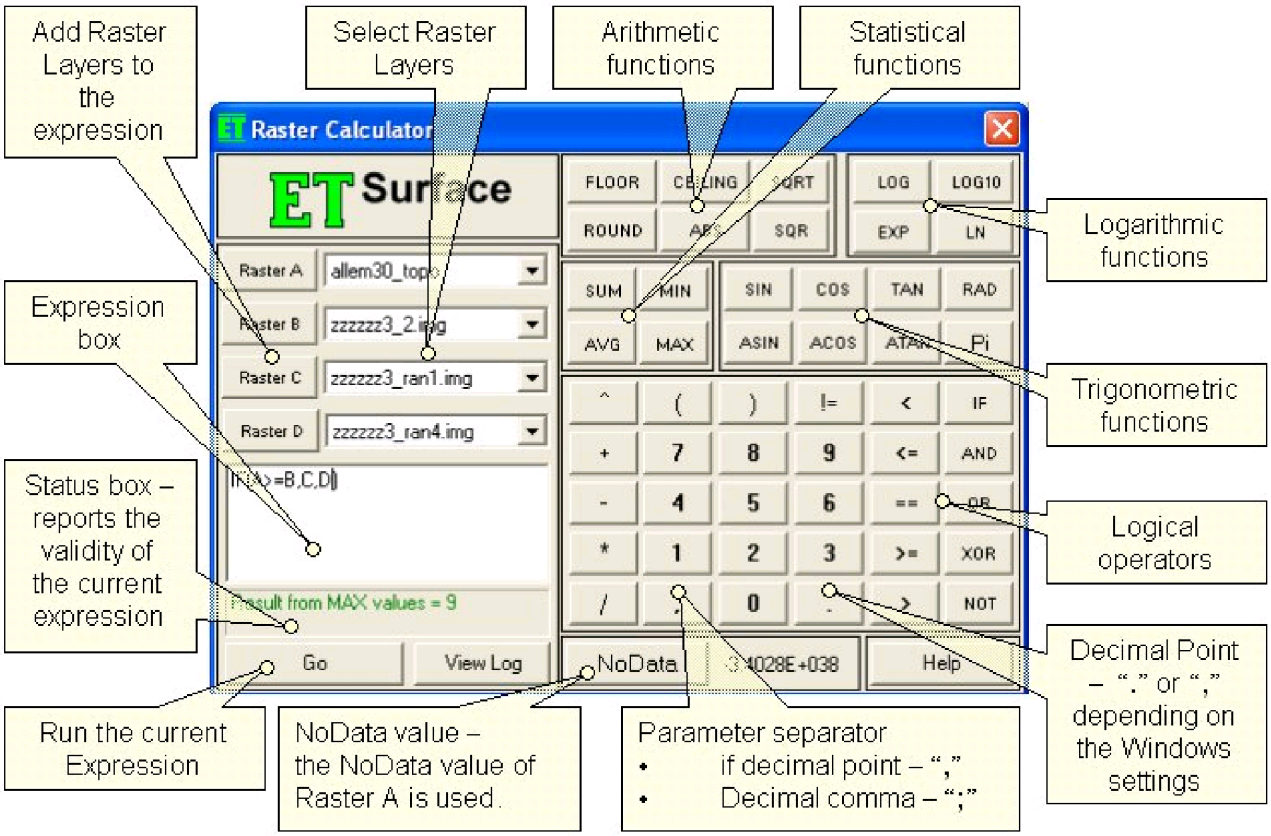
**.NET implementation**
(Go to TOP)

StreamsAndWatershed (demRaster As IRasterDataset2, sMethod As String, dThreshhold As Double, sOutStream As String, Optional sOutNode As String = "", _ Optional sWatershedRaster As String = "") As IFeatureClass

# Raster Calculator

ToolBox Implementation .NET Implementation

The Raster Calculator enables the user to perform complex mathematical calculation on rasters. The user can build expressions using large variety of arithmetic, logical, trigonometric, etc. functions provided. Up to 4 rasters can be used in a single expression. The expressions are evaluated by the Raster Calculator on the fly and the user is provided with the status of the formula as he/her builds it.




**Inputs:**

- **Rasters** - up to 4 rasters can be used. If the Raster Calculator is used from the GUI, the rasters are selected from the raster layers loaded in ArcMap. In the ToolBox implementation the input can be a raster layer or raster dataset. The 4 rasters are called Raster A, Raster B, Raster C and Raster D. Raster A is required, the other rasters are optional.
- **Expression** - the formula to be used for the calculation to be performed. For shortness the rasters should be entered with their letters in the expression - A for Raster A, B for Raster B, etc. All the functions available can be typed in the expression box or selected from the calculator buttons provided. The functions are not case sensitive - SIN, Sin and sin will be accepted as correct entries. Note that the operator for EQUAL is "==" and NOT "=" (which is operator for assignment). The syntax of all functions is discussed below.

Output:

- If the Raster Calculator is used from the GUI, the raster dataset created when an expression is executed is a temp raster and is stored in the temp folder of ET Surface. If you want to save it as a permanent raster, use the Export Data tool.
- If the ToolBox implementation is used, the user is asked for an output name and location and the raster dataset created is permanent.
- The output raster dataset will
  - Be FLOAT type
  - Have the cell size of the Raster A (if any of the other rasters used have a different cell size, it will be resampled)
  - The extent will be calculated as the intersection of the extents of the input rasters.

**Functions and operators**:

- Arithmetic
  - FLOOR - Returns the largest integer less than or equal to a number. FLOOR (3.66) = 3
  - CEILING - Returns the smallest integer greater than or equal to a number. CEILING (3.15)= 4
  - ROUND - Rounds a number to the nearest integer. ROUND (3.66) = 4, ROUND (3.15) = 3
  - ABS - Returns the absolute value of a number. ABS(-14) = 14
  - SQRT - Returns the square root of a number. SQRT(4) = 2
  - SQR - Returns the square of a number. SQR(2) = 4
- Trigonometric
  - Pi - adds constant Pi to the expression
  - RAD - converts an angle in degrees to an angle in radians.
  - SIN - Returns the sine of an angle. The argument should be in radians. SIN(Pi/6) = 0.5; SIN(RAD(30)) = 0.5

    - COS - Returns the cosine of an angle. The argument should be in radians. SIN(Pi/3) = 0.5, COS(RAD(60)) = 0.5
    - TAN - Returns the tangent of an angle. The argument should be in radians. TAN(RAD(45)) = 1
    - ASIN - Returns the angle (in radians) whose sine is the specified number. To convert the result to degrees multiply with 180/pi - ASIN(0.5)*180/Pi = 30
    - ASIN - Returns the angle (in radians) whose cosine is the specified number.
    - ATAN - Returns the angle (in radians) whose tangent is the specified number.
- Logarithmic
    - LOG - Returns the logarithm of a number.
    - LOG10 - Returns the base 10 logarithm of a specified number.
    - LN - Returns the natural logarithm of a number.
    - EXP - Returns e raised to the specified power.
- Statistics
    - SUM - Returns the sum of the arguments - SUM(1,2,3,4,5) = 15
    - MIN - Returns the minimum of the arguments - MIN(1,2,3,4,5) = 1
    - MAX - Returns the maximum of the arguments - MAX(1,2,3,4,5) = 5
    - AVG - Returns the average of the arguments - AVG(1,2,3,4,5) = 3
- Logical
    - IF - logical if. The syntax is IF(expression, true result, false result) - IF(2>1,100,200) = 100
    - != - Not equal - IF(2 != 1,100,200) = 100
    - == - equal - IF(2 == 1,100,200) = 200
    - AND - logical and - IF(2 > 1 AND 3 == 4,100,200) = 200
    - OR - logical or - IF(2 > 1 OR 3 == 4,100,200) = 100
    - XOR - logical xor - IF(2 > 1 XOR 3 == 4,100,200) = 100
    - NOT - logical not - IF(NOT(2 == 1),100,200) = 100

**Priority of the operators**

| Priority | Operators |
|---|---|
| 1 | AND, OR, XOR |
| 2 | ==, !=, <=, >=, <, > |
| 3 | +, - |
| 4 | *, / |
| 5 | ^ |

NoData Values

- The NoData value of the output raster will be equal to the NoData value of raster A
- If any of the rasters participating in an expression have NoData value in certain location, the output will have NoData value at this location
- The user can use NoData in an expression to set portions of the output raster to NoData. Example: IF(A<1500,NoData,A) - this expression will create a raster in which all cells with values less than 1500 will be assigned to NoData, the rest of the cells will have the values of the input raster A.
- To replace the NoData values of a raster with valid values use the Replace NoData function.

**ToolBox implementation**

**Command line syntax**

ETS_GPRasterCalculator <Raster_A> {Raster_B} {Raster_C} {Raster_D} <out_raster>,<Expression>

**Parameters**

| Expression | Explanation |
|---|---|
| <Raster_A> | A Raster dataset or Raster layer |
| {Raster_B} | A Raster dataset or Raster layer |
| {Raster_C} | A Raster dataset or Raster layer |
| {Raster_D} | A Raster dataset or Raster layer |
| <out_raster> | A String - the full name of the output raster (Araster with the same full name should not exist) |
| <Expression> | A String - the expression to be executed |

Example: ETS_GPRasterCalculator elev1 # # # c:\test\output.img IF(A > 1300, NoData, A)

**Scripting syntax**

ETS_GPRasterCalculator (Raster_A, Raster_B, Raster_C, Raster_D, out_raster, expression)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

RasterCalculator (rasterDatasetA As IRasterDataset2, rasterDatasetB As IRasterDataset2, rasterDatasetC As IRasterDataset2, rasterDatasetD As IRasterDataset2, sOutRaster As String, sExpression As String) As IRasterDataset2

The Raster Calculator uses the free MuParser © Ingo Berg (see license).

# Replace NoData

ToolBox Implementation .NET Implementation

Replaces the cells with NODATA values in the input rasters with the values of the corresponding cells of the replace raster or a constant.

**Inputs:**

- Input Raster dataset
- Output raster name and format
- Optional - Replace Raster dataset
- Optional - Constant value to replace the NODATA values.

**Output:**

- A raster dataset.

**Notes:**

- One of the optional parameters is needed
- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPReplaceNoData <Input Raster> <Out Raster> {Replace Raster} {Replace Value}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| {Replace Raster} | A Raster dataset or Raster layer - to be used as a replacement of the NODATA values |
| {Replace Value} | A Number - to be used as a replacement of the NODATA values |

**Scripting syntax**

ETS_GPReplaceNoData (Input Raster, Out Raster, Replace Raster, Replace Value)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**

(Go to TOP)

ReplaceNoData (inRasterDataset As IRasterDataset2, sOutRaster As String, Optional replaceRasterDataset As IRasterDataset2 = Nothing, Optional dValue As Double = 0) As IRasterDataset2

# Zonal Statistics

ToolBox Implementation .NET Implementation

Analyses the values of a raster within each polygon of the input polygon dataset. Calculates statistics for each polygon and adds them in the attribute table of the output.

**Inputs:**

- A polygon feature class
- A raster dataset for which the statistics will be calculated.
- A prefix for the field names. The function adds several fields to the polygon attribute table. The prefix will help the user to calculate the statistics of several rasters for the same polygons and add the results to the same attribute table. For example the user wants to calculate statistics for the slope and elevation for a polygon dataset. This requires two different rasters (Elevation and Slope). The function needs to be run twice
  - on the Elevation raster with prefix "EL"
  - on the Slope raster with prefix "SL"

The result will contain fields "EL_Min", "EL_Max", "SL_Min", "SL_Max", etc.

**Outputs:**

- A polygon dataset - a copy of the original polygons
- All the original attributes will be preserved.
- Depending on the type of the raster used, the following statistics will be calculated for each polygon and added to the attribute table of the output (XXX in the field names below replaces the prefix used in the function).
  - Integer and Floating point rasters
    - XXX_Count - the number of cells within a zone
    - XXX_Sum - the sum of the cell values within a zone
    - XXX_Min - the minimum values within a zone
    - XXX_Max - the maximum value within a zone
    - XXX_Range - the range of values within a zone
    - XXX_Mean - the average of the values within a zone
    - XXX_STD - the standard deviation of the values in a zone
    - XXX_Median - the median value in a zone
  - Integer rasters only
    - XXX_Major - the majority value (the value that appears most timesin a zone).
    - XXX_Minor - the minority value (the value that appears least times in a zone).
    - XXX_Var - variety (the number of unique values in a zone).

**Notes:**

- If the input polygon dataset has multi-part polygons, they will be exploded into single part polygons and the statistics will be calculated for the single part polygons.

**ToolBox implementation**

**Command line syntax**

ETS_GPZonalStatistics <Input Polygons> <Input Raster>  <Out Raster> <Prefix>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Polygons> | A Polygon layer or feature class |
| <Input Raster> | A  Raster dataset or Raster layer |

| | |
|---|---|
| <Out File Name> | A String - the full name of the output feature class. |
| <Prefix> | A String (maximum 3 characters) representing the prefix. See description above. |

**Scripting syntax**

ETS_GPZonalStatistics (Input Polygons, Input Raster, Out File Name, Prefix)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

ZonalStatistics (polygonFeatureClass As IFeatureClass, inRasterDataset As IRasterDataset2, sOutFeature As String, sPrefix As String) As IFeatureClass

# Focal Statistics

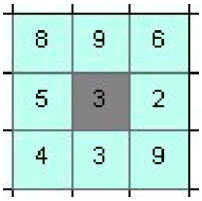
ToolBox Implementation .NET Implementation

Derives value for the output calls from a neighborhood of cells centered in the output cell. The neighborhood is with user defined shape (square or circle) and size (width or diameter) in number of cells.

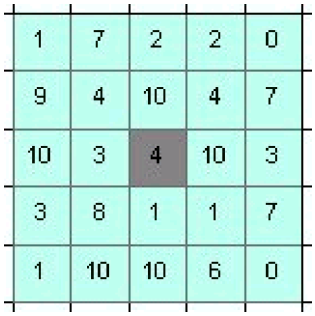
Neighborhoods:

Square

Width = 3

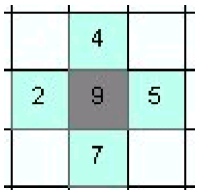



Width = 5

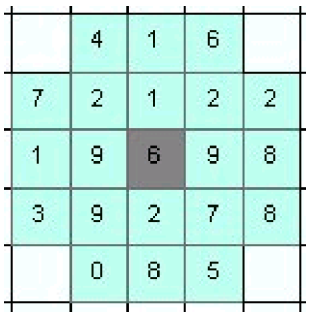



Circle

Diameter = 3




Diameter = 5




**Inputs:**

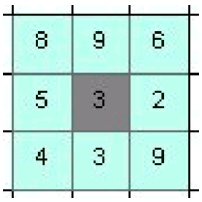
- Input raster dataset
- Output raster name and format
- Neighborhood shape
  - Square
  - Circle
- Neighborhood size (width for square and diameter for circle) in pixels/number of cells. The size needs to be an odd number in order to place the cell for which the calculations are performed in the center of the neighborhood.
- Statistics type .
  - Integer and Floating point rasters
    - Sum - the sum of the cell values within the neighborhood
    - Min - the minimum values within the neighborhood
    - Max - the maximum value within the neighborhood
    - Range - the range of values within the neighborhood
    - Mean - the average of the values within the neighborhood
    - STD - the standard deviation of the values in the neighborhood
    - Median - the median value in the neighborhood
  - Integer rasters only
    - Major - the majority value (the value that appears most times in the neighborhood).
    - Minor - the minority value (the value that appears least times in the neighborhood).
    - Variety - variety (the number of unique values in the neighborhood).

**Output:**

- A raster with type depending on the type of the input raster and the type of statistics performed.
  - Floating point input raster - Floating point output
  - Integer input raster
    - MEAN or STD statistics type - Floating point output
    - Any other type statistics - Integer output

**Example:**

Neighborhood Results for the center cell




- Sum = 49
- Min = 2
- Max = 9
- Range = 7
- Mean = 5.44
- STD = 2.54
- Median = 5
- Major = 3
- Minor = 2
- Variety = 7

**Notes:**

- Supported raster formats are File Geodatabase raster, Personal Geodatabase raster and file based raster formats (ESRI GRID, Erdas Imagine and TIFF).
- For file based rasters initially the name of the output raster defines the raster format
  - no extension specified - ESRI binary GRID
  - .img extension (for example raster1.img) - ERDAS IMAGINE image.
  - .tif extension (for example raster1.tif - Tagged Image File Format (TIFF) image.
  - The initial output raster format can be changed by selecting the desired output in the dialog.
- The input raster must be in a projected coordinate system.

**ToolBox implementation**

**Command line syntax**

ETS_GPFocalStatistics <Input Raster> <Out Raster> <Statistics Type> <Neighborhood Type> <Neighborhood Size>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Raster> | A Raster dataset or Raster layer |
| <Out Raster> | A String - the full name of the output raster (A raster with the same full name should not exist). The output raster type depends on the extension of the output file(see Notes above) |
| <Statistics Type> | A String - the type of the statistics to be calculated. Valid values<br>• Sum<br>• Min<br>• Max<br>• Range<br>• Mean<br>• STD<br>• Median<br>• Major<br>• Minor<br>• Variety |
| <Neighborhood Type> | A String - the neighborhood type. Valid values: Square and Circle |
| <Neighborhood Size> | An integer - the side of the square or the diameter of the circle in pixels/number of cells. This should be an odd number. |

**Scripting syntax**

ETS_GPFocalStatistics (Input Raster, Out Raster, Statistics Type, Neighborhood Type, Neighborhood Size)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

FocalStatistics (inRasterDataset As IRasterDataset2, sOutRaster As String, sStatsType As String, sNeighborhoodType As String, iSize As Integer) As IRasterDataset2

# Point Statistics

ToolBox Implementation

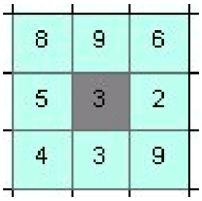
.NET Implementation

Analyses the values of a raster within user defined neighborhood around each point of the input point dataset. Calculates statistics for each point and adds them in the attribute table of the output.

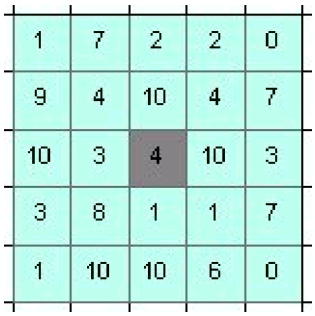
Neighborhoods (the grey cell is the cell in which the point is located):

Square

Width = 3

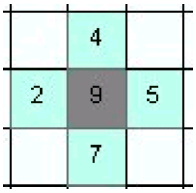



Width = 5

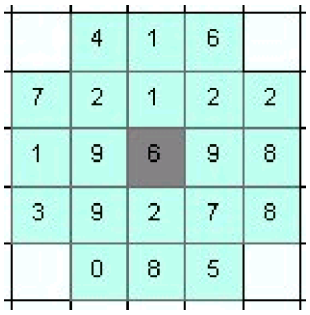



Circle

Diameter = 3




Diameter = 5




**Inputs:**

- A point feature class
- A raster dataset for which the statistics will be calculated.
- A prefix for the field names. The function adds several fields to the point attribute table. The prefix will help the user to calculate the statistics of several rasters for the same points and add the results to the same attribute table. For example the user wants to calculate statistics for the slope and elevation for a point dataset. This requires two different rasters (Elevation and Slope). The function needs to be run twice
    - on the Elevation raster with prefix "EL"
    - on the Slope raster with prefix "SL"

The result will contain fields "EL_Min", "EL_Max", "SL_Min", "SL_Max", etc.

**Outputs:**

- A point dataset - a copy of the original points
- All the original attributes will be preserved.
- Depending on the type of the raster used, the following statistics will be calculated for each point and added to the attribute table of the output (XXX in the field names below replaces the prefix used in the function).
    - Integer and Floating point rasters
        - XXX_Count - the number of cells within a zone
        - XXX_ZVal - the Z value of the cell in which the point is located.
        - XXX_Sum - the sum of the cell values within a zone
        - XXX_Min - the minimum values within a zone
        - XXX_Max - the maximum value within a zone
        - XXX_Range - the range of values within a zone
        - XXX_Mean - the average of the values within a zone
        - XXX_STD - the standard deviation of the values in a zone
        - XXX_Median - the median value in a zone
    - Integer rasters only
        - XXX_Major - the majority value (the value that appears most times in a zone).
        - XXX_Minor - the minority value (the value that appears least times in a zone).
        - XXX_Var - variety (the number of unique values in a zone).

**ToolBox implementation**

**Command line syntax**

ETS_GPPointStatistics <Input Points> <Input Raster>  <Out Raster> <Neighborhood Type> <Neighborhood Size> <Prefix>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Points> | A Point layer or feature class |
| <Input Raster> | A  Raster dataset or Raster layer |
| <Out File Name> | A String - the full name of the output feature class. |
| <Neighborhood Type> | A String - possible values are "SQUARE" and "CIRCLE" |
| <Neighborhood Size> | A Number - the diameter or side in pixels. |
| <Prefix> | A String (maximum 3 characters) representing the prefix. See description above. |

**Scripting syntax**

ETS_GPPointStatistics (Input Points, Input Raster  Out File Name, Neighborhood Type, Neighborhood Size, Prefix)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

PointStatistics (pointFeatureClass As IFeatureClass, inRasterDataset As IRasterDataset2, sOutFeature As String, sKernelType As String, iSize As Integer, sPrefix As String) As IFeatureClass

# PolylineZ Characteristics

ToolBox Implementation

.NET Implementation

Calculates several characteristics of the features from a PolylineZ feature class. The results are stored in fields in the polyline attribute table of the input or in a new polyline feature class

**Inputs:**

- A PolylineZ feature layer
- Linear precision - the number of digits after the decimal point for linear measures
- Angular precision - the number of digits after the decimal point for angular measures
- NODATA value - a number that represents undefined Z values. If the Z values of a geometry are interpolated from a surface and some of the vertices of the geometry are outside of the extent of the surface, they will not have Z values. Since ArcGIS does not accept NaN (Not a Number) values in Z enabled shapes, a numeric value is assigned to these vertices. If the Features To 3D function of ET Surface is used to derive the Z values, the NODATA value is -1000000000. When calculating Z characteristics this values need to be ignored. Segments that have a vertex with NODATA Z value will be ignored in the calculations.

**Outputs:**

The results are stored in the attribute table of the input dataset or in a new feature class. The linear measures are in the units of the spatial reference of the input dataset. The slope is measured in degrees (from -90 to +90). A negative value of the slope indicates downhill. The following fields are added to the attribute table

- [3D_Length] - the true 3D length of the polyline
- [2D_Length] - the 2D length of the polyline
- [Max_Z] - Maximum Z value
- [Min_Z] - Minimum Z value
- [Len_Up] - distance uphill
- [Len_Down] - distance downhill
- [H_Up] - total increase in height
- [H_Down] - total decrease in height
- [Av_S_Up] - average slope uphill
- [Max_S_Up] - maximum slope uphill
- [Av_S_Down] - average slope downhill
- [Max_S_Down] -maximum slope downhill

**ToolBox implementation**

**Command line syntax**

ETS_GPPolylineZChars <Input Dataset> <Out Feature Class> {Linear Precision}, {Angular Precision}, {NODATA Value}

Parameters

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polyline feature layer or feature class |
| <Out Feature Class> | A String - the full name of the outputfeature class |
| {Linear Precision} | An Integer representing the number of digits after the decimal point for linear measures |
| {Angular Precision} | An Integer representing the number of digits after the decimal point for angular measures. |
| {NODATA Value} | A Double - see explanations above |

Scripting syntax

ETS_GPPolylineZChars (Input Dataset, Out Feature Class, Linear Precision, Angular Precision, NODATA Value)

See the explanations above:
<> - required parameter

{} - optional parameter

**.NET implementation**
(Go to TOP)

PolylineZChars (inFeatureClass As IFeatureClass, sOutFeature As String, Optional linearPrecision As Integer = 2, Optional angularPrecision As Integer = 2, Optional dNoData As Double = -1000000000) As IFeatureClass

# Split PolylineZ based on slope direction change

ToolBox Implementation .NET Implementation

Splits the polylines of PolylineZ dataset in the points of change of the direction of the slope. Several characteristics of the resulting PolylineZ features are calculated and populated in the attribute table. Optionally a point feature class that contains the turning points (Ridges and Valleys) can be created.

**Inputs:**

- A PolylineZ feature layer
- Name for the output dataset.
- Linear precision - the number of digits after the decimal point for linear measures
- Angular precision - the number of digits after the decimal point for angular measures
- NODATA value - a number that represents undefined Z values. If the Z values of a geometry are interpolated from a surface and some of the vertices of the geometry are outside of the extent of the surface, they will not have Z values. Since ArcGIS does not accept NaN (Not a Number) values in Z enabled shapes, a numeric value is assigned to these vertices. If the Features To 3D function of ET Surface is used to derive the Z values, the NODATA value is -1000000000. When calculating Z characteristics this values need to be ignored. Segments that have a vertex with NODATA Z value will be ignored in the calculations.
- Units for slope calculations (percent or degrees)
- Optional. Name of the output point feature class.

**Outputs:**

1. A PolylineZ feature class. Each polyline have segments with positive (upwards), negative (downwards) or Zero slope. The attribute table will contain all original attributes. The following fields will be added:
    - [3D_Length] - the true 3D length of the polyline
    - [2D_Length] - the 2D length of the polyline
    - [Max_Z] - Maximum Z value
    - [Min_Z] - Minimum Z value
    - [DeltaH] - the difference in Z between the start and end point of the polyline.
    - [Slope_Dir] - the direction of the slope for this polyline. The values can be Up, Down, Flat or Undefined (for segments that contain NODATA Z values)
    - [Max_Slope] - the maximum slope
    - [Min_Slope] - the minimum slope
    - [Slope] - average slope calculated based on the Z values of the start and end points of the polyline and the 2D length.
2. A Point dataset that contains the turning points (Ridges and Valleys). The following fields will be included in the attribute table
    - [ET_Z] - the Z value of the point
    - [ET_Type] - the type of the turning point - Ridge or Valley.

**ToolBox implementation**

**Command line syntax**

ETS_GPSplitPolylineBySlope <Input Dataset> <Out Feature Class> <Slope Units> {Output Points} {Linear Precision}, {Angular Precision}, {NODATA Value}

Parameters

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polyline feature layer or feature class |
| <Out Feature Class> | A String - the full name of the outputfeature class |

| | |
|---|---|
| <Slope Units> | A String representing the unit of the slope. Valid values - "Degrees" and "Percent" |
| {Output Points} | A String - the full name of the output feature class that will contain the turning points (Ridges and Valleys). |
| {Linear Precision} | An Integer representing the number of digits after the decimal point for linear measures |
| {Angular Precision} | An Integer representing the number of digits after the decimal point for angular measures. |
| {NODATA Value} | A Double - see explanations above |

Scripting syntax

ETS_GPSplitPolylineBySlope (Input Dataset,  Out Feature Class, Slope Units, Out Linear Precision, Angular Precision, NODATA Value)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

SplitPolylinesZBySlope (inFeatureClass As IFeatureClass, sOutFeature As String, slopeUnits As String, Optional outputPoints As String = "", Optional linearPrecision As Integer = 2, Optional angularPrecision As Integer = 2, Optional dNoData As Double = -1000000000) As IFeatureClass

# Polygon 3D Characteristics

ToolBox Implementation .NET Implementation

Calculates some 3D characteristics of the features from a polygon dataset based on a reference ESRI TIN or PolygonZ TIN.

**Inputs:**

- A Polygon feature class
- An ESRI TIN or PolygonZ TIN

**Outputs:**

- New Polygon feature class.
- The original attributes are preserved.
- New fields are added to the attribute table of the Multipatch feature class.
  - ET_ElMin - the minimum Z value
  - ET_ElMax - the maximum Z value
  - ETSlopeMax - the maximum slope
  - ET_AreaZ - the 3D area
  - ET_Area - the 2D area

**ToolBox implementation**

**Command line syntax - two different toolbox tools available depending on the type of the input TIN. Check the color coding for specifics.**

ETS_GPPolygon3DCharsEsriTIN <Input Dataset> <Input ESRI TIN> <Out Feature Class>

ETS_GPPolygon3DCharsPolygonZTIN <Input Dataset> <Input PolygonZ TIN> <Out Feature Class>

**Parameters**

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polygon feature layer or feature class |
| <Input ESRI TIN> | An ESRI TIN layer or dataset |
| <Input PolygonZ TIN> | A PolygonZ TIN (feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |

**Scripting syntax**

ETS_GPPolygon3DCharsEsriTIN (Input Dataset, Input ESRI TIN, Out Feature Class)

ETS_GPPolygon3DCharsPolygonZTIN (Input Dataset, Input PolygonZ TIN, Out Feature Class)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

Polygon3DCharsEsriTIN (inFeatureClass As IFeatureClass, inTin As ITin, sOutFeature As String) As IFeatureClass

Polygon3DCharsPolygonZTIN (inFeatureClass As IFeatureClass, polygonZTin As IFeatureClass, sOutFeature As String) As IFeatureClass

# Analyze TIN

ToolBox Implementation .NET Implementation

Calculates several characteristics for each triangle of a PolygonZ TIN

- Minimum elevation
- Maximum elevation
- Mean elevation
- Slope - identifies the slope, or maximum rate of elevation change for each triangle
- Aspect - the values of the output field represent the compass direction of the slope (horizontal direction in which a slope faces); 0 is true north, a 90 degree aspect is to the east etc. For flat triangles (slope = 0) the value of -1 is assigned for the aspect
- Hill Shade- computes the brightness of each triangle based on a light source location.
- 3D Area

**Inputs:**

- A PolygonZ TIN feature layer
- Characteristics to be calculated
  - Parameters for Hill Shade ( if Hill Shade option is selected )
    - azimuth - the azimuth angle of the light source. The azimuth is expressed in positive degrees from 0 to 360, measured clockwise from the north. The default is 315 degrees.
    - altitude - the altitude angle of the light source above the horizon. The altitude is expressed in positive degrees, with 0 degrees at the horizon and 90 degrees directly overhead. The default is 45 degrees.

**Outputs:**

- New polygon Z feature class. Several fields are added to the attribute table, depending on the options selected
- ET_ElMin - minimum elevation values for each triangle
- ET_ElMax - maximum elevation values for each triangle
- ET_ElMean - mean elevation values for each triangle
- ET_Slope_D - the slope (maximum rate of elevation change) of each triangle in Degrees (from 0 to 90)
- ET_Sope_P - the slope (maximum rate of elevation change) of each triangle in percents
- ET_Aspect - the aspect (compass direction of the slope - 0 is North, 90 degrees - East, 180 degrees - South, 270 - West) of each triangle
- ET_ACode - aspect categories
  - N - North ( 0 to 22.5 and 337.5 to 360)
  - NE - North East (22,5 to 67.5)
  - E - East (67.5 to 112.5)
  - SE - South East (112.5 to 157.5)
  - S - South (67.5 to 112.5)
  - SW - South West (202.5 to 247.5)
  - W - West (247.5 to 292.5)
  - NW - North West (292.5 to 337.5)
  - U - Undefined - Slope = 0
- ET_AreaZ - the 3D area of each triangle

**Notes:**

- If the PolygonZ TIN is created using ET Surface, you need to analyze the tin only if you need to obtain the values for Hill Shade or calculate them for different Azimuth and/or Altitude of the light source
- If you have PolygonZ TIN created with ET GeoWizards (or some other application), you will need to use the Analyze TIN function in order to get correct parameters for use with the ET Surface functions.

**ToolBox implementation**

**Command line syntax**

ETS_GPAnalyzeTIN <Input TIN>  <Out Feature Class>  {Light Azimuth} {Light Altitude}

**Parameters**

| Expression | Explanation |
|---|---|
| <Input TIN> | A PolygonZ TIN (feature layer or feature class) |
| <Out Feature Class> | A String - the full name of the output feature class. |
| {Light Azimuth} | A Double representing azimuth of the light source (0 to 360). 0 indicates North, 90 - East, 180 - South, 270 - West |
| {Light Altitude} | A Double representing the altitude of the light source  in degrees (0 to 90) |

**Scripting syntax**

ETS_GPAnalyzeTIN (Input TIN, Out Feature Class, Light Azimuth, Light Altitude)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

AnalyzePolygonZTIN(inFeatureClass As IFeatureClass, sOutFeature As String, Optional dAzimuth As Double = 315, Optional dAltitude As Double = 45) As IFeatureClass

# Multiply Zs

ToolBox Implementation .NET Implementation

Multiplies the Z values of the geometries of the input Z dataset with an user specified factor.

**Inputs:**

- A PointZ, PolylineZ, PolygonZ, Multipatch feature layer

**Outputs:**

- New feature class of the same type as the input. The Z values of all points/vertices multiplied by the user specified factor.

Notes:

- Can be used to change the Z units of the geometries
- If applied to a PolylineZ feature class with calculated Z characteristics, you will need to recalculate the characteristics to reflect the changes of the Z values.
- If applied on a PolygonZ TIN, you need toanalyze the TIN again to reflect the changes.

**ToolBox implementation**

**Command line syntax**

ETS_GPMultiplyZs <Input Dataset>  <Out Feature Class> <Multiply With>)

Parameters

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polyline feature layer or feature class |
| <Out Feature Class> | A String - the full name of the outputfeature class |
| <Multiply With> | A Number representing the multiply value. |

Scripting syntax

ETS_GPMultiplyZs (Input Dataset,  Out Feature Class, Multiply With)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

MultiplyZs (inFeatureClass As IFeatureClass, sOutFeature As String, dValue As Double) As IFeatureClass

# Offset Zs

ToolBox Implementation .NET Implementation

Adds/subtracts a user specified value from the Z values of the geometries of the input Z dataset.

**Inputs:**

- A PointZ, PolylineZ, PolygonZ, Multipatch feature layer

**Outputs:**

- New feature class of the same type as the input. The user specified value will be added to the Z values of all points/vertices.

Notes:

- If applied to a PolylineZ feature class with calculated Z characteristics, you will need to recalculate the characteristics to reflect the changes of the Z values.
- If applied on a PolygonZ TIN, you need toanalyze the TIN again to reflect the changes.

**ToolBox implementation**

**Command line syntax**

ETS_GPOffsetZs <Input Dataset>  <Out Feature Class> <Offset Value>)

Parameters

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polyline feature layer or feature class |
| <Out Feature Class> | A String - the full name of the outputfeature class |
| <Offset Value> | A Number representing the Offset value |

Scripting syntax

ETS_GPOffsetZs (Input Dataset,  Out Feature Class, Offset  Value)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**
(Go to TOP)

OffsetZs (inFeatureClass As IFeatureClass, sOutFeature As String, dValue As Double) As IFeatureClass

# Clean Contour Gaps

ToolBox Implementation .NET Implementation

Cleans the gaps in a polyline dataset representing contours.

**Inputs:**

- A Polyline dataset
- A field representing the elevation value of the contours
- Tolerance - the gaps smaller than this tolerance will be closed.

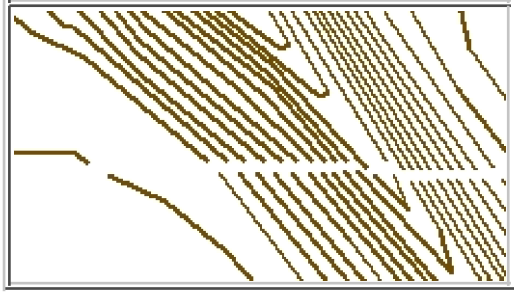
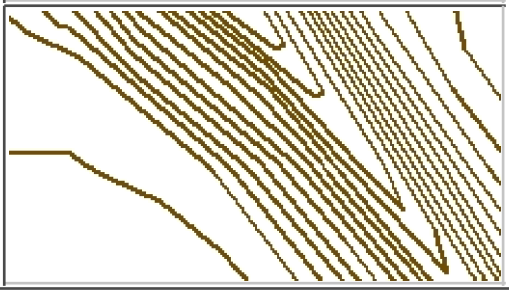
**Outputs:**

- New polyline feature class. The gaps in the contours that a smaller than the selected tolerance are closed.

**Notes:**

- The function is designed specifically for contours, but it can be used on datasets representing different features.
- Always inspect the results before accepting them as valid.

**Example :**

| Input Contours | Resulting Polylines |
|---|---|
|  |  |

**ToolBox implementation**

**Command line syntax**

ETS_GPCleanContourGaps <Input Dataset>  <Out Feature Class> <Gap Size> <Height Field>

Parameters

| Expression | Explanation |
|---|---|
| <Input Dataset> | A Polyline feature layer or feature class |
| <Out Feature Class> | A String - the full name of the outputfeature class |
| <Gap Size> | A Double representing the tolerance. The gaps in the contours that a smaller than this tolerance will be closed. |
| <Height Field> | A String representing the name of the height field in the input contour dataset. |

Scripting syntax

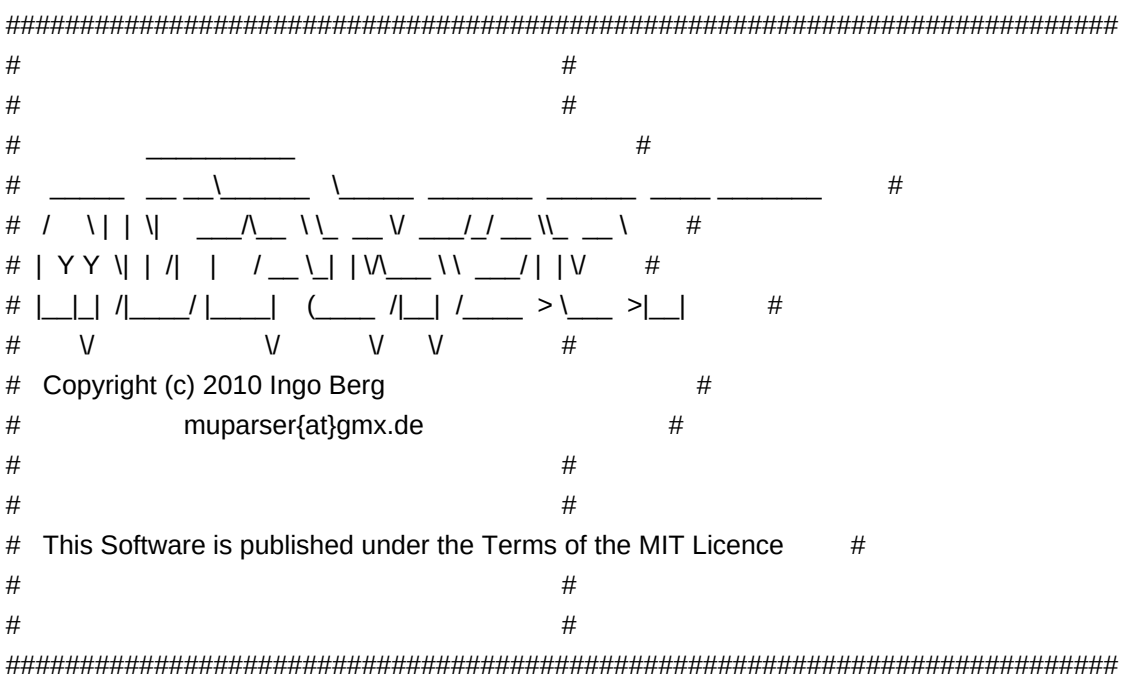
ETS_GPCleanContourGaps (Input Dataset,  Out Feature Class, Gap Size, Height Field)

See the explanations above:
<> - required parameter
{} - optional parameter

**.NET implementation**

(Go to TOP)

CleanContourGaps (inFeatureClass As IFeatureClass, sOutFeature As String, gapSize As Double, heightField As String) As IFeatureClass



Open Table of Contents

# TRIANGULATED IRREGULAR NETWORK

The TIN model represents a surface as a set of contiguous, non-overlapping triangles. Within each triangle the surface is represented by a plane. The triangles are made from a set of points called mass points.

Mass points can occur at any location, the more carefully selected, the more accurate the model of the surface. Well-placed mass points occur where there is a major change in the shape of the surface, for example, at the peak of a mountain, the floor of a valley, or at the edge (top and bottom) of cliffs.

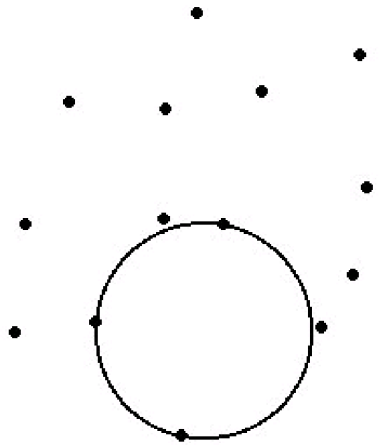
The TIN model is attractive because of its simplicity and economy and is a significant alternative to the regular raster of the GRID model.

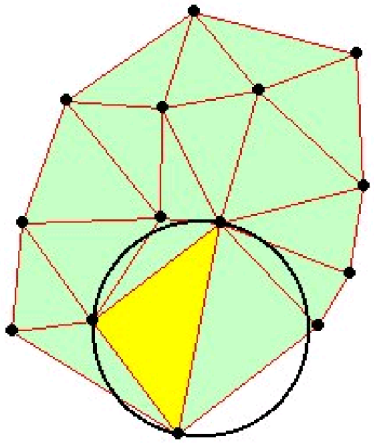
### Quick comparison:

| | TIN | GRID |
|---|---|---|
| Advantages | • ability to describe the surface at different level of resolution<br>• efficiency in storing data | • easy to store and manipulate<br>• easy integration with raster databases<br>• smoother, more natural appearance of derived terrain features |
| Disadvantages | • in many cases require visual inspection and manual control of the network | • inability to use various grid sizes to reflect areas of different complexity of relief. |

### The Delaunay Triangulation

Delaunay triangulation is a proximal method that satisfies the requirement that a circle drawn through the three nodes of a triangle will contain no other node





Delaunay triangulation has several advantages over other triangulation methods:

- The triangles are as equi-angular as possible, thus reducing potential numerical precision problems created by long skinny triangles
- Ensures that any point on the surface is as close as possible to a node
- The triangulation is independent of the order the points are processed

### TINs from contours

Contours are a common source of digital elevation data. In general all the vertices of the contour lines are used as mass points for triangulation. In many cases this will cause the presence of flat triangles in the surface.

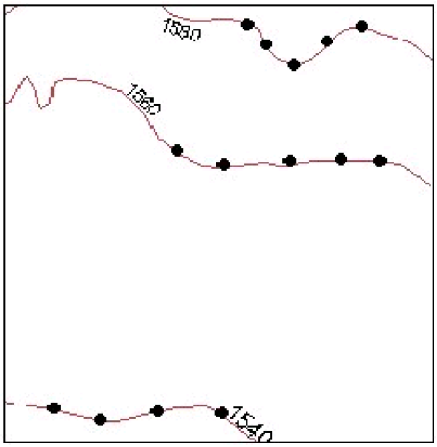
Flat triangles are created whenever a triangle is formed from three nodes with the same elevation value

Flat triangles are frequently generated along contours when the sample points occur along the contour at a distance that is less than the distance between contours. When these "excess" vertices are not removed , the Delaunay triangulation discovers that the closest sample points are those along the same contour, causing the generation of flat triangles.

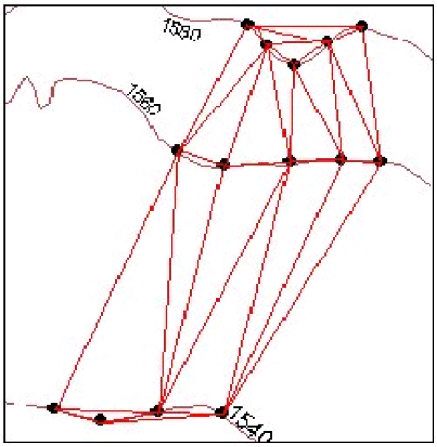
The flat triangles have a slope of 0 and do not have defined aspect. They might cause problems when the surface is used for modeling.

### Example:

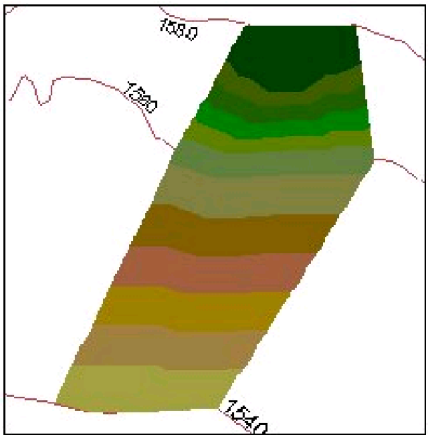
| The contours | The triangulation - We can see several flat triangles here |
|---|---|

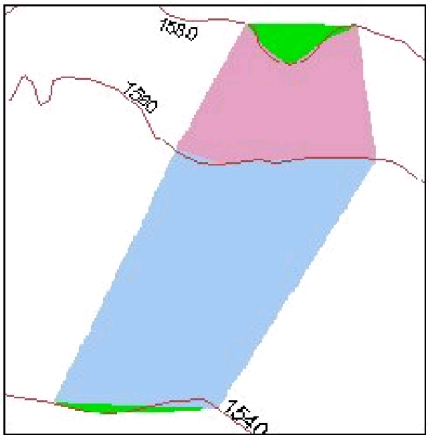



The elevation




The slope- The green areas indicate Slope = 0 (flat triangles)




How can we avoid the flat triangles?

- By adding more mass points
- Generalizing the contours
- By adding break lines

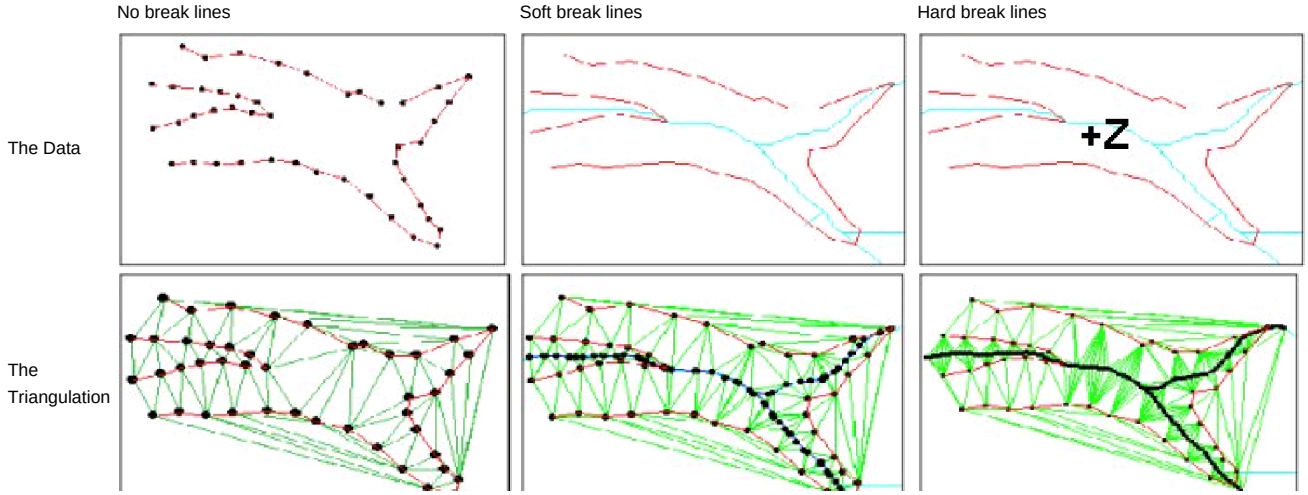
### Break lines

Linear features which define and control surface behavior in terms of smoothness and continuity are called break lines.

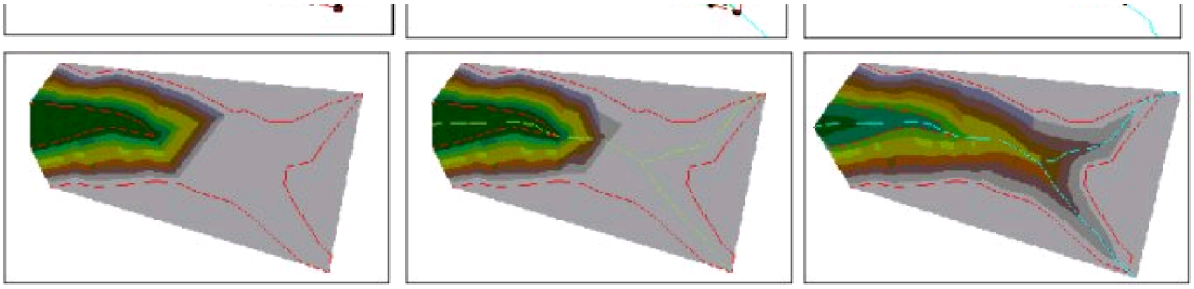
### Types break lines:

- Soft break lines are used to ensure that linear features and polygon edges are maintained in the tin surface model by enforcing the break line as tin edges. However, they do not define interruptions in surface smoothness – break lines with no Z value
- Hard break lines define interruptions in surface smoothness – break lines with Z value

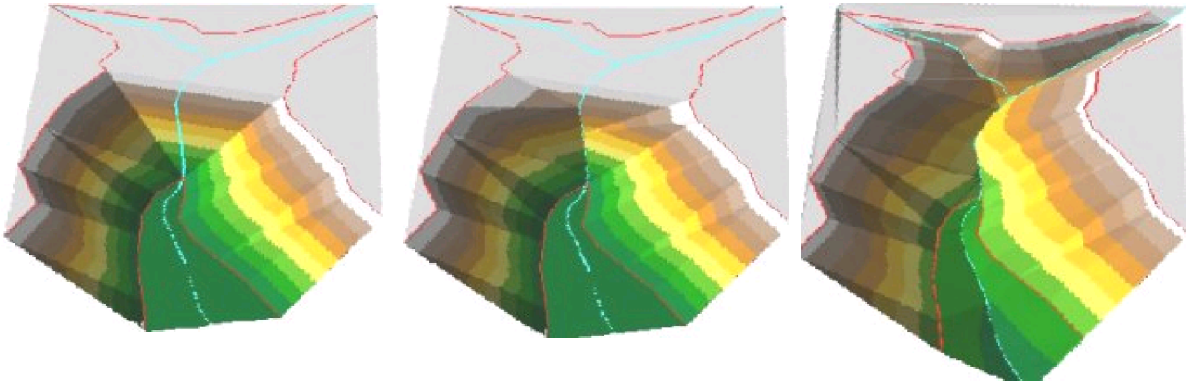
### Example:



The Surface



3D View



### Storing TINs

There are basically two ways of storing triangulated networks:

- Triangle by triangle
- Points and their neighbors

The first method is better for storing attributes (slope, aspect ..) for each triangle, but uses more storage space. The second one is better for generating contours and uses less storage space, but slope, aspect , etc must be calculated and stored separately.