DrawTools 3.2.0

For MapInfo Pro

January 2026

Content

Installation 2
Combine Features 3
Node Features 3
Set Node Coordinate 4
Update ZM Values 5
Update ZM Values using Nodes 5
Transform Features and Copies of Features 6
Create Offset Copies 6
Create Parallels 8
Offset in XYZ 10
Insert Features 11
Extending Polyline 11
Arrow 12
Isosceles Trapezoid 13
Point Buffer 14
Create Region Features 15
Create Cells Sectors 15
Extract and Create Features 16
Extract Nodes 16
Create Angle Node Points 17
Functions 17
DTSetCoordsysUsingTable 17
DTSetCoordsysUsingMap 17
DTLineDirection 18
DTLineBearing 18
DTMath2NorthAngle 19
DTNorth2MathAngle 19
DTRemoveDuplicateNodes 19
DTCreateTrapez 19
DTCreateTriangle 20
DTCartesianCreateArrow 20
DTCreateCircularSector 21
DTCreateAnnulusSector 21
DTCartesianOffsetXYZ 22
DTSphericalOffsetXYZ 22
DTObjectSetM 22
DTObjectSetZ 23
DTObjectScaleM 23
DTObjectScaleZ 23

Installation

DrawTools is downloaded and installed from the MapInfo Marketplace.

This distribution is automated so that you only click a button to get the add-in installed and loaded into MapInfo Pro.

If any updates to the tool gets published, you will see the small ***Notification*** symbol in the lower right corner of the MapInfo Pro window turn red. Double-click on the symbol to open the ***Notification*** window and from here access the updates from the MapInfo Marketplace where you easily can install the updates.

Please note that DrawTools requires MapInfo Pro v2019.3 or newer to work.

Some of the features such as the tools for M and Z-values require MapInfo Pro v2023 to work.

Read more about DrawTools on the MapInfo Community.

Combine Features

From the *Combine* dropdown in the *Edit* group on the *Spatial* tab, you can select *Combine Selected into New* to combine 2 or more selected features and save the new feature into the editable layer.

This feature will only be enabled when you have selected at least two objects and have an editable layer in the active map window.

Note that the selected records will not be erased and that you can select from any layer in the map, including the editable layer. Only the spatial object will get inserted into the editable layer. No attributes will be included.

Node Features

From the *Nodes* dropdown in the *Edit* group on the *Spatial* tab, you can find 6 features added by DrawTools.

Move Nodes: Allows you to enable/disable move duplicate nodes in same layer. Normally, this setting is only available via Map *preferences*.

Add Node: Allows you to add a node to a selected polyline at the closest end to where you clicked.

Add Start Node: Allows you to add a node to a selected polyline at the start of the polyline.

Add End Node: Allows you to add a node to a selected polyline at the end of the polyline.

All these three tools work in a similar way. First, you need to select a polyline from the editable layer, then you click with the tool where you want the new start/end node to be. The tool will automatically add the new node to the polyline.

Set Node Coordinate: Allows you to change existing nodes of a polyline or polygon by entering coordinates (X, Y, Z, and M) for these. See more details below.

Remove Start Node: Allows you to delete the start node from the selected polyline.

Remove End Node: Allows you to delete the end node from the selected polyline.

Update ZM Values: This allows you to update all nodes on one or more selected objects from the editable layer. Supports only points, polylines, and polygons.

Update ZM Values using Nodes: Allows you to update M and/or Z values on objects in the editable layer using a table of node points. The nodes in the node points table controls which records will get updated. The table of node points can be created using *Extract Nodes*. Supports only points, polylines, and polygons.

Set Node Coordinate

To use this, you need to select a polyline or polygon from the editable layer.

Now select the ***Set Node Coordinate*** tool and click very close to the node you want to change. It's recommended to activate snap to find a matching node to where you click.

The tool will search for the nearest node in the selected object. If a node is found with the search distance, currently 1/10 of the zoom width of the map window, an error is shown. Try enabling Snap before using the tool again.

If a node is found, the ***Set Node Coordinate dialog*** will be shown.

At the top of the dialog, you can see what segment and what node you currently are editing. In the map the current node is highlighted with a special symbol.

Enter the new coordinate for the node.

If the table supports M and Z values, you can also change these through this dialog. If the object doesn't already have M/Z values, you need to check the M/Z checkbox to be able to assign these values.

If you want to edit more nodes, you can use the two arrows at the top right to move to the next or previous node. When you do, you can see that the changes to the polyline or polygon are rendered on the map with a new dotted line style. In this way, you can see the changes as you make them.

You can also jump between the sections if your object is made up of several sections.

You can also delete the current node by clicking the ***Delete Node*** button on the dialog.

You can also delete an entire section of the object, unless the object only has one section. Use the *Delete Section button* to delete the current section. A warning will be shown before deleting the section.

You can also use the *Zoom In* and *Zoom Out* button to control the zoom of the map window.

Hit *Save* to push your changes to the selected object.

Update ZM Values

Select the objects you want to update. Make sure you select the features from the editable layer.

Now click on *Update ZM Values* in the *Nodes* dropdown on the *Spatial* tab.

In the *Update ZM Values* dialog, you can now control which values you want to update and how you want to update these values.

First, check the *Update Z Values* and/or *Update M Values* option depending on which value you want to update. You can check one of them or both.

Now select if you want to update using a fixed value or using values in a column. If you check *Value*, make sure to enter the value to assign to the Z and/or M values on the nodes in the appropriate field. If you check *Column*, make sure you select the correct column holding the values to assign to the Z and/or M values on the nodes.

Finally, you can select if you want to set the Z and/or M values on the nodes to the specified value, if you want to add the specified value to the existing value on the nodes, or if you want to multiply the existing value on the nodes by the specified value.

Click *OK* to update the M and/or Z values on the nodes.

Update ZM Values using Nodes

For this to work, you will need an editable layer. There should also be a table with the postfix of "`_Nodes`" open. If your editable layer is called `Buildings`, the table `Buildings_Nodes` also needs to be open.

It is recommended to use *Extract Nodes* to create the Nodes table, but it's not required. You can create this table in other ways too.

Click on *Update ZM Value using Nodes* in the *Nodes* dropdown on the *Spatial* tab.

In the dialog *Update MZ Values using Nodes*, you must now specify how to update the nodes in the editable layer.

Select *Use M and Z Values from Point Objects* to use the M and Z values on the spatial point objects in the Nodes table.

Alternatively, you can select *Use M and Z Values from Column* to read the values from columns in the Nodes table. Note that the second method also allows you to select if you want to update the Z value, the M value or both. For each, you will have to pick the column to read the value from.

Click *OK* to update the nodes on the objects in the editable layer.

Note that it is the Nodes table that controls which records in the editable layer get updated.

The Nodes table must contain these columns:

- `OBJECT_ID`: Refers to the `RowID` in the editable layer.
- `SEGMENT_ID`: refers to the segment for the specific object.
- `NODE_ID`: Refers to the node on the specific segment for the specific object.

The value in these columns controls which object, which segment, and which node will get updated.

This feature can be used to update the Z values on your polylines and polygons with for example the elevation from a digital elevation model. First, you extract the nodes using the *Extract Nodes* feature of DrawTools. Now you can use *Point Inspection* in the *Raster Processing* gallery to update these node points with the elevation from your elevation model. Finally, you can use *Update ZM Values using Nodes* to assign the value from the elevation model back to the original polyline or polygon.

Transform Features and Copies of Features

From the *Transform* dropdown in the *Edit* group on the *Spatial* tab, you can find 4 features added by DrawTools.

Create Offset Copies: Allows you to create several copies of the selected object and offset these by an increasing distance. See below for more details.

Create Parallels: Allows to create several copies of the selected object and make these copies a parallel copy by increasing the distance. See below for more details.

Add to Region: Allows you to add area to an existing selected polygon. Make sure that the layer is editable. Draw the area/polygon that you want to add to the existing polygon. The drawn polygon will be merged/combined with the existing polygon.

Remove from Region: Allows you to add area to an existing selected polygon. Make sure that the layer is editable. Draw the area/polygon that you want to remove from the existing polygon. The drawn polygon will be used to erase from the existing polygon. This could be holes you want to cut or areas you want to remove.

Offset in XYZ: Allows you to offset selected object(s) from the editable layer in Y, X and Z. You can also create copies of the selected object(s) and for each copy offset it the given value in X, Y and Z. See below for more details.

Create Offset Copies

Select the (poly)line you want to offset several times.

Now select the *Create Offset Copies tool* and draw a line on the map illustrating the distance and direction you want to offset the copies you are creating.

Make sure you have an active map with an editable layer and a single selected (poly)line. Otherwise, this tool will be disabled.

Only the offset spatial objects will get inserted into the editable layer. No attributes will be included.

Specify the times you want to offset the object and hereby the number of copies created.

You can also change the distance you want to offset the parallel copies. The distance here is based on the line you drew on the map.

The direction is also measured based on the line you drew on the map. You can change it here if required.

In the image below, you can see the resulting copies of the selected polyline.

Create Parallels

Select the line or polyline you want to create multiple parallel copies of and activate the *Create Parallels feature* from the *Buffer dropdown* on the *Spatial tab*.

Draw a line that will be used as the distance you want to move the parallel copies.

Make sure you have a map active with an editable layer and one selected (poly)line. Otherwise, the tool will be disabled.

Only the parallel spatial objects will get inserted into the editable layer. No attributes will be included.

Enter the number of parallel copies you want to create.

You can also change the distance you want to offset the parallel copies. The distance here is based on the line you drew on the map.

Choose the Orientation of the parallel copies. They can be on the left side, the right side or on both sides of the selected (poly)line.

Below you can see the resulting parallel lines created from the selected polyline.

This feature is using MapCAD in the background so ensure you have MapCAD running when using it.

Offset in XYZ

First, select the objects(s) that you want to offset in X, Y and/or Z. Make sure you select the object(s) from the editable layer.

Now use the *Offset XYZ tool* to draw a line indicating the distance in X and Y that you want to move the objects. The dialog will afterward allow you to change these values if needed.

When you have drawn the line, the *Offset XYZ* dialog will appear.

The values in the ***X, m*** and ***Y, m*** fields have been calculated from the line you drew.

Modify these and the value in the *Z* field if necessary.

Now, check if the calculation method is correct. Use *Spherical* for Lat/Long based coordinate system, and *Cartesian* in other cases.

Finally, you can check the option *Create Copies of the Records* if you want to create copies of the selected record(s).

If you check this option, you can also specify how many times. Each time the objects are copied they will be offset by the given values in X, Y and Z. The first time it will use the values entered. The next time, the values will be multiplied by 2. The third time by 3, and so on.

Insert Features

In the *Insert* dropdown in the *Create* group on the *Spatial* tab, you can find several features added by DrawTools.

In the *Lines section*, you can find a single feature and in the *Regions section*, you can find four.

Extending Polyline: Allows you to draw a polyline that will look for segments to snap to at the end and beginning of the polyline after you have finished drawing it. See more below.

Add Gaps: Allows you to create small polygons where you click in the map. These polygons will only be created if there is a gap in the polygons in the editable layer where you clicked. The gab will be created as a new polygon in the editable layer.

Arrow: Allows you to draw an arrow in the editable layer based on the polygon you draw with the tool. See more below.

Isosceles Trapezoid: Allows you to draw an isosceles trapezoid in the editable layer using the line created with the tool. See more below.

Point Buffer: Allows you to draw buffers in the editable layer by dragging a line representing the size of the buffer. See more below.

Extending Polyline

Start by drawing the polyline as you normally would. Make sure the end(s) that you want extended are close to the segment you want it extended to. Make sure you don't overlap.

Now the tool will look for objects in the editable layer and see if any segments of these objects are within the tolerance of the tool. If they are, the beginning or the end of the drawn polyline will be extended to intersection as you can see in the image below.

The tolerance is currently hardcoded to 10 meters.

Arrow

Make sure you have an editable layer in your map, then select the ***Arrow*** tool from the ***Insert*** dropdown on the ***Spatial*** tab.

This tool has a polyline draw mode which means you can draw a line with multiple nodes. Double-click to end the arrow and the ***Arrow*** dialog will pop up.

Now you can specify the size of your arrow.

Enter the width of the buffer around the polyline you have drawn and specify the resolution of the buffer.

For the two ends of the polyline, you can control how they should appear:

- Rounded
- Flat
- Arrow

If you choose ***Arrow***, you also need to specify the with and length of the arrow. This is specified in percentage of the buffer width. 200 will make it twice the size of the buffer. These two values do not have to be the same.

The width must be larger than 100% and the length must be larger than 10%.

Here are 3 arrows with widths of 150, 200 and 250% for your reference. This might help you select the best width.

Isosceles Trapezoid

Make sure you have an editable layer in your map, then select the ***Isosceles Trapezoid*** tool from the ***Insert*** dropdown on the ***Spatial*** tab.

This tool has a line draw mode which means you can draw a simple line. Hold the left mouse button down, drag and release the mouse button and the ***Isosceles Trapezoid*** dialog will pop up.

The tool will use the start point of the line and the direction specified and populate these details into the dialog. The length of the line will also be populated into the dialog.

If you know the precise coordinates you can enter that.

You can also change the direction. Add 180 to or subtract 180 from the value to reverse the direction.

Specify the length if it is different from the length of the drawn line.

Specify the width of the trapezoid at the beginning and at the end. If you specify a width of 0 at one of the ends, you will end up with a triangle.

You can use either a ***Cartesian*** or a ***Spherical*** calculation method. This depends on your data. For lat/long data, use ***Spherical***. Otherwise use ***Cartesian***.

Finally, you can choose to ***Add Node at Center of parallel sides***. This makes it easier to find the center of the parallel sizes, for example if you want to add a center line to the trapezoid.

The result can look like this where the trapezoid shows an altitude zone (I think that's the term) at the end of a runway. For your information, I haven't used the actual size for such a zone, but I think it gives you the idea

Point Buffer

The Point Buffer tool lets you draw a simple line representing the center of the buffer, and the radius of the buffer.

When you have drawn your line, the Create Point Buffer dialog is shown allowing you to modify the center as well as the distance (buffer width) used.

Click on the OK button to create the Point Buffer. What you get is a buffer around the center point with a currently hardcoded resolution of 36 points.

Create Region Features

Create Cells Sectors

From the *Regions dropdown* in the *Create group* on the *Spatial tab*, you can access the *Create Cell Sectors control*. This will allow you to create either Circular Sectors or Annulus Sectors from a mappable table using either the existing points in the table or columns with X and Y coordinates.

Make sure to have at least one mappable and editable table open before accessing the tool.

When you click on the *Create Cell Sectors tool*, the *Create Cell Sectors dialog* will be opened for you to configure how to create the sectors.

Start by selecting the table to use. Notice that the dialog shows you the coordinate system of the table under the Table list. This coordinate system will be used when creating the objects.

Next, you will have to decide if you want to use the potentially existing objects as the *Center* of the sectors or you want to use coordinates from two columns. Do remember that the coordinates must be in the coordinate system of the table.

You can control the size of the sectors using the *Inner Radius* and *Outer Radius options*. If you leave Inner Radius empty or set it to 0 (zero), the result will be circular sectors. If you set it to a value, the result will be annulus sectors or doughnut sectors.

For the *Inner Radius* and the *Outer Radius,* you can either specify a fixed value, or you can let the tool read the values from a column in your table making the size vary from sector to sector.

For Directions, you can specify a fixed value or set it to read the direction from a column. This is the direction that the sector will point out from the center. The direction can either be specified as a bearing where North is 0 and the values grow clockwise, or the direction can be specified as a mathematical angle where East is 0 and the values grow counterclockwise.

The width can also, like most other values, be set as a fixed value or set to be read from a column making the width very across the sectors.

Finally, you can set the number of sectors in a full circle and the calculation method. The dialog tries to guess the calculation method based on the coordinate system of that table. *Cartesian* is for projected coordinate systems and *Spherical* is for all things Lat/long.

Also note that DrawTools has function to create these object types through a script, see `DTCreateCircularSector` and `DTCreateAnnulusSector`.

Extract and Create Features

Extract Nodes: Allows you extract all nodes from the selected polygons/polylines into a nodes table.

Create Angle Node Points: Calculate the minimum angle for all nodes and store this angle at a point matching the node, see more below.

Extract Nodes

Select the object for which you want to extract nodes. Now click the *Extract Nodes* control.

The nodes will now get extracted into a new table including a reference to the RowID of the object in the original table, the segment number and the node number. These values can be used if you later want to use the *Update ZM Values using Nodes* feature.

The nodes are stored in a temporary table in the Temp folder. Remember to save a copy of this table if you want to keep it. The table will not be maintained in your workspace.

If the nodes table already exists, you will be asked if you want to overwrite it or add to it. This allows you to extract nodes multiple times during a session.

Create Angle Node Points

Click the *Create Angle Node Points* control and then select the table to create these node points for.

You can also select the selection or a query.

The tool will run through all the records in the table and for each node after the first, it will create a node and store the minimum angle for that node.

The nodes and angles are stored in a temporary table in the Temp folder. Remember to save a copy of this table if you want to keep it. The table will not be maintained in your workspace.

The angle stored is the minimum of the two angles between the two segments going out from the node.

In the image on the right, you can see the minimum angles calculated for these four nodes. They are all around 90 degrees. For all four nodes, there is another angle on the outside of the grey polygon of 270 degrees. But that is of course larger than the inner angle of just 90 degrees and so not stored.

Functions

DrawTools also comes with several functions that can be used from the user interface in MapInfo Pro. These functions can be found in the *Functions* list in the *SQL Window,* and you can use them from the MapBasic window and from the Update dialog, too.

DTSetCoordsysUsingTable

It sets DrawTools internally to use the same projection as the given table.

```
DTSetCoordsysUsingTable(tableName) As Void
```

- *tableName As String*: The name of an open table whose coordsys you want DrawTools to use.

This is useful when using some of the spatial functions of DrawTools where you pass in coordinates.

```
Print DTSetCoordsysUsingTable("CellTowers")
```

DTSetCoordsysUsingMap

It sets DrawTools internally to use the same projection as the given map window.

```
Function DTSetCoordsysUsingMap(windowID) As Void
```

- `windowID As Integer`: The ID of the map whose coordsys you want WindowHelper to use.

This is useful when using some of the spatial functions of DrawTools where you pass in coordinates.

```
Print DTSetCoordsysUsingMap(FrontWindow())
```

DTLineDirection

Calculates the direction of a line or polyline, East is zero, counterclockwise. For polylines, the direction is calculated as the direction from the first to the last node.

```
Function DTLineDirection( oInput ) As Float
```

- `oInput As Object`: A spatial object in the form of a line or polyline.
- Returns a float value as the direction of object in 360 degrees where East is 0 and the degrees grow counterclockwise. Unsupported objects return -1,
- This function only works with projected coordinate systems

In the example below, we update an existing column (`DIRECTION`) with the direction of the (poly)lines of the table `Highways`.

```
Update Highways Set DIRECTION = DTLineDirection(OBJ)
```

I have mapped the result as a label expression on only longer lines to avoid the label clutter. Below you can see the values calculated for three lines that all seem to have a similar direction. One is however northbound (49°) where the other two (230° and 229°) are southbound.

DTLineBearing

Calculates the bearing of a line or polyline, North is zero, clockwise. For polylines, the direction is calculated as the direction from the first to the last node.

```
Function DTLineBearing( oInput ) As Float
```

- `oInput As Object`: A spatial object in the form of a line or polyline.
- Returns a float value as the direction of object in 360 degrees where North is 0 and the degrees grow clockwise. Unsupported objects return -1.
- This function only works with longitude/latitude coordinate systems

In the example below, we update an existing column (`AZIMUTH`) with the direction of the (poly)lines in the table `Highways`.

```
Update Highways Set AZIMUTH = DTLineBearing(OBJ)
```

DTMath2NorthAngle

Converts a Math angle to a North/Compass angle. North angle has North as 0 and goes clockwise. A mathematical angle has 0 to the east and goes counterclockwise. Bearings are often North-based angles and must be converted to mathematical angles for use in the functions in DrawTools.

```
Function DTNorth2MathAngle( fAngle ) As Float
```

- *fAngle As Float*: The north-based angle to be converted to a mathematical angle.
- Returns a float value in the form of the mathematical angle

In the example above we calculated the mathematical angle for some roads. If we need these values as north-based angles instead, we can convert these using the expression below

```
Update Highways Set DIRECTION = DTMath2NorthAngle(DTLineDirection(OBJ))
```

DTNorth2MathAngle

Converts a North angle to a mathematical angle. North angle has North as 0 and goes clockwise. A mathematical angle has 0 to the east and goes counterclockwise. Bearings are often North-based angles and must be converted to mathematical angles for use in the functions in DrawTools.

```
Function DTMath2NorthAngle( fAngle) As Float
```

- *fAngle As Float*: The mathematical angle to be converted to a north-based angle.
- Returns a float value in the form of the north-based angle

If your angles are stored as north-based angles, you will have to convert these to math angles to be able to use them in the functions in DrawTools. The expression below will do this conversion for you.

```
Print DTNorth2MathAngle(90)
```

DTRemoveDuplicateNodes

Removes duplicate nodes from input object using the given tolerance in meters.

```
Function DTRemoveDuplicateNodes(oObj, fTolerance) As Object
```

- *oObj As Object*: The object (polyline or polygon) to remove duplicate nodes in
- *fTolerance As Float*: The tolerance defining whether two nodes are "the same", distance is measured using Cartesian calculation
- Returns the object with duplicate nodes removed

The example below updates the table `ROAD_SEGMENTS` with objects where close (duplicate) nodes have been removed. The tolerance has been set to 2 meters.

```
Print DTSetCoordsysUsingTable(ROAD_SEGMENTS)

Update ROAD_SEGMENTS Set OBJ = DTRemoveDuplicateNodes(OBJ, 2.0)
```

DTCreateTrapez

Creates a trapeze using a starting point, direction, length, width at start and end, optional midpoint creation and calculation method (1=Cartesian, 2=Spherical)

```
Function DTCreateTrapez( oStartPoint, fDirection, fLength, fWidthAtStart, fWidthAtEnd,
bCreateMidpoint, nCalculationMethod ) As Object
```

- *oStartPoint As Object*: Start point using current session coordinate system
- *fDirection As Float*: Direction from the start point

- `fLength As Float`: Width at starting point, if zero geometry will be a triangle
- `fWidthAtStart As Float`: Width at start point, if zero geometry will be a triangle
- `fWidthAtEnd As Float`: Width at end point, if zero geometry will be a triangle
- `bCreateMidpoint As Logical`: Determines if the trapeze should have a node at the midpoint on the two parallel sides
- `nCalculationMethod As Integer`: Specifies if the calculations should use Cartesian (1) or Spherical (2) methods. For lat/long based coordinates, use Spherical (2), and for projected coordinates, use Cartesian (1).

DTCreateTriangle

Creates a triangle using a starting point, direction, length, width, optional midpoint creation and calculation method (1=Cartesian, 2=Spherical)

```
Function DTCreateTriangle( oStartPoint, fDirection, fLength, fWidthAtEnd, bCreateMidpoint,
nCalculationMethod ) As Object
```

- `oStartPoint As Object`: Start point using current session coordinate system
- `fDirection As Float`: Direction from the start point
- `fLength As Float`: Width at starting point, if zero geometry will be a triangle
- `fWidthAtEnd As Float`: Width at end point
- `bCreateMidpoint As Logical`: Determines if the triangle should have a node at the midpoint on the end side
- `nCalculationMethod As Integer`: Specifies if the calculations should use Cartesian (1) or Spherical (2) methods. For lat/long based coordinates, use Spherical (2), and for projected coordinates, use Cartesian (1).
- Returns a spatial object in the form of a triangle using the specified values

DTCartesianCreateArrow

Creates an arrow from a polyline giving size and end types.

```
Function DTCartesianCreateArrow(oInput, fNodeDensity, nBufferResolution, fArrowWidth,
nArrowHeadType, fArrowHeadLengthPct, fArrowHeadWidthPct, nArrowTailType, fArrowTailLengthPct,
fArrowTailWidthPct) As Object
```

- `oInput As Object`: Input polyline to draw arrow based on
- `fNodeDensity As Float`: Density for removing duplicate points on input line, 0 for not removing
- `nBufferResolution As Integer`: Buffer resolution for arrow line
- `fArrowWidth As Float`: Width of arrow, in meters
- `nArrowHeadType As Integer`: Arrowhead Type, 1 (Rounded), 2 (Flat), 3 (Arrow)
- `fArrowHeadLengthPct As Float`: Length of Arrowhead in % of arrow width, given as a decimal value where 1.0 is 100%, typical value is around 2.0, for flat and rounded set it to 1.0
- `fArrowHeadWidthPct As Float`: Width of Arrowhead in % of arrow width, given as a decimal value where 1.0 is 100%, typical value is around 2.0, for flat and rounded set it to 1.0
- `nArrowTailType As Integer`: Arrow Tail Type, 1 (Rounded), 2 (Flat), 3 (Arrow)
- `fArrowTailLengthPct As Float`: Length of Arrow Tail in % of arrow width, given as a decimal value where 1.0 is 100%, typical value is around 2.0, for flat and rounded set it to 1.0
- `fArrowTailWidthPct As Float`: Width of Arrow Tail in % of arrow width, given as a decimal value where 1.0 is 100%, typical value is around 2.0, for flat and rounded set it to 1.0
- Returns a spatial object in the form of an arrow using the specified values

DTCreateCircularSector

Create a circular sector using a point, direction, and sizes. If you are updating the spatial object of a table using this function, the parameters can either be fixed values or come from a column in the table.

The image here shows some circular sectors created with this function.

```
Function DTCreateCircularSector( oCenter, fDirection, fRadiusOuter, fAngleWidth, nResolution,
nCalculationMethod ) As Object
```

- *oCenter As Object*: The center or starting point of the circular sector in the form of a spatial point.
- *fDirection As Float*: The direction of the circular sector measured using a mathematical angle
- *fRadiusOuter As Float*: The outer radius of the circular sector, in meters
- *fAngleWidth As Float*: The width of the circular sector measured in angles
- *nResolution As Integer*: The resolution for the buffer used. Specifies the number of points to describe a full circle
- *nCalculationMethod As Integer*: Specifies if the calculations should use Cartesian (1) or Spherical (2) methods. For lat/long based coordinates, use Spherical (2), and for projected coordinates, use Cartesian (1).
- Returns a spatial object in the form of a circular sector using the specified values

In the example below, I am creating circular sector from a table with existing points.

```
Set CoordSys Table CellTowerPoints
Update CellTowerPoints Set OBJ = DTCreateCircularSector(OBJ, DTNorth2MathAngle(BEARING), 900,
SPREAD_ANGLE, 36, 2)
```

You can also use lat/long values from columns as input instead of the OBJ column.

```
Set CoordSys Table CellTowerPoints
Update CellTowerPoints Set OBJ = DTCreateCircularSector(CreatePoint(LONG, LAT),
DTNorth2MathAngle(BEARING), 900, SPREAD_ANGLE, 36, 2)
```

Also notice the use of the function `DTNorth2MathAngle()` to convert a bearing to a mathematical angle which MapInfo Pro uses.

DTCreateAnnulusSector

Creates an annulus sector or doughnut sector using a point, a direction, and sizes. The annulus sector is basically a circular sector where the center has been removed. If you are updating the spatial object of a table using this function, the parameters can either be fixed values or come from a column in the table.

The image here shows some circular sectors created with this function. Notice how the center has been removed compared to the circular sectors above.

```
Function DTCreateAnnulusSector( oCenter, fDirection, fRadiusInner, fRadiusOuter, fAngleWidth,
nResolution, nCalculationMethod ) As Object
```

- `oCenter As Object`: The center or starting point of the annulus in the form of a spatial point.
- `fDirection As Float`: The direction of the circular sector measured using a mathematical angle
- `fRadiusInner As Float`: The inner radius of the annulus, in meters
- `fRadiusOuter As Float`: The outer radius of the annulus, in meters
- `fAngleWidth As Float`: The width of the annulus sector measured in angles
- `nResolution As Integer`: The resolution for the buffer used. Specifies the number of points to describe a full circle
- `nCalculationMethod As Integer`: Specifies if the calculations should use Cartesian (1) or Spherical (2) methods. For lat/long based coordinates, use Spherical (2), and for projected coordinates, use Cartesian (1).
- Returns a spatial object in the form of an annulus sector using the specified values

```
Set CoordSys Table CellTowerPoints
Update CellTowerPoints Set OBJ = DTCreateAnnulusSector(OBJ, DTNorth2MathAngle(BEARING), 50,
900, SPREAD_ANGLE, 36, 2)
```

Also notice the use of the function `DTNorth2MathAngle()` to convert a bearing to a mathematical angle which MapInfo Pro uses.

DTCartesianOffsetXYZ

Cartesian Offsets an existing object by XYZ values using defined distance units.

```
Function DTCartesianOffsetXYZ(oInput, fOffsetX, fOffsetY, fOffsetZ, sUnits) As Object
```

- `oInput As Object`: Object to be offset.
- `fOffsetX As Float`: Value to offset the Spatial Object in X
- `fOffsetY As Float`: Value to offset the Spatial Object in Y
- `fOffsetZ As Float`: Value to offset the Spatial Object in Z
- `sUnits As String`: Distance Unit for the offset values, for example `"m"`.
- Returns the Spatial Object offset using the specified values in XYZ

```
Print DTSetCoordsysUsingTable(BUILDINGS)

Update ROAD_SEGMENTS Set OBJ = DTCartesianOffsetXYZ (OBJ, 0, 0, HEIGHT, "m")
```

DTSphericalOffsetXYZ

Spherical Offsets an existing object by XYZ values using defined distance units.

```
Function DTSphericalOffsetXYZ(oInput, fOffsetX, fOffsetY, fOffsetZ, sUnits) As Object
```

- `oInput As Object`: Object to be offset.
- `fOffsetX As Float`: Value to offset the Spatial Object in X
- `fOffsetY As Float`: Value to offset the Spatial Object in Y
- `fOffsetZ As Float`: Value to offset the Spatial Object in Z
- `sUnits As String`: Distance Unit for the offset values, for example `"m"`.
- Returns the Spatial Object offset using the specified values in XYZ

DTObjectSetM

Sets the M value on an existing object to the given value.

```
Function DTObjectSetM(oInput, fValueM) As Object
```

- `oInput As Object`: Spatial Object to change the M value on
- `fValueM As Float`: New M value to set on the Input Object
- Returns the Input Spatial Object with a changed M value

DTObjectSetZ

Sets the Z value on an existing object to the given value.

```
Function DTObjectSetZ(oInput, fValueZ) As Object
```

- `oInput As Object`: Spatial Object to change the Z value on
- `fValueM As Float`: New Z value to set on the Input Object
- Returns the Input Spatial Object with a changed Z value

DTObjectScaleM

Multiplies the M value on an existing object by the given value.

```
Function DTObjectScaleM(oInput, fScaleMBy) As Object
```

- `oInput As Object`: Spatial Object to change the M value on
- `fScaleMBy As Float`: Value to multiply the existing M value of the Input Object by
- Returns the Input Spatial Object with a changed M value

DTObjectScaleZ

Multiplies the Z value on an existing object by the given value.

```
Function DTObjectScaleZ(oInput, fScaleMBy) As Object
```

- `oInput As Object`: Spatial Object to change the Z value on
- `fScaleMBy As Float`: Value to multiply the existing Z value of the Input Object by
- Returns the Input Spatial Object with a changed Z value