

## TMS FNC Chart DEVELOPERS GUIDE

## Index

Availability .....	3
Online references .....	3
Description .....	3
Opacity Support.....	3
Organization .....	4
Getting Started .....	7
Design Time .....	7
Runtime .....	8
Properties & Events .....	8
Virtual vs Collection based mode .....	19
Persistence .....	22
Adding and removing series .....	23
Accessing series .....	23
Adding and removing points .....	24
Annotations .....	25
Labels.....	26
X-axis and y-axis values .....	27
Autorange .....	31
Mathematical vs Statistical .....	33
Multi-Point Series.....	33
Pie.....	35
Spider.....	37
Legend .....	40
Markers .....	42
Stacked series.....	43
3D .....	47
Interaction .....	48

## Availability

---

Supported frameworks and platforms

- VCL Win32/Win64
- FMX Win32/Win64, MacOS-X, iOS, Android
- LCL Win32/Win64, Mac OS-X, iOS, Android, numerous Linux variants including Raspbian

Supported IDE's

- Delphi XE7 and C++ Builder XE7 or newer releases
- Lazarus 1.4.4 with FPC 2.6.4 or newer releases.

Important Notice: TMS FNC Chart requires TMS FNC Core (separately available at the [My Products page](#))

## Online references

---

TMS software website:

<http://www.tmssoftware.com>

TMS FNC Chart page:

<http://www.tmssoftware.com/site/tmsfncchart.asp>

## Description

---

The TMS FNC Chart (further referred to as “Chart”) is a fully cross-platform component designed to display different kinds of data such as financial and marketing data, monthly business sales, graphical and math data and much more as a chart. The Chart supports types such as bar, area, line, marker and variants such as stacked bar, stacked area, stacked percentage area, stacked percentage bar XY-line, XY-scatter and digital line. ...

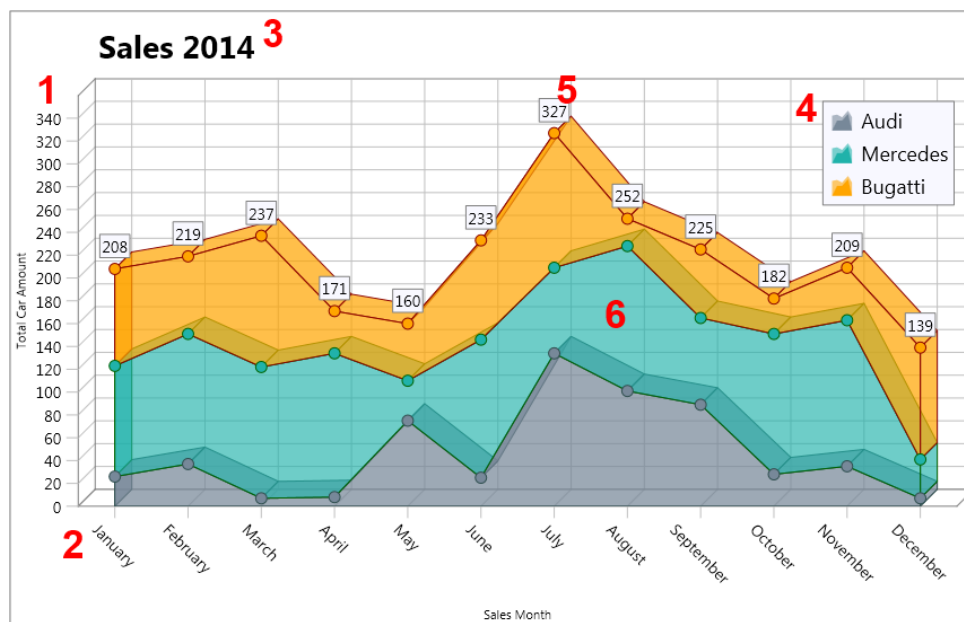
## Opacity Support

---

The TMS FNC Chart exposes a set of Fill and Stroke properties for drawing various elements such as the title, x-axis, y-axis, legend and many more. The fill and stroke properties are shared between FMX, VCL and LCL but only support opacity drawing on FMX.

## Organization

The Chart consists of multiple configurable visual elements that are numbered and explained in the screenshot below.



### 1: Y-axis

The y-axis displays a range of series values from a predefined range or an automatic calculated minimum and maximum range. The y-axis can be set at the left, center and/or right side of the Chart. Different y-axis values can be shown for different series. The y-axis has the capability to show major & minor units with a different font and has multiple events for further customization.

### 2: X-axis

The x-axis displays a range of series values from a predefined range or an automatic calculated minimum and maximum range. The x-axis can be set at the top, center and/or bottom side of the Chart. Different x-axis values can be shown for different series. The x-axis has the capability to show major & minor units with a different font and has multiple events for further customization.

### 3: Title

The title optionally displays a text with customizable position, font and font color.

### 4: Legend


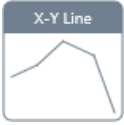



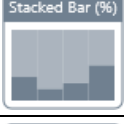






The legend optionally displays the text with a small glyph representing the chart type for each series added to the Chart.





### 5: Labels/Annotations

Each series has support for labels and multiple annotations for each added series point and can be customized in terms of appearance and formatting.

### 6: Series

The series of the Chart can be changed to one of the types listed below. Multiple series can be combined to create stacked series and show the summed total.

Name	Type	Description
ctLine		Series are shown as a line from value to value.
ctXYLine		Series are shown as a line from value to value, with a custom X value.
ctDigitalLine		Series are shown as a digital line from value to value.
ctBar		Series is shown as bars with height representing the value.
ctStackedBar		Multiple series joined in stacked bar which shows the summed value of all grouped series of type ctStackedBar.
ctStackedPercentageBar		Same as stacked bar but values per series are represented by percentage.
ctArea		Series are shown as a filled area.
ctStackedArea		Multiple series joined in stacked area which shows the summed value of all grouped series of type ctStackedArea.
ctStackedPercentageArea		Same as stacked bar but values per series are represented by percentage.
ctMarker		Series are shown as shape/image per value.
ctXYMarker		Series are shown as shape/image per value with a custom x-value.
ctPie		Series are shown as a pie shape with individual colourable slices.

<b>ctSizedPie</b>		Series are shown as a pie with slices with a fixed angle and a radius based on the added values.
<b>ctVariableRadiusPie</b>		Series are shown as a pie with slices with a variable angle and variable radius based on the added values.
<b>ctSpider</b>		Series are shown as a spider chart with multiple spokes based on the added values.
<b>ctBand</b>		Series are shown as a band chart with an upper (YValue) and lower value (YValueSecond)

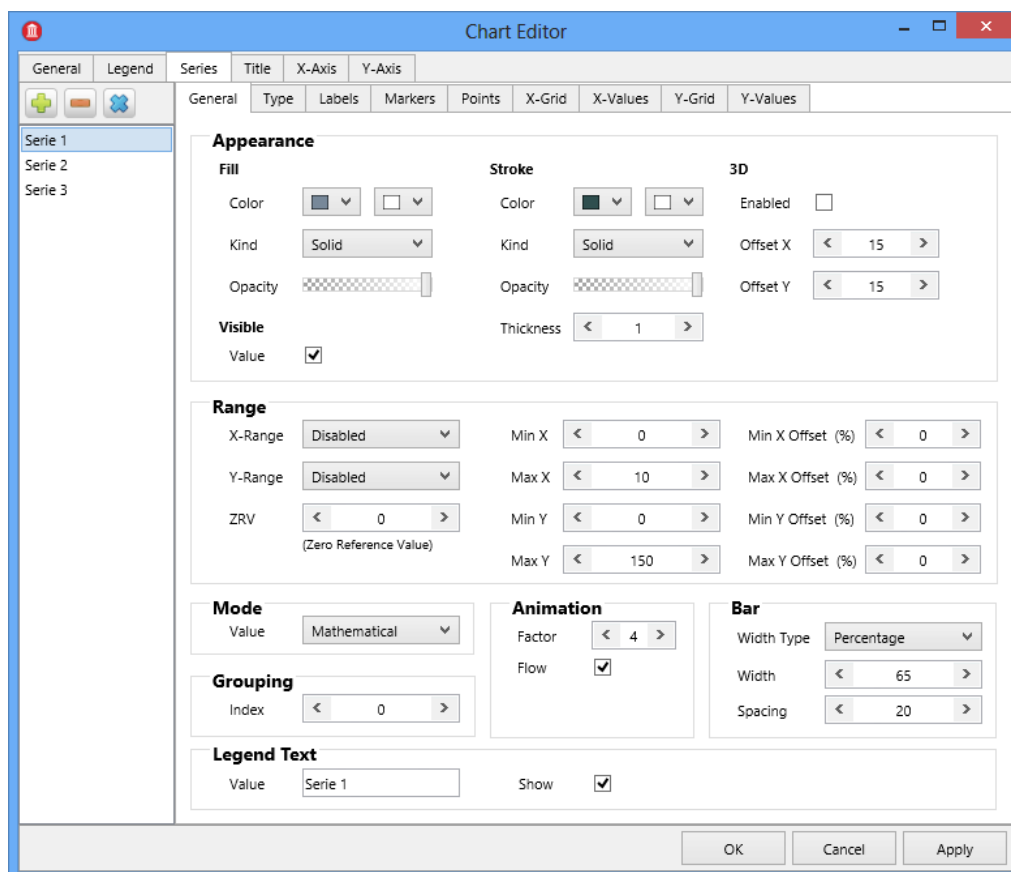
## Getting Started

The Chart has a visual and a non-visual component. When creating a new project, the tool palette will allow you to select and use these 2 components. The TTMSFNCChart component is the visual representation of your data and the TTMSFNCChartEditorDialog can be used to manipulate the appearance and the data at runtime.

## Design Time

When dropping an instance of the TTMSFNCChart on the form you will notice that the component already has added 3 series with a set of points. This allows you to quickly set up a test environment for your application.

Selecting the component allows you to manipulate the appearance of the various elements, the collection of series, points and annotations through the object inspector. To allow you to change these properties in a more convenient way, we have created a designtime editor available when double-clicking on the TTMSFNCChart instance, or when editing the Series property.



The editor presents each chart and series properties and its subproperties in different tabs and groups the properties in logical categories.

## Runtime

The introduction of this chapter mentioned a non-visual component TTMSFNCChartEditorDialog. This component is a wrapper around the design time editor and can be used to edit the Chart at runtime.

The data that is initialized by default is also available at runtime when starting the application. This data can be cleared by removing the points from the series, or by removing all series from the Chart. The code that can be used to accomplish this is demonstrated below.

```
//Clear points, keep series
TMSFNCChart1.BeginUpdate;
for I := 0 to TMSFNCChart1.Series.Count - 1 do
    TMSFNCChart1.Series[I].Points.Clear;
TMSFNCChart1.EndUpdate;
```

```
//Clear series and points
TMSFNCChart1.BeginUpdate;
TMSFNCChart1.Clear;
TMSFNCChart1.EndUpdate;
```

When at some point you wish to add some sample series and points for testing purposes, you can use the code below.

```
//Initialize a sample
TMSFNCChart1.BeginUpdate;
TMSFNCChart1.InitSample;
TMSFNCChart1.EndUpdate;
```

## **Properties & Events**

---

### **Properties**

ClickMargin	The margin which is used to detect a point when clicking on the Chart.
Fill*	The fill of the background of the Chart.
Interaction	Enables or disables interaction on the Chart.
InteractionOptions	Interaction options for the Chart.
InteractionOptions → Panning	Enables or disables panning of the Chart in x / y direction. Please note that the AutoXRange and/or AutoYRange of the series need to be set to arDisabled to allow panning.
InteractionOptions → ScaleMode	Enables or disables scaling of the Chart in x / y direction. Please note that the AutoXRange and/or AutoYrange of the series need to be set to arDisabled to allow scaling.
Legend	The Legend of the Chart.
Legend → Fill*	The fill of the legend.
Legend → Font*	The font of the legend.
Legend → Left	The left offset of the legend.
Legend → Position	The position of the legend relative from the series rectangle.
Legend → Stroke	The stroke of the legend.
Legend → Top	The top offset of the legend.



Legend → Visible	Shows / hides the legend.
Series	The collection of series.
Series → AnimationFactor	The factor of the animation when the series are animated.
Series → AnimationFlow	Enables an alternative animation mode when animating the series.
Series → AutoXRange	<p>The range of the series x-values.</p> <ul style="list-style-type: none"> <li>- arDisabled Does not automatically calculate the range of the series x-values and applies the Series → MinX and Series → MaxX properties.</li> <li>- arEnabled Automatically calculates the range of the series x-values.</li> <li>- arEnabledZeroBased Automatically calculates the range of the series x-values based on a minimum of 0 (default reference value).</li> <li>- arCommon Automatically calculates the common range of all series x-values with the Series → AutoXRange property set to arCommon or arCommonZeroBased.</li> <li>- arCommonZeroBased Automatically calculates the common range of all series x-values with a common minimum of 0 (default reference value) with the Series → AutoXRange property set to arCommon or arCommonZeroBased.</li> </ul>
Series → Bar → Spacing	The spacing between a group of bars.
Series → Bar → Width	The width of a bar chart
Series → Bar → WidthType	The type of the width applied, which can be actual pixels or a percentage of the available X-Scale.
Series → Fill	The fill of the series.
Series → GroupIndex	The group index of the series when using bar charts and stacked bar charts or stacked area charts.
Series → Labels	The labels of the series that can be made visible and display for each point.
Series → Labels → Fill*	The fill of the labels.
Series → Labels → Font	The font of the text of the labels. The text is based on the Series → Points → YValue property and formatted with the Series → Labels → Format property.
Series → Labels → Format	The format string of the labels.
Series → Labels → FormatType	The type of format of the labels. The type can be set to normal (Format Delphi function), float

	(FormatFloat Delphi function) or DateTime (FormatDateTime function).
Series → Labels → Mode	The mode of the labels. The mode is set to normal by default which displays the actual Series → Points → YValue property value with the chosen formatting. When set to stacked, the same formatting is applied but on the summed total of each series.
Series → Labels → OffsetX	An additional x offset starting from the default location of the label.
Series → Labels → OffsetY	An additional y offset starting from the default location of the label.
Series → Labels → Stroke*	The stroke of the label.
Series → Labels → Visible	Shows or hides the labels.
Series → Legend	The Legend of the series.
Series → Legend → Fill*	The fill of the legend of the series.
Series → Legend → Font*	The font of the legend of the series.
Series → Legend → Left	The left offset of the legend of the series.
Series → Legend → Position	The position of the legend relative from the series rectangle.
Series → Legend → Stroke	The stroke of the legend of the series.
Series → Legend → Top	The top offset of the legend of the series.
Series → Legend → Visible	Shows / hides the legend of the series.
Series → LegendText	The legend text of the series that is displayed in the legend.
Series → Markers	The markers of the series. Markers are predefined shapes or images with the ability to apply customization through one of the various events.
Series → Markers → Bitmap	The default bitmap of a marker when the Series → Markers → Shape property is set to contain a bitmap.
Series → Markers → Fill*	The fill of the marker.
Series → Markers → Height	The height of the marker.
Series → Markers → Shape	The shape of the marker.
Series → Markers → Stroke*	The stroke of the marker.
Series → Markers → Visible	Shows / hides the marker.
Series → Markers → Width	The width of the marker.
Series → MaxX	The maximum x value when the Series → AutoXRange property is set to arDisabled.
Series → MaxXOffsetPercentage	An additional offset in percentage that is applied to the series maximum x value.
Series → MaxY	The maximum y value when the Series → AutoYRange property is set to arDisabled.
Series → MaxYOffsetPercentage	An additional offset in percentage that is applied to the series maximum y value.
Series → MinX	The minimum x value when the Series → AutoXRange property is set to arDisabled.
Series → MinXOffsetPercentage	An additional offset in percentage that is applied to the series minimum x value.
Series → MinY	The minimum y value when the Series → AutoYRange property is set to arDisabled.
Series → MinYOffsetPercentage	An additional offset in percentage that is applied to the series minimum y value.
Series → Mode	Changes the mode of the series between

	mathematically and statistical.
Series → MultiPoint → DecreaseFillColor	The fill color applied to a multi-point chart when the open or Q3 value is lower than the close or Q1 value.
Series → MultiPoint → DecreaseStrokeColor	The stroke color applied to a multi-point chart when the open or Q3 value is lower than the close or Q1 value.
Series → MultiPoint → IncreaseFillColor	The fill color applied to a multi-point chart when the open or Q3 value is higher than the close or Q1 value.
Series → MultiPoint → IncreaseStrokeColor	The stroke color applied to a multi-point chart when the open or Q3 value is higher than the close or Q1 value.
Series → MultiPoint → Width	The width of the rectangular area of a multi-point series.
Series → MultiPoint → WidthType	The type of width applied to the rectangular area of a multi-point series, which can be actual pixels or a percentage of the available X-Scale.
Series → Offset3DX	The x offset used to draw the series in a 3D-like view in a 2D coordinate space.
Series → Offset3DY	The y offset used to draw the series in a 3D-like view in a 2D coordinate space.
Series → Pie	The settings of a series, when the series contain one or multiple pie charts.
Series → Pie → AutoSize	Enables auto-sizing of the pie. If false, the size property is used to determine the size of the pie.
Series → Pie → InnerSize	The inner size of the pie.
Series → Pie → Position	The position of the pie relative to the series rectangle, and determined by the stacked property and the number of series.
Series → Pie → Size	The size of the pie, if the AutoSize property is false.
Series → Pie → Stacked	When one or multiple pie(s) have the stacked property to True, the series rectangle is not divided by the number of series. Instead all series are drawn with the same rectangle. The Margins still apply.
Series → Pie → StartAngle	The start angle of the pie.
Series → Pie → SweepAngle	The sweep angle of the pie.
Series → Pie → Margins	The pie margins, which are also used to determine the pie rectangle, based on the stacked property and the number of series.
Series → Points	The points
Series → Points → Annotations	The annotations of a point for a series. Each series can have multiple points, and each point can have multiple annotations.
Series → Points → Annotations → Arrow	The type of arrow that is drawn when displaying an annotation.
Series → Points → Annotations → ArrowColor	The color of the arrow(s).
Series → Points → Annotations → ArrowOpacity (FireMonkey only)	The opacity of the arrow(s).
Series → Points → Annotations → ArrowSize	The size of the arrow(s).
Series → Points → Annotations → AutoSize	Enables auto-sizing of the annotation based on the text.

Series → Points → Annotations → BalloonArrowSize	The size of the balloon arrow when the shape is set to asBalloon.
Series → Points → Annotations → BalloonDirection	The direction to which the balloon should point.
Series → Points → Annotations → CornerRadius	The radius of the corners of the shape when the shape is set to asBalloon or asRectangle.
Series → Points → Annotations → Fill*	The fill of the annotation.
Series → Points → Annotations → Font*	The font of the text of the annotation.
Series → Points → Annotations → Height	The height of the annotation when AutoSize is false.
Series → Points → Annotations → LineColor	The color of the line of the annotation.
Series → Points → Annotations → LineOpacity (FireMonkey only)	The opacity of the line of the annotation.
Series → Points → Annotations → LineThickness	The thickness of the line of the annotation.
Series → Points → Annotations → OffsetX	An additional x offset applied to the annotation starting from the default position.
Series → Points → Annotations → OffsetY	An additional y offset applied to the annotation starting from the default position.
Series → Points → Annotations → Shape	The shape of the annotation. An annotation can be change to a rectangle, a balloon or an ellipse shape.
Series → Points → Annotations → Stroke*	The stroke of the annotation.
Series → Points → Annotations → Text	The text of the annotation.
Series → Points → Annotations → TextHorizontalAlignment	The horizontal alignment of the text of the annotation.
Series → Points → Annotations → TextVerticalAlignment	The vertical alignment of the text of the annotation.
Series → Points → Annotations → Visible	Shows / hides the annotation.
Series → Points → Annotations → Width	The width of the annotation with AutoSize is false.
Series → Points → Annotations → WordWrap	Enables / disables wordwrapping on the annotation.
Series → Points → Color	The color of the point, used to color bars or pie slices of a series.
Series → Points → Explode	The explode value of a slice of a pie type chart.
Series → Points → LegendText	The legend text of a point.
Series → Points → XValue	The x value of the point. When adding a new point, this value is incremented by 1.
Series → Points → XValueText	The text of the x value used in combination with an unmodified XValue.
Series → Points → YValue	The y value of the point.
Series → Points → YValueSecond	The second / lower y value of the point (ctBand).
Series → Points → YValueVariable	The second or variable y value of the point.
Series → ShowInLegend	Optionally displays the LegendText of the series in the legend.
Series → Stroke	The stroke of the series, applied on a line type Chart and the border of area and bar charts.
Series → Visible	Shows / hides the series.
Series → XGrid	The grid based on the x values of the series which automatically follows the XValues MajorUnit and MinorUnit properties.
Series → XGrid → Extended	Extends the grid to the series rectangle with or without SeriesMargins or limits the grid to the

	same rectangle with the SeriesMargins.
Series → XGrid → MajorUnitStroke*	The major unit stroke settings for the x-grid.
Series → XGrid → MinorUnitStroke*	The minor unit stroke settings for the x-grid.
Series → XGrid → Visible	Shows / hides the x-grid.
Series → XValues	The values drawn on the x-axis.
Series → XValues → Angle	The angle that is used to rotate the values drawn on the x-axis.
Series → XValues → AutoUnits	Applies automatic unit calculation based on the available width of the x-axis.
Series → XValues → MajorUnit	The major unit value used to draw the x-axis values in divisions.
Series → XValues → MajorUnitFont*	The font of the major unit values.
Series → XValues → MajorUnitFormat	The formatting of the major unit values.
Series → XValues → MajorUnitFormatType	The format type of the major unit values. The formatting is identical to the series labels formatting.
Series → XValues → MajorUnitSpacing	The spacing of the major unit values between the text and the tickmark.
Series → XValues → MajorUnitTickMarkColor	The color of the tickmark of the minor unit.
Series → XValues → MajorUnitTickMarkSize	The size of the tickmark of the minor unit.
Series → XValues → MinorUnit	The minor unit value used to draw the x-axis values in divisions.
Series → XValues → MinorUnitFont*	The font of the minor unit values.
Series → XValues → MinorUnitFormat	The formatting of the minor unit values.
Series → XValues → MinorUnitFormatType	The format type of the minor unit values. The formatting is identical to the series labels formatting.
Series → XValues → MinorUnitSpacing	The spacing of the minor unit values between the text and the tickmark.
Series → XValues → MinorUnitTickMarkColor	The color of the tickmark of the minor unit.
Series → XValues → MinorUnitTickMarkSize	The size of the tickmark of the minor unit.
Series → XValues → Positions	The position of the x-axis values of a series. Each series can position its values optionally top, center and/or bottom.
Series → XValues → Title	The title of the x-axis values of a series.
Series → YGrid	The grid based on the y values of the series which automatically follows the YValues MajorUnit and MinorUnit properties.
Series → YGrid → Extended	Extends the grid to the series rectangle with or without SeriesMargins or limits the grid to the same rectangle with the SeriesMargins.
Series → YGrid → MajorUnitStroke*	The major unit stroke settings for the y-grid.
Series → YGrid → MinorUnitStroke*	The minor unit stroke settings for the y-grid.
Series → YGrid → SpiderKind	Sets the kind of grid that is displayed when the ctSpider chart type is chosen.
Series → YGrid → SpiderLegend	Shows / hides the legend values on the outside of the grid when the ctSpider chart type is chosen.
Series → YGrid → SpiderVisible	Shows / hides the grid when the ctSpider chart type is chosen.
Series → YGrid → Visible	Shows / hides the y-grid.
Series → YValues	The values drawn on the y-axis.
Series → YValues → AutoUnits	Applies automatic unit calculation based on the available width of the y-axis.
Series → YValues → MajorUnit	The major unit value used to draw the y-axis

	values in divisions.
Series → YValues → MajorUnitFont*	The font of the major unit values.
Series → YValues → MajorUnitFormat	The formatting of the major unit values.
Series → YValues → MajorUnitFormatType	The format type of the major unit values. The formatting is identical to the series labels formatting.
Series → YValues → MajorUnitSpacing	The spacing of the major unit values between the text and the tickmark.
Series → YValues → MajorUnitTickMarkColor	The color of the tickmark of the minor unit.
Series → YValues → MajorUnitTickMarkSize	The size of the tickmark of the minor unit.
Series → YValues → MinorUnit	The minor unit value used to draw the y-axis values in divisions.
Series → YValues → MinorUnitFont*	The font of the minor unit values.
Series → YValues → MinorUnitFormat	The formatting of the minor unit values.
Series → YValues → MinorUnitFormatType	The format type of the minor unit values. The formatting is identical to the series labels formatting.
Series → YValues → MinorUnitSpacing	The spacing of the minor unit values between the text and the tickmark.
Series → YValues → MinorUnitTickMarkColor	The color of the tickmark of the minor unit.
Series → YValues → MinorUnitTickMarkSize	The size of the tickmark of the minor unit.
Series → YValues → Positions	The position of the y-axis values of a series. Each series can position its values optionally top, center and/or bottom.
Series → YValues → SpiderValues	Shows / hides the values on the grid when the ctSpider chart type is chosen.
Series → YValues → Title	The title of the y-axis values of a series.
Series → ZeroReferenceValue	The value used as a reference for drawing the bar and area Chart types.
SeriesMargins	Additional margins applied to the series rectangle after calculation based on the x-axis, y-axis and title.
Stroke*	The stroke of the Chart.
Title	The title of the Chart.
Title → Border	Optionally enables / disables a border on the title.
Title → Fill	The fill of the title.
Title → Font*	The font of the title.
Title → Height	The height of the title.
Title → Line	Draws a single line on the title rectangle based on its position in the Chart.
Title → Positions	The title positions, which can be top, bottom or both.
Title → Stroke	The stroke of the title.
Title → Text	The text of the title.
Title → TextHorizontalAlignment	The horizontal alignment of the text of the title.
Title → TextMargins	The text margins of the title.
Title → TextVerticalAlignment	The vertical alignment of the text of the title.
Title → Visible	Shows / hides the title.
XAxis	The x-axis of the Chart.
XAxis → Autosize	Enables autosizing of the x-axis. Autosizing automatically calculates the spacing for all x-axis enabled series.
XAxis → Border	Optionally displays the border of the x-axis.

XAxis → DisplayAtReferenceValue	Optionally displays the centered X-Axis at a specific reference value based on the ReferenceValueSeriesIndex property
XAxis → Fill*	The fill of the x-axis.
XAxis → Height	The height of the x-axis.
XAxis → Line	Draws a single line on the x-axis rectangle based on its position in the Chart.
XAxis → Positions	The x-axis positions, which can be top, center, bottom or combinations of those three values.
XAxis → ReferenceValue	The value where the centered X-Axis is placed based on the ReferenceValueSeriesIndex and ReferenceValue properties.
XAxis → ReferenceValueSeriesIndex	The index of the series that is being referenced to calculate the position of the centered X-Axis based on the ReferenceValue property
XAxis → Stroke*	The stroke of the x-axis.
XAxis → Visible	Shows / hides the x-axis.
YAxis	The y-axis of the Chart.
YAxis → Autosize	Enables autosizing of the y-axis. Autosizing automatically calculates the spacing for all y-axis enabled series.
YAxis → Border	Optionally displays the border of the y-axis.
YAxis → DisplayAtReferenceValue	Optionally displays the centered Y-Axis at a specific reference value based on the ReferenceValueSeriesIndex property
YAxis → Fill*	The fill of the y-axis.
YAxis → Line	The height of the y-axis.
YAxis → Positions	Draws a single line on the y-axis rectangle based on its position in the Chart.
YAxis → ReferenceValue	The value where the centered Y-Axis is placed based on the ReferenceValueSeriesIndex and ReferenceValue properties.
YAxis → ReferenceValueSeriesIndex	The index of the series that is being referenced to calculate the position of the centered Y-Axis based on the ReferenceValue property
YAxis → Stroke*	The y-axis positions, which can be top, center, bottom or combinations of those three values.
YAxis → Visible	The stroke of the y-axis.
YAxis → Width	Shows / hides the y-axis.

## Events

OnAfterDrawBackground	Event called after the background of the Chart is drawn.
OnAfterDrawChart	Event called after the Chart is drawn.
OnAfterDrawLegend	Event called after the legend is drawn.
OnAfterDrawLegendIcon	Event called after the icon of an entry in the legend is drawn.
OnAfterDrawSerieAnnotation	Event called after an annotation of a point on a series is drawn.
OnAfterDrawSerieBar	Event called after a bar of point on a series is drawn.
OnAfterDrawSerieLabel	Event called after a label of a point on a series is drawn.



OnAfterDrawSerieLegend	Event called after the legend of a series is drawn.
OnAfterDrawSerieLegendIcon	Event called after the icon of an entry in the legend of a series is drawn.
OnAfterDrawSerieLegendIconVirtual	
OnAfterDrawSerieLine	Event called after a line between 2 points on a series is drawn.
OnAfterDrawSerieMarker	Event called after a marker of a point on a series is drawn.
OnAfterDrawSeries	Event called after all series have been drawn.
OnAfterDrawSerieSlice	Event called after the slice of a series is drawn.
OnAfterDrawSerieXGridLine	Event called after an x grid line of a series has been drawn.
OnAfterDrawSerieXValue	Event called after an x value of a series has been drawn.
OnAfterDrawSerieYGridLine	Event called after a y grid line of a series has been drawn.
OnAfterDrawSerieYValue	Event called after a y value of a series has been drawn.
OnAfterDrawTitle	Event called after the title has been drawn.
OnAfterDrawXAxis	Event called after the x-axis has been drawn.
OnAfterDrawXValuesTitle	Event called after the x-values title of a series has been drawn.
OnAfterDrawYAxis	Event called after the y-axis has been drawn.
OnAfterDrawYValuesTitle	Event called after the y-values title of a series has been drawn.
OnAnimateSerieFinished	Event called when the series animation is finished.
OnAnimateSerieStarted	Event called when the series animation is started.
OnBeforeDrawBackground	Event called before the background of the Chart is drawn.
OnBeforeDrawChart	Event called before the Chart is drawn.
OnBeforeDrawLegend	Event called before the legend is drawn.
OnBeforeDrawLegendIcon	Event called before the icon of an entry in the legend is drawn.
OnBeforeDrawSerieAnnotation	Event called before an annotation of a point on a series is drawn.
OnBeforeDrawSerieBar	Event called before a bar of point on a series is drawn.
OnBeforeDrawSerieLabel	Event called before a label of a point on a series is drawn.
OnBeforeDrawSerieLegend	Event called before the legend of a series is drawn.
OnBeforeDrawSerieLegendIcon	Event called before the icon of an entry in the legend of a series is drawn.
OnBeforeDrawSerieLegendIconVirtual	
OnBeforeDrawSerieLine	Event called before a line between 2 points on a series is drawn.
OnBeforeDrawSerieMarker	Event called before a marker of a point on a series is drawn.
OnBeforeDrawSeries	Event called before all series have been drawn.
OnBeforeDrawSerieSlice	Event called before the slice of a series is drawn.
OnBeforeDrawSerieXGridLine	Event called before an x grid line of a series has been drawn.
OnBeforeDrawSerieXValue	Event called before an x value of a series has



	been drawn.
OnBeforeDrawSerieYGridLine	Event called before a y grid line of a series has been drawn.
OnBeforeDrawSerieYValue	Event called before a y value of a series has been drawn.
OnBeforeDrawTitle	Event called before the title has been drawn.
OnBeforeDrawXAxis	Event called before the x-axis has been drawn.
OnBeforeDrawXValuesTitle	Event called before the x-values title of a series has been drawn.
OnBeforeDrawYAxis	Event called before the y-axis has been drawn.
OnBeforeDrawYValuesTitle	Event called before the y-values title of a series has been drawn.
OnCustomizeAnnotationFill	Event called to customize the fill for an annotation.
OnCustomizeAnnotationFont	Event called to customize the font for an annotation.
OnCustomizeAnnotationStroke	Event called to customize the stroke for an annotation.
OnDrawTitleText	Event called when the tile of the Chart is being drawn.
OnGetAnnotation	Event called to retrieve the data for a virtual annotation based on an index after looping through the number of annotations returns in the OnGetNumberOfAnnotations event.
OnGetNumberOfAnnotations	Event called when retrieving the number of annotations for a specific point.
OnGetNumberOfPoints	Event called when retrieving the number of points in virtual mode. When implemented, the points collection is cleared.
OnGetPoint	Retrieves the data for a virtual point based on an index after looping through the the number of points returned in the OnGetNumberOfPoints event.
OnGetSerieLabel	Event called when retrieving the value of a series label on a specific point.
OnGetSerieLabelVirtual	Event called when retrieving the value of a series label on a specific point when using virtual mode.
OnGetSerieLegendText	Event called to change the text that will be drawn in the series legend, for each value in a series point collection.
OnGetSerieLegendTextVirtual	Event called to change the text that will be drawn in the series legend, for each value in a series point collection when using virtual mode.
OnGetSerieSpiderLegendText	Event called to change the text that will be drawn in the spider grid legend, for each value in a series point collection.
OnGetSerieSpiderLegendTextVirtual	Event called to change the text that will be drawn in the spider grid legend, for each value in a series point collection when using virtual mode.
OnGetSerieXValue	Event called when retrieving the value of a series x value on the x-axis.
OnGetSerieYValue	Event called when retrieving the value of a series y value on the y-axis.
OnSerieBarClick	Event called when clicking on a bar.

OnSerieBarClickVirtual	Event called when clicking on a bar in virtual mode.
OnSeriePointClick	Event called when clicking on a point.
OnSeriePointClickVirtual	Event called when clicking on a point in virtual mode.
OnSerieSliceClick	Event called when clicking on a slice.
OnSerieSliceClickVirtual	Event called when clicking on a slice in virtual mode.

## Virtual vs Collection based mode

---

When dropping a new instance of the TTMSFNCChart on the form, the chart is initialized with three series, all containing a set of points added through the Points collection. This is called a collection-based mode which is also the default mode. When taking a look at the events, you will notice that some events have a virtual equivalent that is only called when implementing a virtual mode. The reason for having these events is to make a clear difference between virtual and collection based modes whenever a point is passed through as a parameter. All the other events can access the internal record data that holds a reference to the point collection item (Reference property), or the virtual point record data (VirtualReference property).

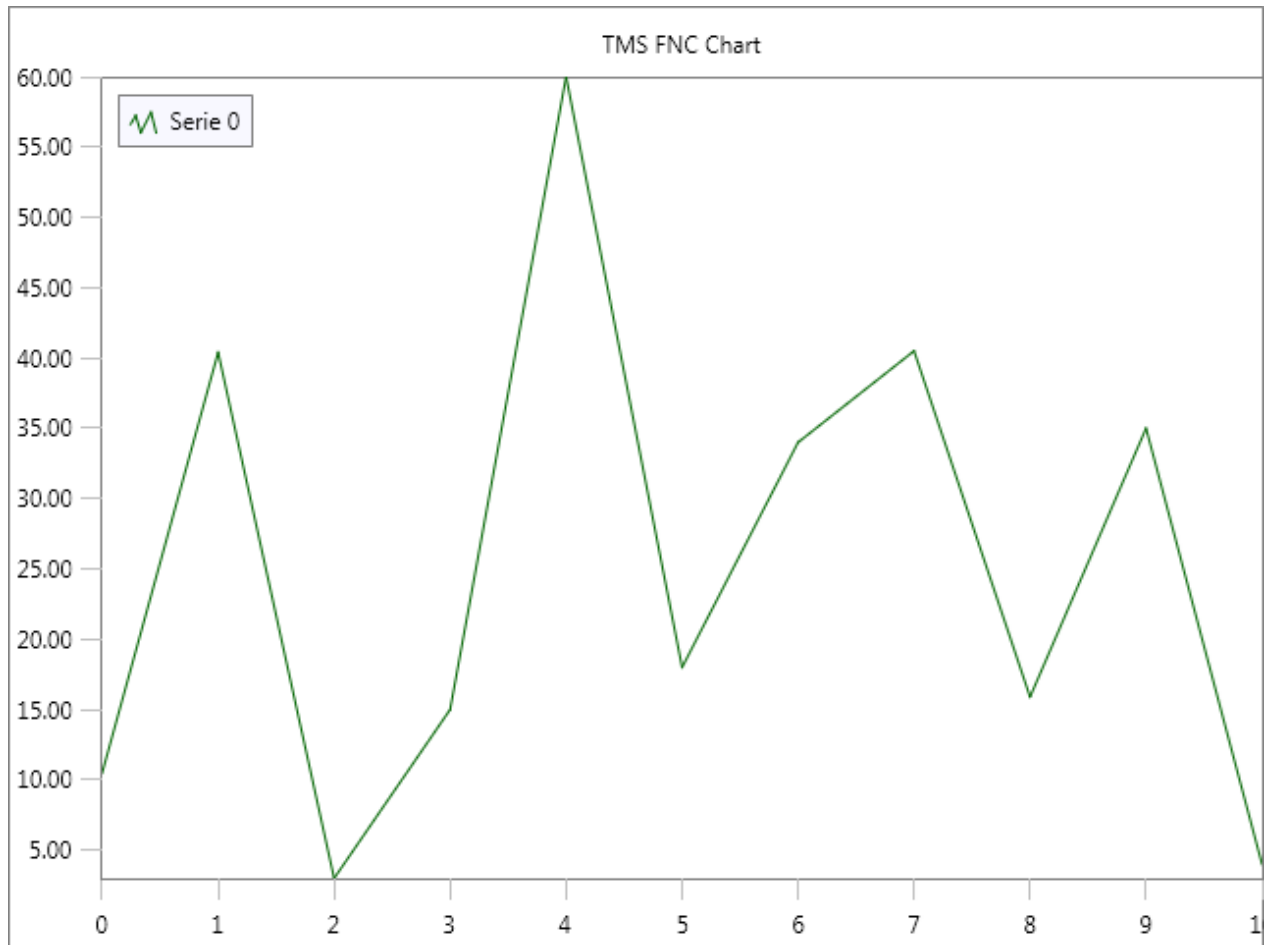
The virtual mode is enabled as soon as you implement the OnGetNumberOfPoints. Virtual mode is a global chart mode, so it is not possible to combine a collection-based and a virtual mode series. After implementing the OnGetNumberOfPoints, the OnGetPoint event is called to retrieve the data for a point. This is done through a record that can be directly accessed and manipulated. The advantage is that the event signature will not change when adding more properties in the future. There is no difference between virtual and collection-based mode in terms of series. The series are through the Series collection. Below is a sample that demonstrates this.

```
const
    PointArray: array[0..10] of Double = (10.5, 40.4, 3, 15, 60, 18, 34,
    40.5, 15.9, 35, 4);

procedure TForm1.FormCreate(Sender: TObject);
begin
    TMSFNCChart1.BeginUpdate;
    TMSFNCChart1.Series.Clear;
    TMSFNCChart1.Series.Add;
    TMSFNCChart1.EndUpdate;
end;

procedure TForm1.TMSFNCChart1GetNumberOfPoints(Sender: TObject;
    ASerie: TTMSFNCChartSerie; var ANumberOfPoints: Integer);
begin
    ANumberOfPoints := Length(PointArray);
end;

procedure TForm1.TMSFNCChart1GetPoint(Sender: TObject;
    ASerie: TTMSFNCChartSerie; AIndex: Integer;
    var APoint: TTMSFNCChartPointVirtual);
begin
    APoint.YValue := PointArray[AIIndex];
    APoint.XValue := AIndex;
end;
```



The virtual equivalent for annotations is available through the OnGetNumberOfAnnotations and OnGetAnnotation events as demonstrated in the sample below.

```

const
  PointArray: array[0..10] of Double = (10.5, 40.4, 3, 15, 60, 18, 34,
40.5, 15.9, 35, 4);

procedure TForm1.FormCreate(Sender: TObject);
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Series.Clear;
  TMSFNCChart1.Series.Add;
  TMSFNCChart1.EndUpdate;
end;

procedure TForm1.TMSFNCChart1GetAnnotation(Sender: TObject;
  ASerie: TTMSFNCChartSerie; APoint: TTMSFNCChartPointVirtual; AIndex:
Integer;
  var AAnnotation: TTMSFNCChartAnnotationVirtual);
begin
  AAnnotation.Text := 'Hello World !';
end;

procedure TForm1.TMSFNCChart1GetNumberOfAnnotations(Sender: TObject;
  ASerie: TTMSFNCChartSerie; APoint: TTMSFNCChartPointVirtual;

```

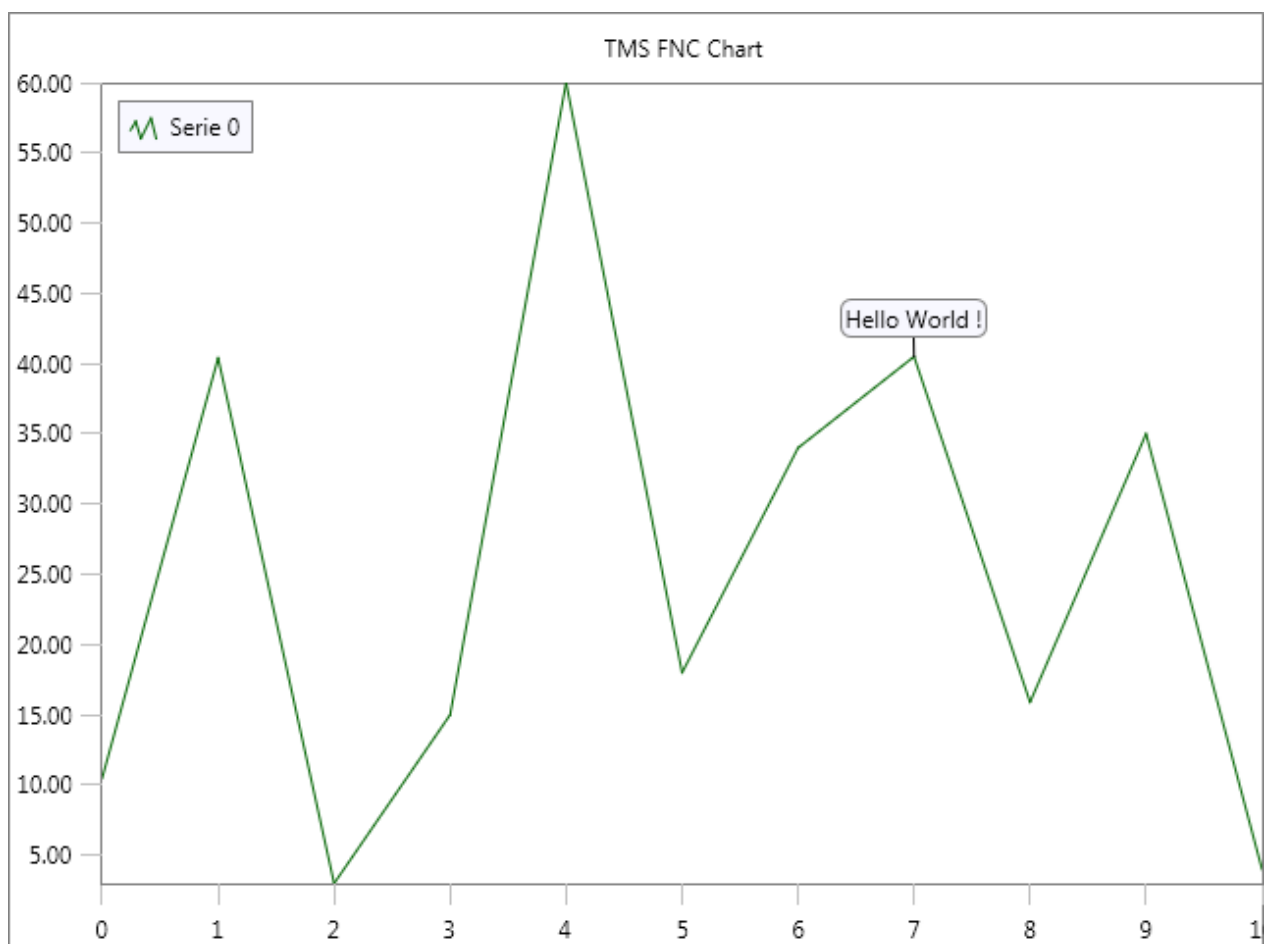
```

    var ANumberOfAnnotations: Integer);
begin
    if APoint.Index = 7 then
        ANumberOfAnnotations := 1;
end;

procedure TForm1.TMSFNCChart1GetNumberOfPoints(Sender: TObject;
    ASerie: TTMSFNCChartSerie; var ANumberOfPoints: Integer);
begin
    ANumberOfPoints := Length(PointArray);
end;

procedure TForm1.TMSFNCChart1GetPoint(Sender: TObject;
    ASerie: TTMSFNCChartSerie; AIndex: Integer;
    var APoint: TTMSFNCChartPointVirtual);
begin
    APoint.YValue := PointArray[AIndex];
    APoint.XValue := AIndex;
end;

```



## Persistence

---

The chart is capable of saving its published properties (settings), to a file or stream. The format that is being used is JSON. To save the chart settings, use the code below.

```
TMSFNCCChart1.SaveSettingsToFile();  
TMSFNCCChart1.SaveSettingsToStream();
```

To load an existing settings stream/file use the following code.

```
TMSFNCCChart1.LoadSettingsFromFile();  
TMSFNCCChart1.LoadSettingsFromStream();
```

The Chart additionally exposes events to control which properties need to be saved to the settings file. In some circumstances, it might be required to only save a specific set of properties. The OnCanLoadProperty and OnCanSaveProperty events are responsible for this. Below is a sample that excludes a property 'Extra' from the persistence list.

```
procedure TForm1.TMSFNCCChart1CanLoadProperty(Sender, AObject: TObject;  
  APropertyName: string; APropertyType: TTypeKind; var ACanLoad: Boolean);  
begin  
  ACanLoad := ACanLoad and not (APropertyName = 'Extra');  
end;
```

```
procedure TForm1.TMSFNCCChart1CanSaveProperty(Sender, AObject: TObject;  
  APropertyName: string; APropertyType: TTypeKind; var ACanSave: Boolean);  
begin  
  ACanSave := ACanSave and not (APropertyName = 'Extra');  
end;
```

Please note that the above AND operation is crucial to maintain the existing exclusion list. Returning a true for each property will additionally save its default published properties such as Align, Position and many more.

## Adding and removing series

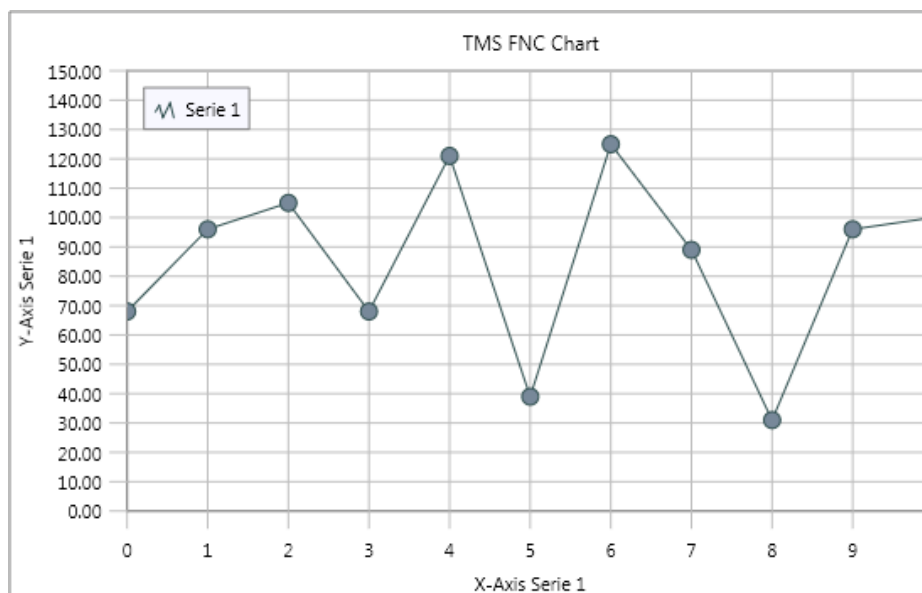
---

The Chart has a collection of series that can be accessed programmatically or through the editor. The code below shows you how to add a new series based on a default Chart. With the code, the Chart is cleared, and a new series is added.

```
TMSFNCChart1.BeginUpdate;
TMSFNCChart1.Clear;
TMSFNCChart1.Series.Add;
TMSFNCChart1.EndUpdate;
```

To delete a series, you will need to know the index of the series you wish to delete. By default, the Chart adds 3 series with random values. With the following code, the last 2 series in the collection are removed.

```
TMSFNCChart1.BeginUpdate;
TMSFNCChart1.Series.Delete(1);
TMSFNCChart1.Series.Delete(1);
TMSFNCChart1.EndUpdate;
```



When adding a new series, the series does not contain any points, so the Chart will not draw any lines, bars or other chosen Chart types. Adding and removing points is explained in the chapter Adding and removing points.

## Accessing series

---

When adding a new series, the series automatically adds an identifier, set with the LegendText property. This property is used in the legend, and in the editor. In code, you can access the series with the index in the collection, but more convenient, with a function called SerieByName. Below is a sample that demonstrates how this function can be used.

```
var
  I: Integer;
  s: TTMSFNCChartSerie;
```

```
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  TMSFNCChart1.Series.Add; //adds 'Serie 0' by default
  TMSFNCChart1.Series.Add; //adds 'Serie 1' by default

  s := TMSFNCChart1.SeriesByName['Serie 0'];
  for I := 0 to 7 do
    s.AddPoint(Random(100));

  s := TMSFNCChart1.SeriesByName['Serie 1'];
  for I := 0 to 7 do
    s.AddPoint(Random(100));

  TMSFNCChart1.EndUpdate;
end;

var
  I: Integer;
  s: TTMSFNCChartSerie;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  s := TMSFNCChart1.Series.Add;
  s.LegendText := 'Mercedes';
  TMSFNCChart1.EndUpdate;
end;

var
  s: TTMSFNCChartSerie;
begin
  s := TMSFNCChart1.SeriesByName['Mercedes'];
  for I := 0 to 7 do
    s.AddPoint(Random(100));
end;
```

## Adding and removing points

---

Adding or removing points is as easy as adding or removing series. Simply use the new series or retrieve an already existing series and use the Points collection property to add or remove new or existing points. By default, the Points collection already adds a random value to the YValue property. The XValue property is automatically incremented and has a direct relation to the number of points.

To add a new point, use the following code:

```
var
  s: TTMSFNCChartSerie;
begin
  TMSFNCChart1.BeginUpdate;
  s := TMSFNCChart1.Series.Add;
  s.Points.Add;
  TMSFNCChart1.EndUpdate;
end;
```



As explained, the point that is added to the series contains a random value. To change this value, define a variable that gives you access to the point properties like demonstrated in the code below.

```
var
  s: TTMSFNCChartSerie;
  pt: TTMSFNCChartPoint;
begin
  TMSFNCChart1.BeginUpdate;
  s := TMSFNCChart1.Series.Add;
  pt := s.Points.Add;
  pt.YValue := 123;
  TMSFNCChart1.EndUpdate;
end;
```

Each point has a value on the y-axis which is set with the YValue property, and a value on the x-axis. The value on the x-axis is set with the XValue property but is only used in XY type charts such as the ctXYLine or the ctXYMarker types.

An alternative to add a new point with a value is to use one of the AddPoint or AddXYPoint overloads that are publically accessible on serie level. The code below has an identical result as the previous code.

```
var
  s: TTMSFNCChartSerie;
begin
  TMSFNCChart1.BeginUpdate;
  s := TMSFNCChart1.Series.Add;
  s.AddPoint(Random(100));
  TMSFNCChart1.EndUpdate;
end;
```

To remove a point, simply use the same approach as removing a series.

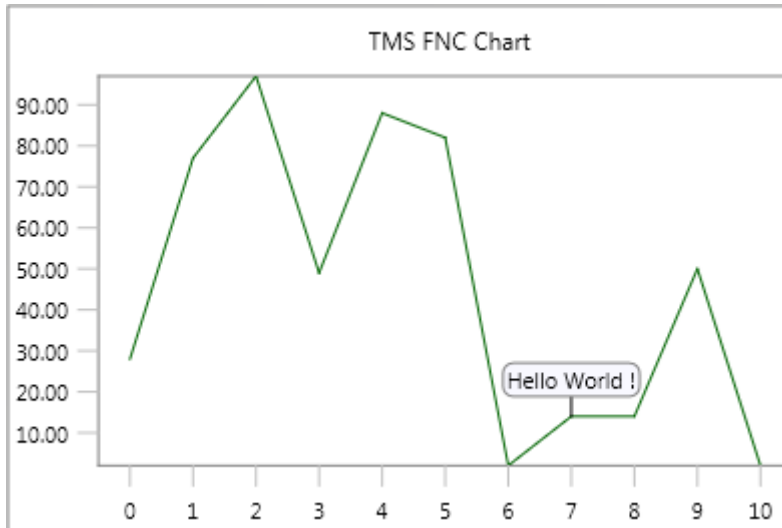
## Annotations

---

Annotations can be used to attach additional information to a specific point, shaped in a rectangle, ellipse or balloon, with many customization options. Annotations are added and deleted in the same way as series, but on point level. Below is a sample which adds an annotation to a specific point in a line Chart.

```
var
  s: TTMSFNCChartSerie;
  I: Integer;
  an: TTMSFNCChartAnnotation;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  s := TMSFNCChart1.Series.Add;
  s.Mode := smStatistical;
  for I := 0 to 10 do
  begin
    s.Points.Add;
    if I = 7 then
    begin
      an := s.Points[I].Annotations.Add;
      an.Text := 'Hello World !';
    end;
  end;
```

```
end;  
end;  
TMSFNCChart1.EndUpdate;
```

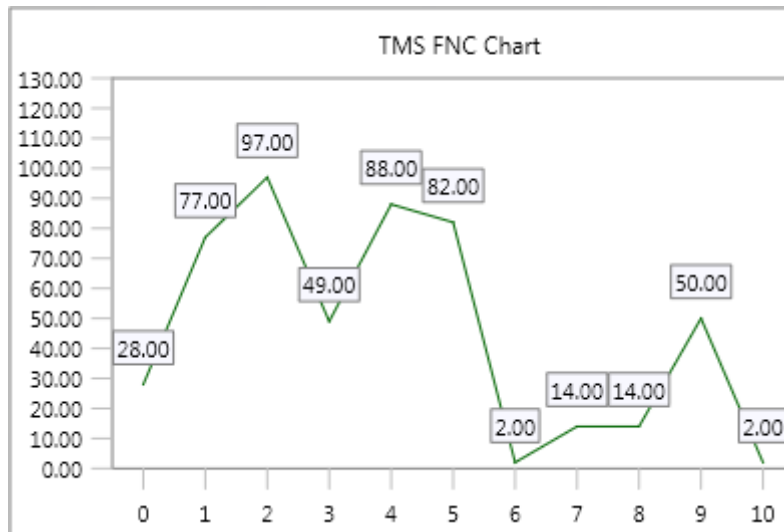


Annotations are auto-sized by default but can be configured to allow text alignment and wordwrapping.

## Labels

Each series has the ability to show labels, which display a formatted string based on the YValue of the point. Labels have the same appearance, and are added to each point. Through events, labels can optionally be hidden, but are less configurable compared to annotations.

```
var  
  s: TTMSFNCChartSerie;  
  I: Integer;  
begin  
  TMSFNCChart1.BeginUpdate;  
  TMSFNCChart1.Clear;  
  s := TMSFNCChart1.Series.Add;  
  s.Mode := smStatistical;  
  s.AutoYRange := arDisabled;  
  s.MinY := 0;  
  s.MaxY := 130;  
  s.Labels.Visible := True;  
  for I := 0 to 10 do  
    s.Points.Add;  
  TMSFNCChart1.EndUpdate;  
end;
```



The labels shown in the sample are already formatted with the Delphi “Format” function which can be optionally modified to format floating point values or datetime values. The type of formatting can be changed with the FormatType property.

## X-axis and y-axis values

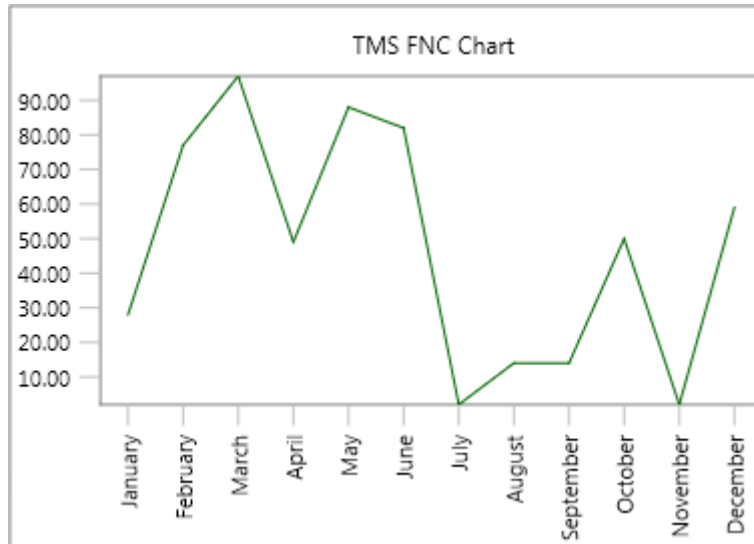
By default, the Chart enables the x-axis and the y-axis for the first series only, but each series has its own x-axis and y-axis range and can configure the position and formatting for each axis separately. The amount of values that are shown depend on a number of properties, the available width / height, the major and minor unit and the font size are the most important properties.

Each series can position its x-axis top, center, bottom or a combination of those three values and the same applies for the y-axis, but with a left, center and right position. Further customization can be done with one of the various events for custom drawing, formatting, positioning, etc...

The x-axis has an additional feature that is based on the collection of points inside a series. Each point has an XValueText property that is linked to the XValue of that point. When the XValueText is set, the series will automatically detect and display the text at that point. Below is a sample which adds the months of the year as values of the x-axis.

```
var
  s: TTMSFNCChartSerie;
  I: Integer;
  pt: TTMSFNCChartPoint;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  s := TMSFNCChart1.Series.Add;
  s.Mode := smStatistical;
  s.AutoXRange := arEnabled;
  s.XValues.Angle := -90;
  for I := 1 to 12 do
  begin
    pt := s.Points.Add;
    pt.XValueText := FormatSettings.LongMonthNames[I];
  end;
  TMSFNCChart1.EndUpdate;
```

**end;**

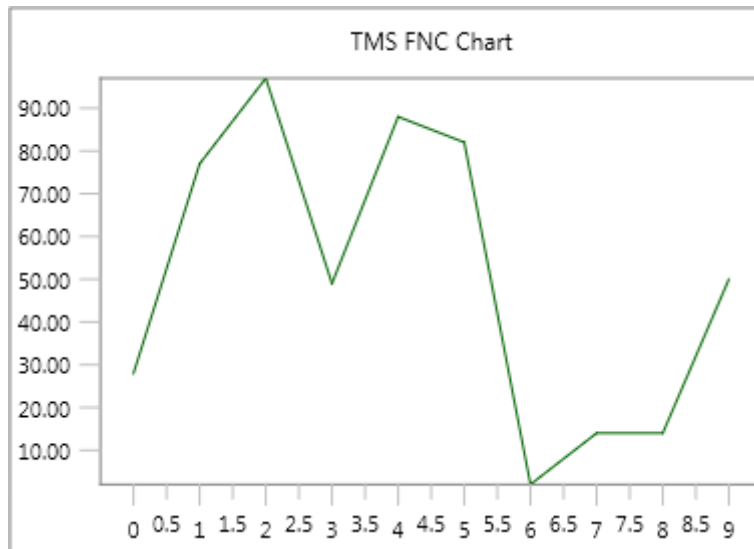


The values of the x-axis are now replaced with the months set via the XValueText property. In the sample, the values are rotated, because there isn't enough room to place all values horizontally. This is achieved with the Angle property of the series object.

The MajorUnit and MinorUnit properties that are available for the x-axis values at serie level can be used to change the appearance. By default the MajorUnit is 1 and the MinorUnit is 0. The x-axis values do not automatically calculate the units as the y-axis does with the AutoUnits property. This property is false by default on the x-axis. Without a value assigned to the XValueText property, the x-axis will draw the floating point values with a specific formatting, based on the MajorUnit and MinorUnit.

Below is a sample which sets the MinorUnit to 0.5 for the x-axis.

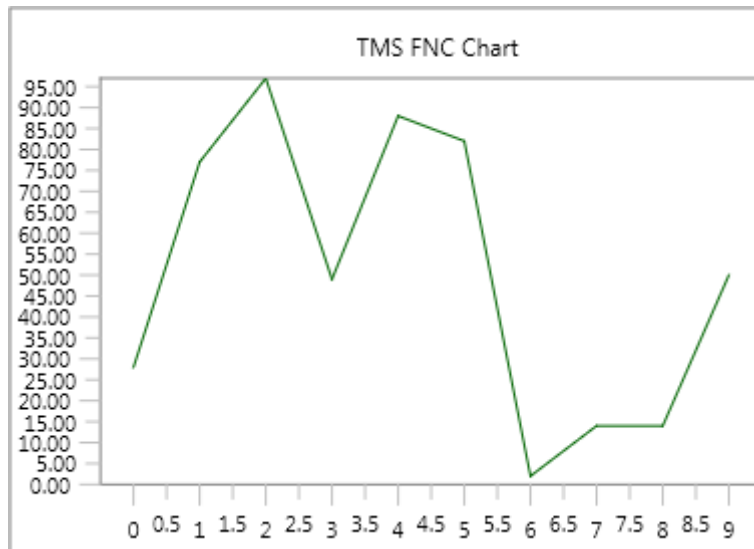
```
var
    s: TTMSFNCChartSerie;
    I: Integer;
begin
    TMSFNCChart1.BeginUpdate;
    TMSFNCChart1.Clear;
    s := TMSFNCChart1.Series.Add;
    s.Mode := smStatistical;
    s.AutoXRange := arEnabled;
    s.XValues.MinorUnit := 0.5;
    s.XValues.MinorUnitFormat := '%.1f';
    for I := 0 to 9 do
        s.Points.Add;
    TMSFNCChart1.EndUpdate;
end;
```



In this sample, we have also changed the MinorUnitFormat to display the fractional part of the MinorUnit set to 0.5.

The y-axis automatically calculates the best possible MajorUnit and MinorUnit by default. Changing the MajorUnit and MinorUnit on the y-axis will only be possible when the AutoUnits property is set to false.

```
var
  s: TTMSFNCChartSerie;
  I: Integer;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  s := TMSFNCChart1.Series.Add;
  s.Mode := smStatistical;
  s.AutoXRange := arEnabled;
  s.AutoYRange := arEnabledZeroBased;
  s.XValues.MinorUnit := 0.5;
  s.XValues.MinorUnitFormat := '%.1f';
  s.YValues.AutoUnits := False;
  s.YValues.MajorUnit := 10;
  s.YValues.MinorUnit := 5;
  for I := 0 to 9 do
    s.Points.Add;
  TMSFNCChart1.EndUpdate;
end;
```



When the default formatting, or adding text to a point with the XValueText property is not sufficient, you can implement an event that returns a string at a specific x-axis value. This can be applied for both the x-axis and the y-axis and is demonstrated in the code below based on the previous sample.

```
var
  s: TTMSFNCChartSerie;
  I: Integer;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  s := TMSFNCChart1.Series.Add;
  s.Mode := smStatistical;
  s.AutoXRange := arEnabled;
  s.AutoYRange := arEnabledZeroBased;
  s.XValues.MinorUnit := 0.5;
  s.XValues.MinorUnitFormat := '%.1f';
  s.YValues.AutoUnits := False;
  s.YValues.MajorUnit := 10;
  s.YValues.MinorUnit := 5;
  s.XValues.Angle := -90;
  for I := 0 to 9 do
    s.Points.Add;
  TMSFNCChart1.EndUpdate;
end;

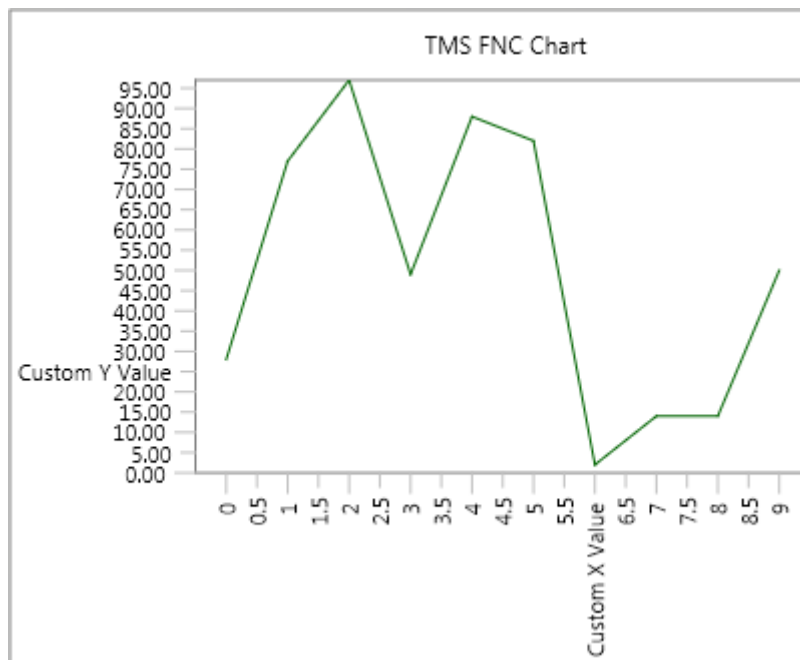
procedure TForm1.TMSFNCChart1GetSerieXValue(Sender: TObject;
  ASerie: TTMSFNCChartSerie; AIndex: Integer;
  AKind: TTMSFNCChartDrawXYValueKind; AValue: Double; var AValueString:
  string);
begin
  if (AKind = vkMajor) and (AValue = 6) then
    AValueString := 'Custom X Value';
end;

procedure TForm1.TMSFNCChart1GetSerieYValue(Sender: TObject;
  ASerie: TTMSFNCChartSerie; AIndex: Integer;
```

```

    AKind: TTMSFNCChartDrawXYValueKind; AValue: Double; var AValueString:
string);
begin
    if (AKind = vkMinor) and (AValue = 25) then
        AValueString := 'Custom Y Value';
end;

```



## Autorange

Each series has an AutoXRange and an AutoYRange property. By default the AutoXRange property is set to arDisabled and the AutoYRange is set to arEnabled. When one of those properties has a disabled auto range, the range is set with the MinX and MaxX properties for the x-axis, and the MinY and MaxY properties for the y-axis. Each range can be extended with the percentage variant for each property.

The autorange is especially useful for common ranges, ranges which have the same minimum and maximum for all series. Below is a sample with 2 series with random values which have a common range. This sample also shows each common range at the left and right side y-axis.

```

var
    s: TTMSFNCChartSerie;
    I, J: Integer;
begin
    TMSFNCChart1.BeginUpdate;
    TMSFNCChart1.Clear;
    for I := 0 to 1 do
    begin
        s := TMSFNCChart1.Series.Add;
        s.Mode := smStatistical;
        s.AutoYRange := arCommonZeroBased;
        s.XValues.MinorUnit := 0.5;
        s.XValues.MinorUnitFormat := '%.1f';
        s.YValues.AutoUnits := False;
    end;
end;

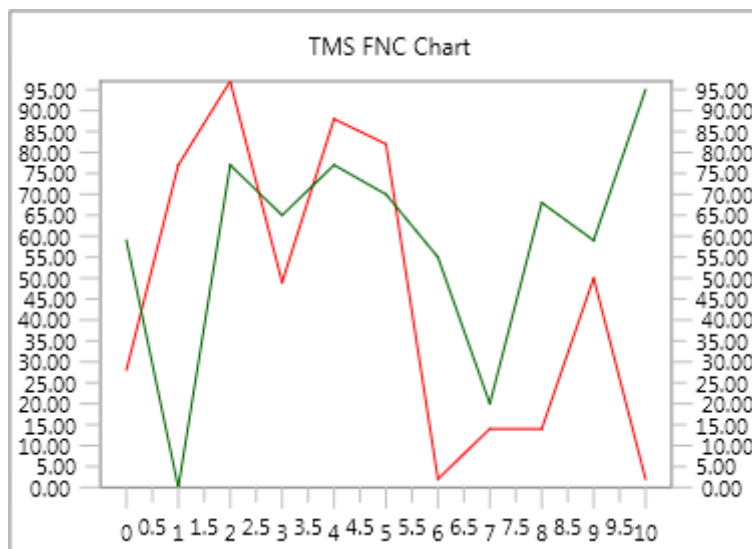
```

```

s.YValues.MajorUnit := 10;
s.YValues.MinorUnit := 5;
if I = 0 then
begin
    s.Stroke.Color := gcRed;
    s.YValues.Positions := [ypLeft, ypRight];
    s.XValues.Positions := [xpBottom];
end
else
begin
    s.YValues.Positions := [];
    s.XValues.Positions := [];
end;

for J := 0 to 10 do
    s.Points.Add;
end;
TMSFNCChart1.EndUpdate;
end;

```

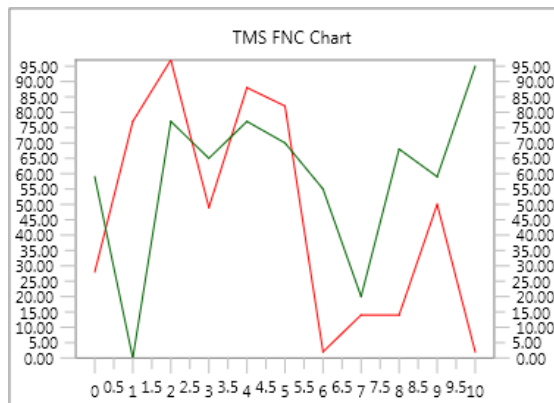




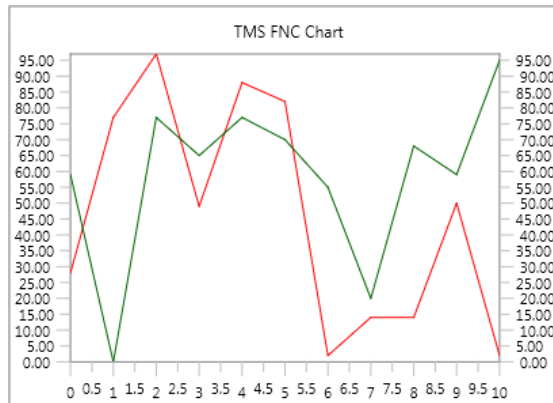
## Mathematical vs Statistical

The Chart can display each series in a different mode. The default mode for each series is Mathematical. In mathematical mode, the X-axis zero value is at the crossing point of X-axis and Y-axis and thus the first value is displayed at the Y-axis. Further, it uses the complete available width of the series rectangle. The alternative is Statistical mode, which automatically calculates and applies an offset on the x-axis to evenly distribute the values across the available chart width.

Statistical



Mathematical

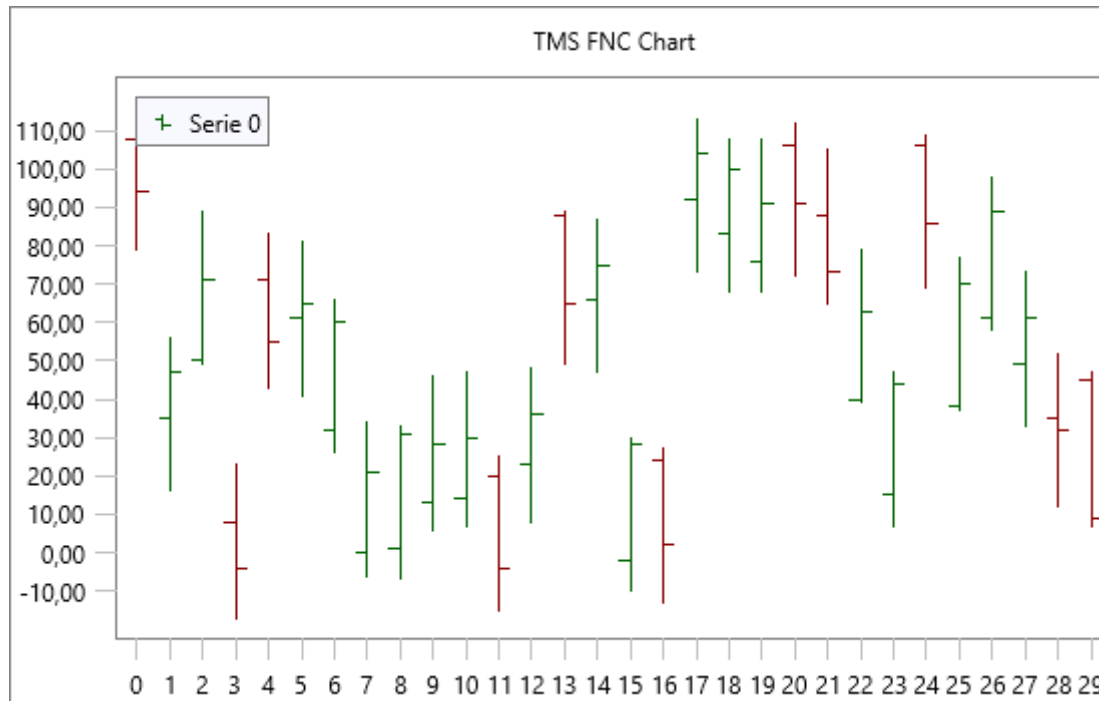


## Multi-Point Series

The Chart supports three types of multi-points series: ctOHLC, ctCandleStick and ctBoxPlot. Points can be added by using one of the AddMultiPoint overload methods. For the ctCandleStick and ctBoxPlot types, a separate increase and decrease fill and stroke color can be set under the series MultiPoints property. Below is a sample that demonstrates this.

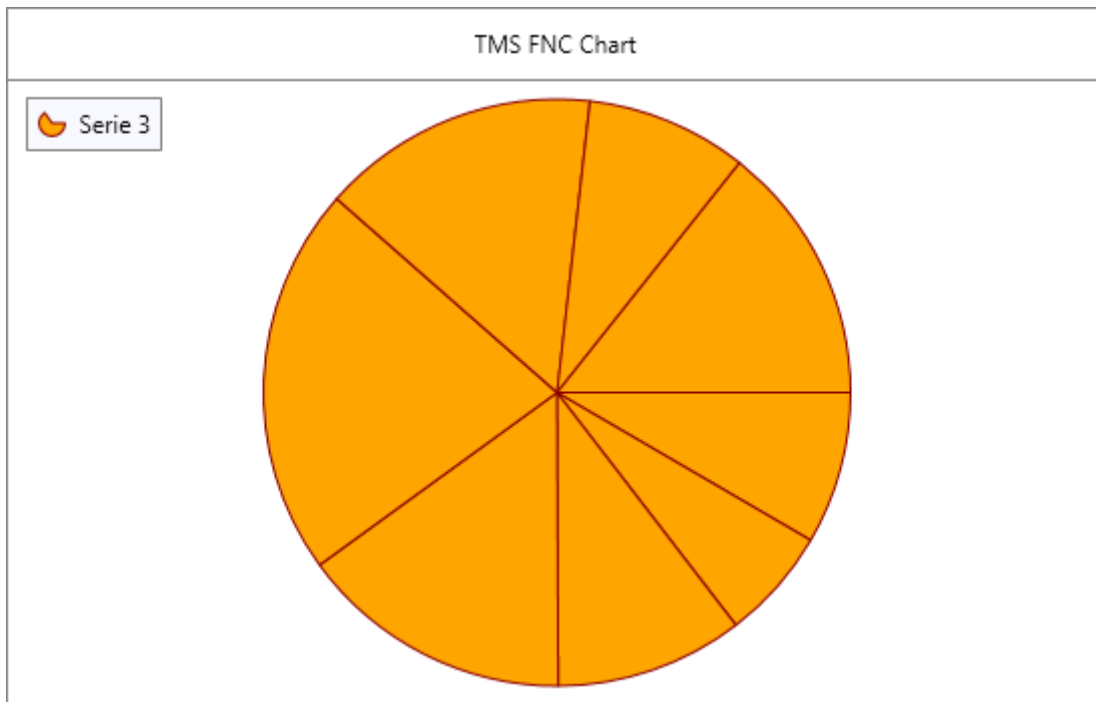
```
var
  s: TTMSFNCChartSerie;
  c: Integer;
  l: Integer;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Series.Clear;
  TMSFNCChart1.SeriesMargins.Left := 10;
  TMSFNCChart1.SeriesMargins.Top := 10;
  TMSFNCChart1.SeriesMargins.Right := 10;
  TMSFNCChart1.SeriesMargins.Bottom := 10;
  s := TMSFNCChart1.Series.Add;
  s.ChartType := ctOHLC;
  s.AutoXRange := arCommonZeroBased;
  s.AutoYRange := arCommon;
  for l := 0 to 29 do
    begin
      c := Random(100);
      if Random(c) mod (Random(10) + 1) = 0 then
        s.AddMultiPoint(c + Random(20), c + 20, c - 20, c - Random(20))
      else
        s.AddMultiPoint(c - Random(20), c + 20, c - 20, c + Random(20));
```

```
end;  
TMSFNCChart1.EndUpdate;
```



## Pie

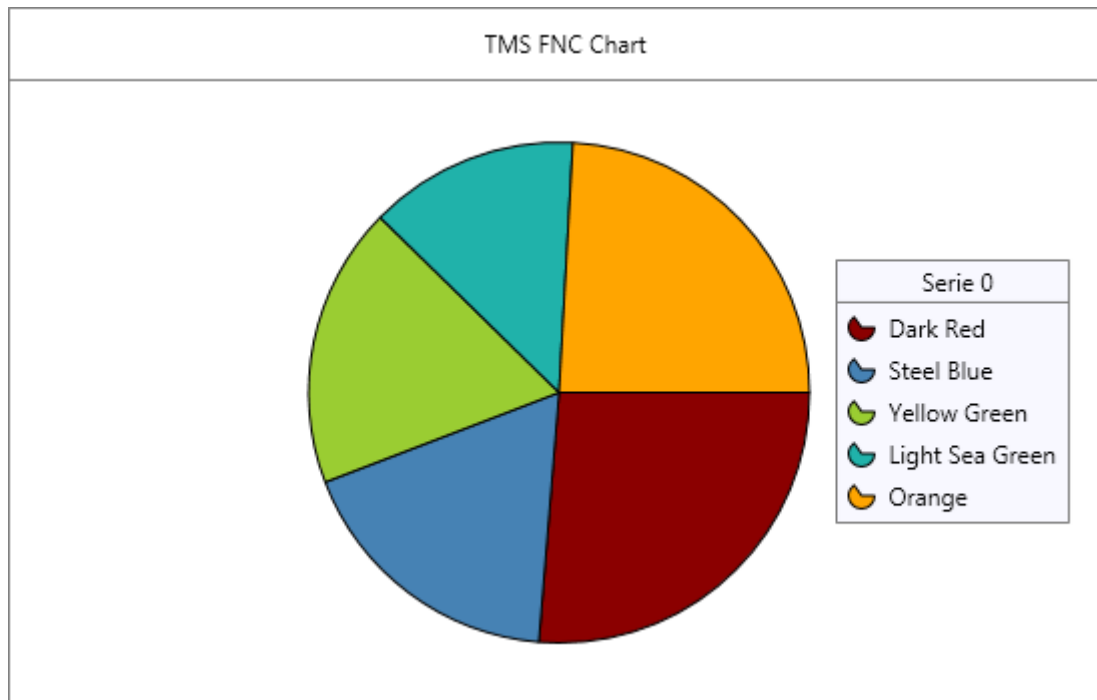
Changing the chart-type of one or multiple series to ctPie will automatically hide the x- and y-axis and the x- and y-grid. By default, the slices of the Pie will automatically take over the fill color of the series.



By default, the points that are added when initializing the series are reflected as separate slices. Adding points can be done with one of the AddPoint overloads. The general properties of a pie series can be changed at the Pie property. In the below sample, the main legend is hidden, the pie legend is shown and the colors for each slice are changed.

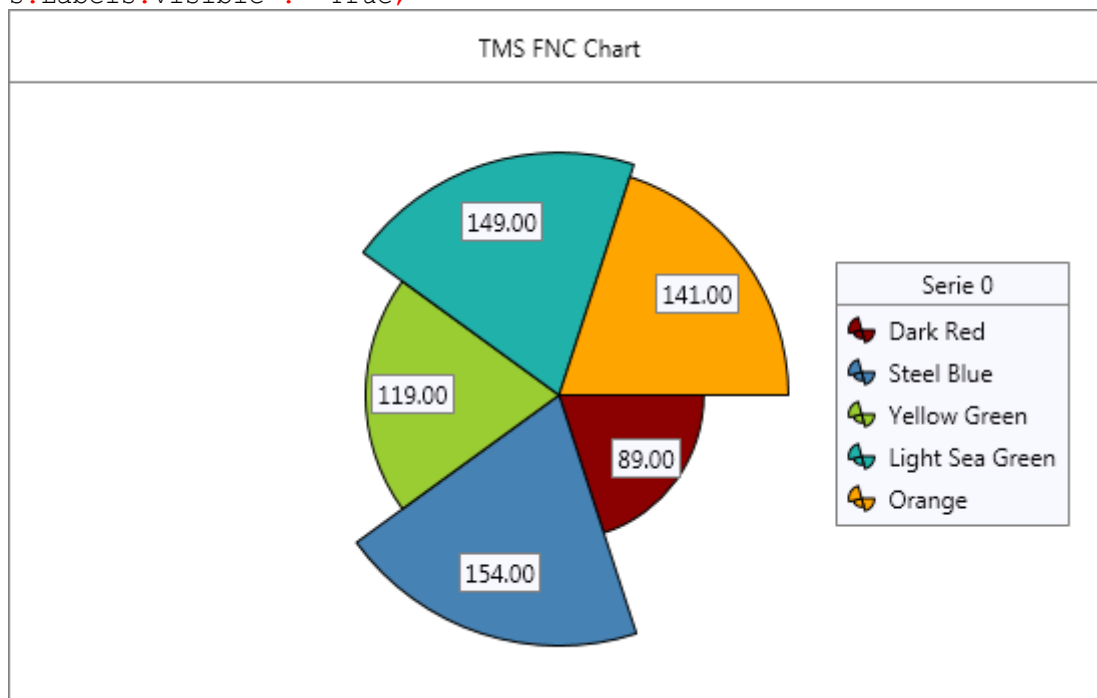
```
var
    s: TTMSFNCChartSerie;
begin
    TMSFNCChart1.BeginUpdate;
    TMSFNCChart1.Legend.Visible := False;
    TMSFNCChart1.Series.Clear;
    s := TMSFNCChart1.Series.Add;
    s.ChartType := ctPie;
    s.Pie.Size := 250;
    s.Points.Clear;
    s.Legend.Visible := True;
    s.Pie.AutoSize := False;
    s.Pie.Size := 250;
    s.Stroke.Color := gcBlack;
    s.AddPoint(Random(100) + 40, gcDarkred, 'Dark Red');
    s.AddPoint(Random(100) + 40, gcSteelblue, 'Steel Blue');
    s.AddPoint(Random(100) + 40, gcYellowgreen, 'Yellow Green');
    s.AddPoint(Random(100) + 40, gcLightseagreen, 'Light Sea Green');
    s.AddPoint(Random(100) + 40, gcOrange, 'Orange');
    TMSFNCChart1.EndUpdate;
```

**end;**



Labels and annotations are supported in the same way as they are for the other chart types. You simply add annotations and/or turn on the labels by setting the visible property to true.

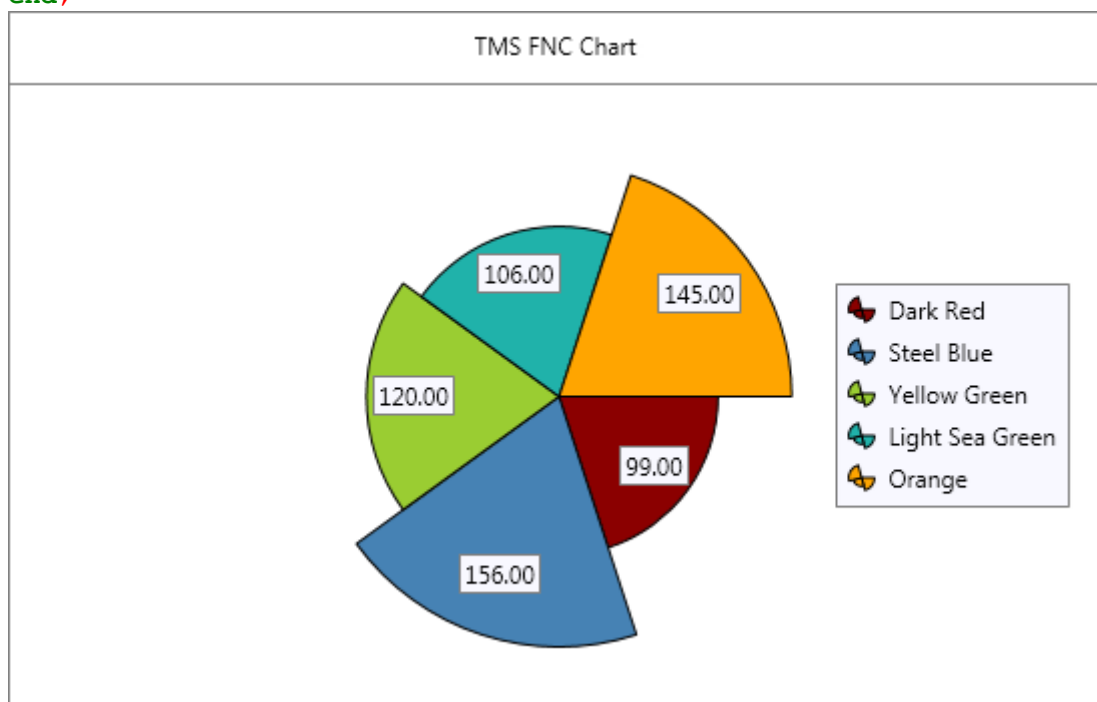
```
s.Labels.Visible := True;
```



There are 2 variants for this charttype that allow handling more types of data. When setting the ChartType to ctSizedPie, the slices are equally divided, but the radius depends on the value of the

point. When setting the ChartType to ctVariableRadiusPie, the slices are calculated as they are with a normal pie chart, but the radius can be controlled with an additional property called YValueVariable. This property can be accessed through the AddVariablePoint overload or directly at point level. Below is a sample of a ctSizedPie.

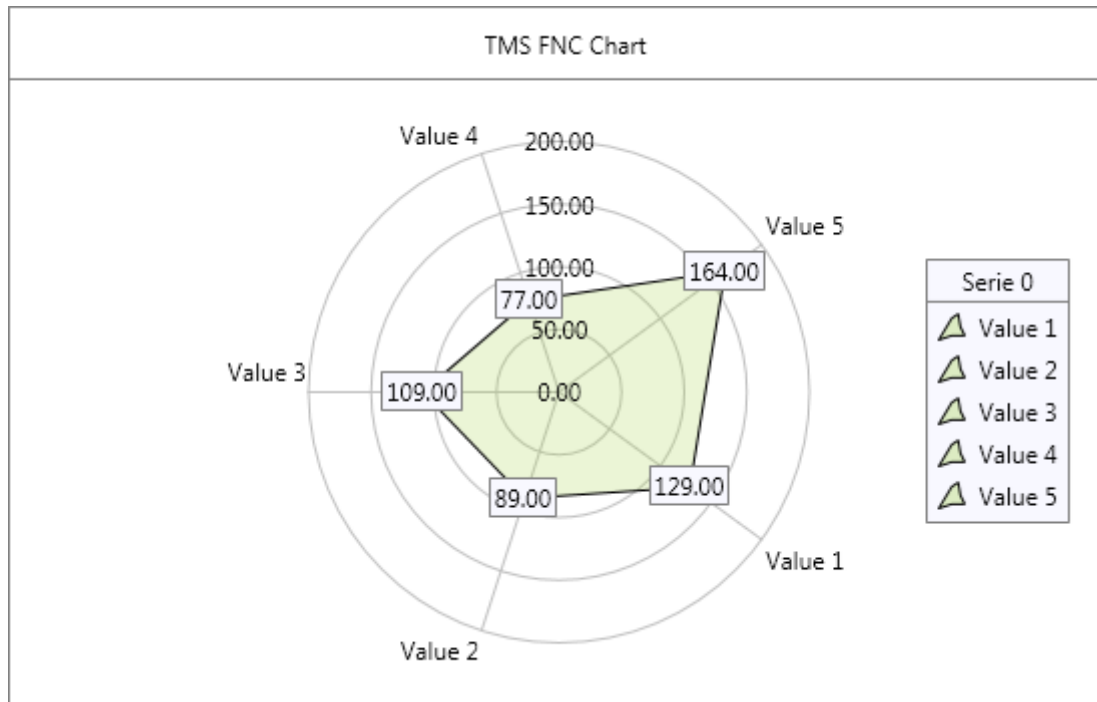
```
var
  s: TMSFNCChartSerie;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Legend.Visible := False;
  TMSFNCChart1.Series.Clear;
  s := TMSFNCChart1.Series.Add;
  s.ChartType := ctSizedPie;
  s.Pie.Size := 400;
  s.Points.Clear;
  s.Legend.Visible := True;
  s.Pie.AutoSize := False;
  s.Pie.Size := 250;
  s.Stroke.Color := gcBlack;
  s.Labels.Visible := True;
  s.Labels.OffsetX := 0;
  s.Labels.OffsetY := 0;
  s.AddPoint(Random(100) + 75, gcDarkred, 'Dark Red');
  s.AddPoint(Random(100) + 75, gcSteelblue, 'Steel Blue');
  s.AddPoint(Random(100) + 75, gcYellowgreen, 'Yellow Green');
  s.AddPoint(Random(100) + 75, gcLightseagreen, 'Light Sea Green');
  s.AddPoint(Random(100) + 75, gcOrange, 'Orange');
  TMSFNCChart1.EndUpdate;
end;
```



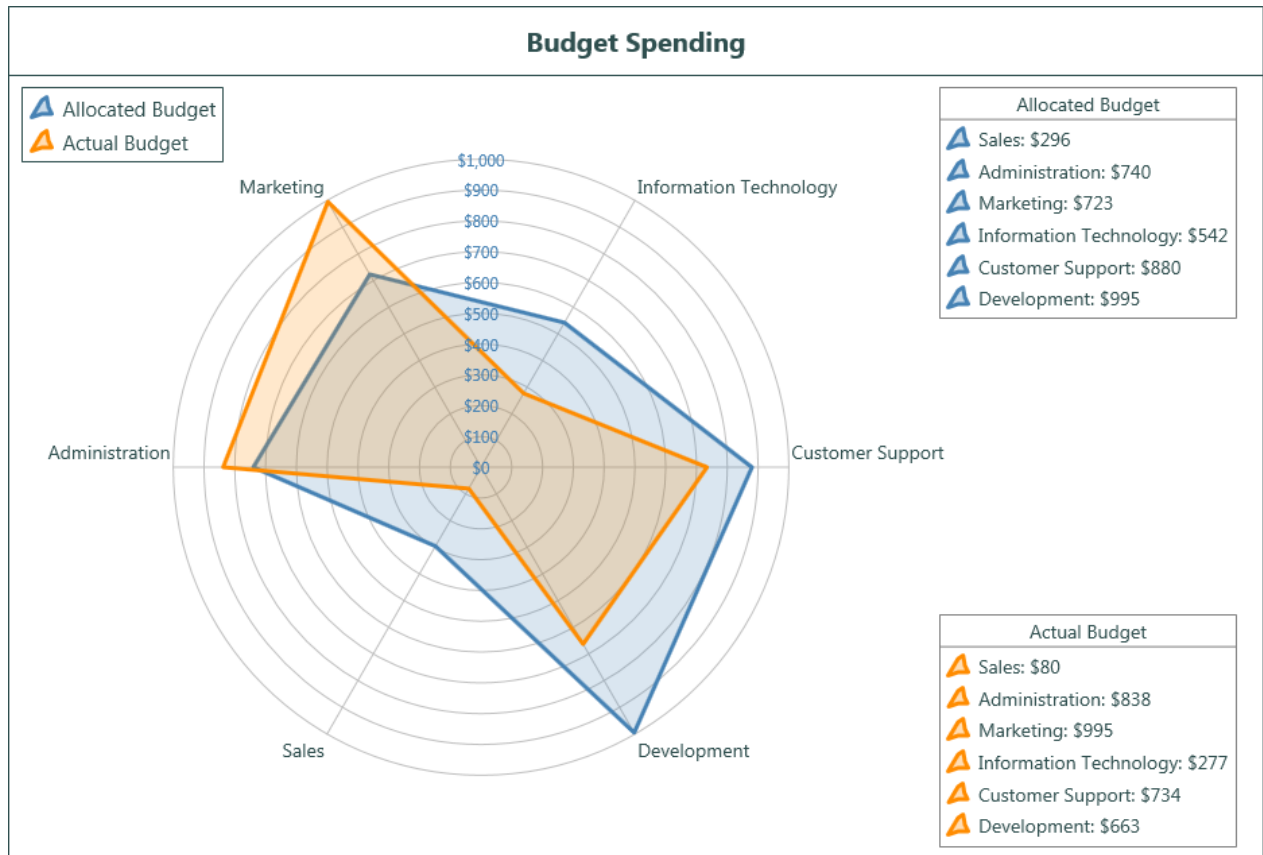
## Spider

When changing the ChartType property to ctSpider, the properties of the YValues and YGrid properties are used to configure the visuals of the grid, while the start, sweep angles and dimensions are stored under the Pie property. Additionally, the Spider\* properties under YValues and YGrid are used to further fine-tune spider chart specific features such as the grid kind and the values rotation angle. Applying the ctSpider chart type on the previous sample, changing the value labels, and applying the Spider properties to change Spider specific features generates the output below.

```
var
  s: TTMSFNCChartSerie;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Legend.Visible := False;
  TMSFNCChart1.Series.Clear;
  s := TMSFNCChart1.Series.Add;
  s.ChartType := ctSpider;
  s.Pie.Size := 400;
  s.Points.Clear;
  s.Legend.Visible := True;
  s.Pie.AutoSize := False;
  s.Pie.Size := 250;
  s.Fill.Opacity := 0.2; (FireMonkey only)
  s.Stroke.Color := gcBlack;
  s.Labels.Visible := True;
  s.Labels.OffsetX := 0;
  s.Labels.OffsetY := 0;
  s.YGrid.SpiderLegend := True;
  s.MaxY := 200;
  s.AutoYRange := arDisabled;
  s.YValues.AutoUnits := False;
  s.YValues.MajorUnit := 50;
  s.YValues.MinorUnit := 0;
  s.AddPoint(Random(100) + 75, 0, 'Value 1');
  s.AddPoint(Random(100) + 75, 0, 'Value 2');
  s.AddPoint(Random(100) + 75, 0, 'Value 3');
  s.AddPoint(Random(100) + 75, 0, 'Value 4');
  s.AddPoint(Random(100) + 75, 0, 'Value 5');
  TMSFNCChart1.EndUpdate;
end;
```



The ctSpider chart type follows the AutoYRange property to determine the maximum. When set to arDisabled (default), the MaxY property can be used to set a maximum for the grid drawing and spider chart calculation. Additionally, the enabled and common auto ranges can be used in combination with the stacked property on Pie level to compare multiple series in one chart. The above screenshot demonstrates this, and the code to accomplish this is demonstrated in the “Desktop” demo, available after installation.








## Legend

The legend displays the amount of series, each with their own legend text. If you want to display a legend for each series, with an entry for each point, you can turn off the Legend on chart level, and turn on the legend on series level. The properties are identical, and have the same customization events. In the previous chapter on the Pie chart type, the legend was displayed at the right side of the chart, with the typical chart type icon next to each entry. If you want further customization for this icon, you can override the `OnBeforeDrawSerieLegendIcon`. Below is a sample that demonstrates this.

```
procedure TForm1.TMSFNCChart1BeforeDrawSerieLegendIcon(Sender: TObject;
  ACanvas: TCanvas; ASerie: TTMSFNCChartSerie; APoint: TTMSFNCChartPoint;
  ARect: TRectF; var ADefaultDraw: Boolean);
begin
  ADefaultDraw := False;
  ACanvas.FillRect(ARect, 2, 2, AllCorners, 1);
  ACanvas.DrawRect(ARect, 2, 2, AllCorners, 1);
end;
```



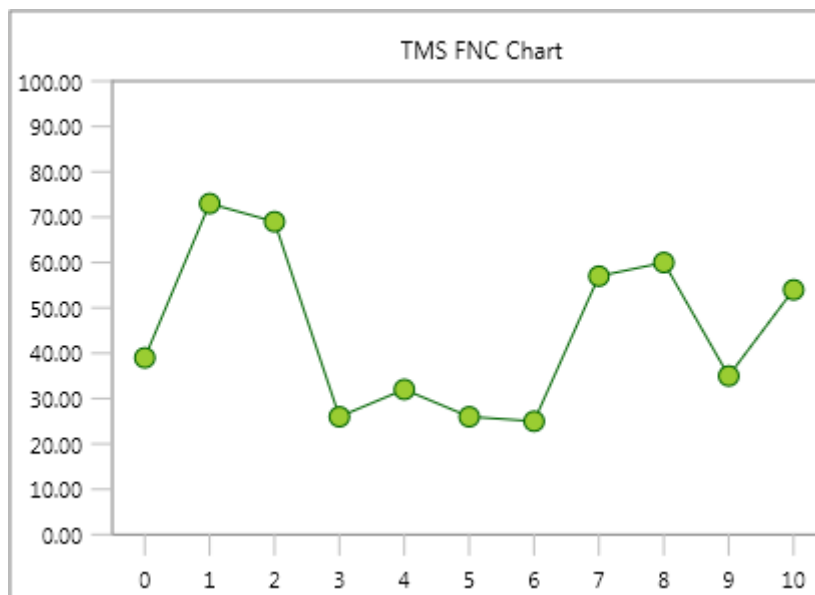
Serie 0	
	Dark Red
	Steel Blue
	Yellow Green
	Light Sea Green
	Orange

## Markers

Each series has the ability to show markers. Markers have an ellipse shape by default but can be changed to draw a triangle, square, diamond and an image. Further customization can be achieved through the `OnBeforeDrawSerieMarker` and the `OnAfterDrawSerieMarker`.

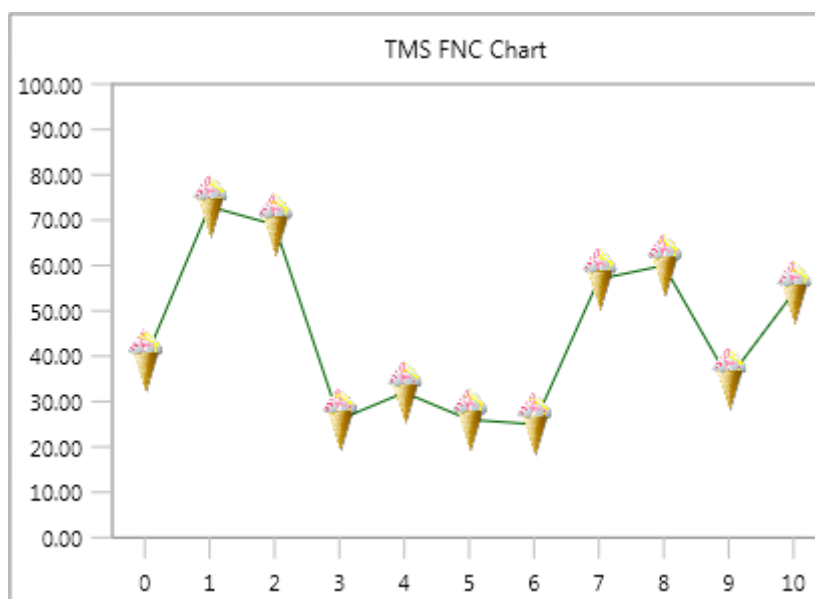
Below is a sample that shows how to enable the markers on a serie and how to change the shape to a bitmap.

```
var
  s: TTMSFNCChartSerie;
  I: Integer;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  s := TMSFNCChart1.Series.Add;
  s.Mode := smStatistical;
  s.Markers.Visible := True;
  s.AutoYRange := arDisabled;
  s.MinY := 0;
  s.MaxY := 100;
  for I := 0 to 10 do
    s.AddPoint(RandomRange(25, 75));
  TMSFNCChart1.EndUpdate;
end;
```



```
var
  s: TTMSFNCChartSerie;
  I: Integer;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  s := TMSFNCChart1.Series.Add;
  s.Mode := smStatistical;
```

```
s.Markers.Visible := True;
s.Markers.Bitmap.LoadFromFile('icecream.png');
s.Markers.Width := 32;
s.Markers.Height := 32;
s.Markers.Shape := msBitmap;
s.AutoYRange := arDisabled;
s.MinY := 0;
s.MaxY := 100;
for I := 0 to 10 do
    s.AddPoint(RandomRange(25, 75));
TMSFNCChart1.EndUpdate;
end;
```



## Stacked series

When choosing a bar or area chart, and adding multiple series, you are able to combine those series in stacked variants based on the type of chart and the GroupIndex property on series level. The requirement is that each of those combined series have the same range, and have a value that is larger than 0. There are 2 types of stacked charts: the normal stacked charts combine the values for each group and the percentage stacked charts that represent the values of each series as a percentage of a maximum of 100.

Below is a sample that adds 4 bar series in 2 stacked groups.

```
var
    s: TTMSFNCChartSerie;
    I, J: Integer;
begin
    TMSFNCChart1.BeginUpdate;
    TMSFNCChart1.Clear;
    for I := 0 to 3 do
        begin
            s := TMSFNCChart1.Series.Add;
            s.ChartType := ctStackedBar;
            s.Mode := smStatistical;
```

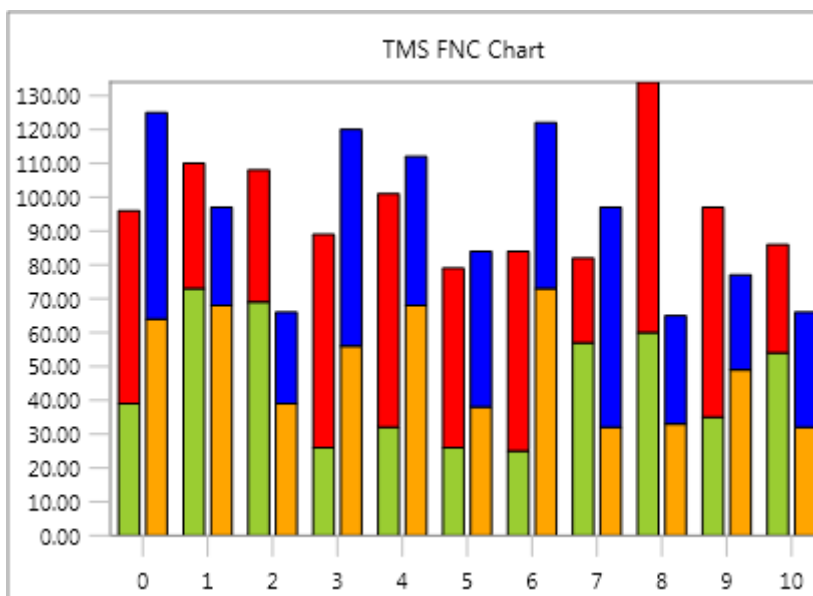
```
s.AutoYRange := arCommonZeroBased;
s.GroupIndex := I div 2;

case I of
1: s.Fill.Color := gcRed;
2: s.Fill.Color := gcOrange;
3: s.Fill.Color := gcBlue;
end;

s.Stroke.Color := gcBlack;

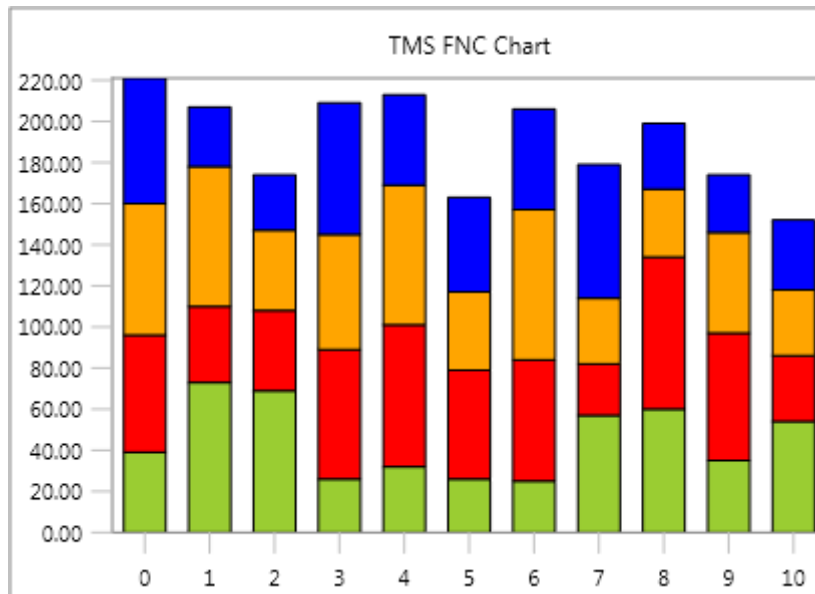
if I > 0 then
begin
s.YValues.Positions := [];
s.XValues.Positions := [];
end;

for J := 0 to 10 do
s.AddPoint(RandomRange(25, 75));
end;
TMSFNCChart1.EndUpdate;
end;
```

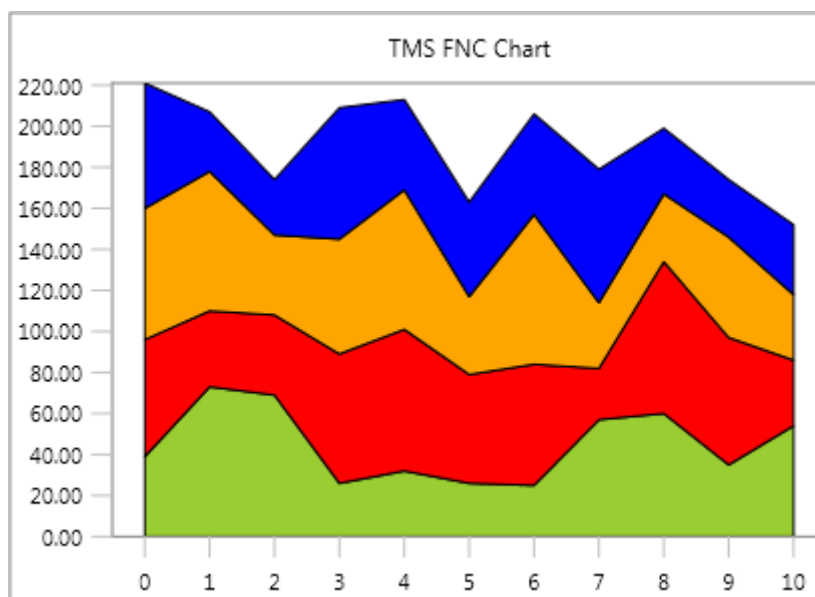


In this sample, the GroupIndex is 0 for the first 2 series and 1 for the last 2 series resulting in a multi-group stacked bar series Chart.

Changing the GroupIndex property to 0 for all series gives the result below.



For an area type, the GroupIndex doesn't have any effect. With this type, all series are stacked like the previous sample with a GroupIndex that equals 0 for all series.



The second type of stacked series are based on a minimum of 0 and a maximum of 100. The values that needs to be added can remain identical to the previous stacked chart and will internally be recalculated to match the 0 to 100 range. Below is a sample that demonstrates this.

```
var
  s: TTMSFNCChartSerie;
  I, J: Integer;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  for I := 0 to 3 do
  begin
```

```

s := TMSFNCChart1.Series.Add;
s.ChartType := ctStackedPercentageBar;
s.Mode := smStatistical;
s.AutoYRange := arCommonZeroBased;
s.GroupIndex := I div 2;

case I of
1: s.Fill.Color := gcRed;
2: s.Fill.Color := gcOrange;
3: s.Fill.Color := gcBlue;
end;

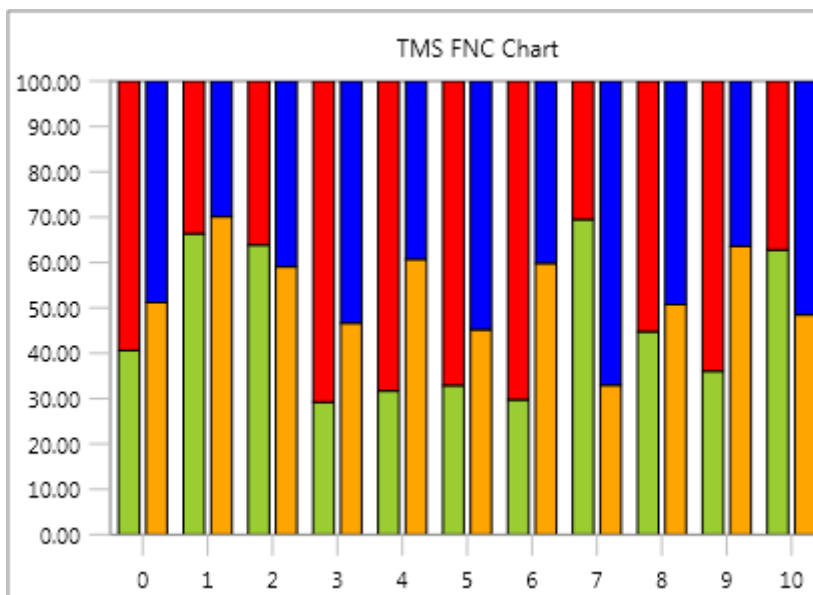
s.Stroke.Color := gcBlack;

if I > 0 then
begin
s.YValues.Positions := [];
s.XValues.Positions := [];
end;

for J := 0 to 10 do
s.AddPoint(RandomRange(25, 75));
end;

TMSFNCChart1.EndUpdate;
end;

```



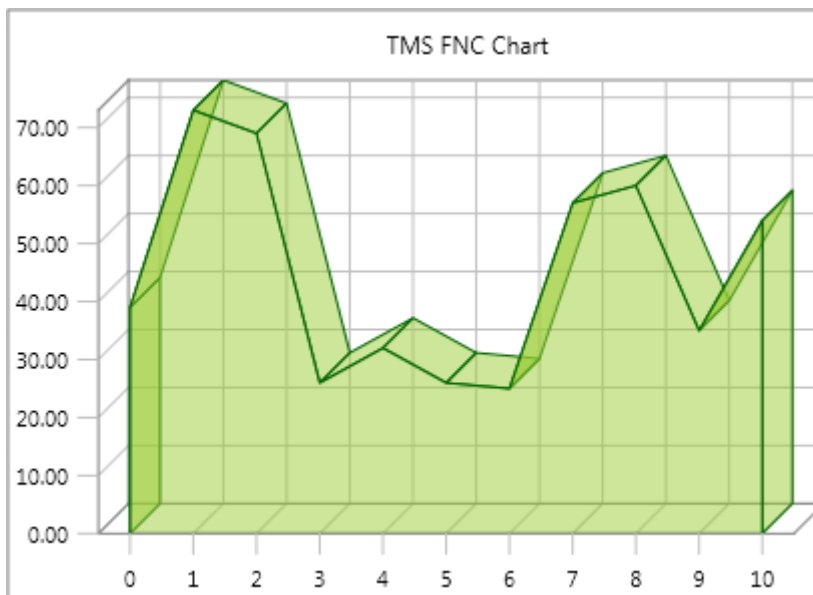
## 3D

Each series has an Enable3D property that enables 2D rendering of a 3D like environment. The Offset3DX and Offset3DY properties on serie level determine the “depth” of the 3D visualization. Using the following code on an area chart, displays the chart with the following result:

```
var
  s: TMSFNCChartSerie;
  I: Integer;
begin
  TMSFNCChart1.BeginUpdate;
  TMSFNCChart1.Clear;
  s := TMSFNCChart1.Series.Add;
  s.ChartType := ctArea;
  s.Mode := smStatistical;
  s.AutoYRange := arEnabledZeroBased;
  s.Enable3D := True;
  s.XGrid.Visible := True;
  s.YGrid.Visible := True;
  s.Fill.Opacity := 0.5; (FireMonkey only)

  for I := 0 to 10 do
    s.AddPoint(RandomRange(25, 75));

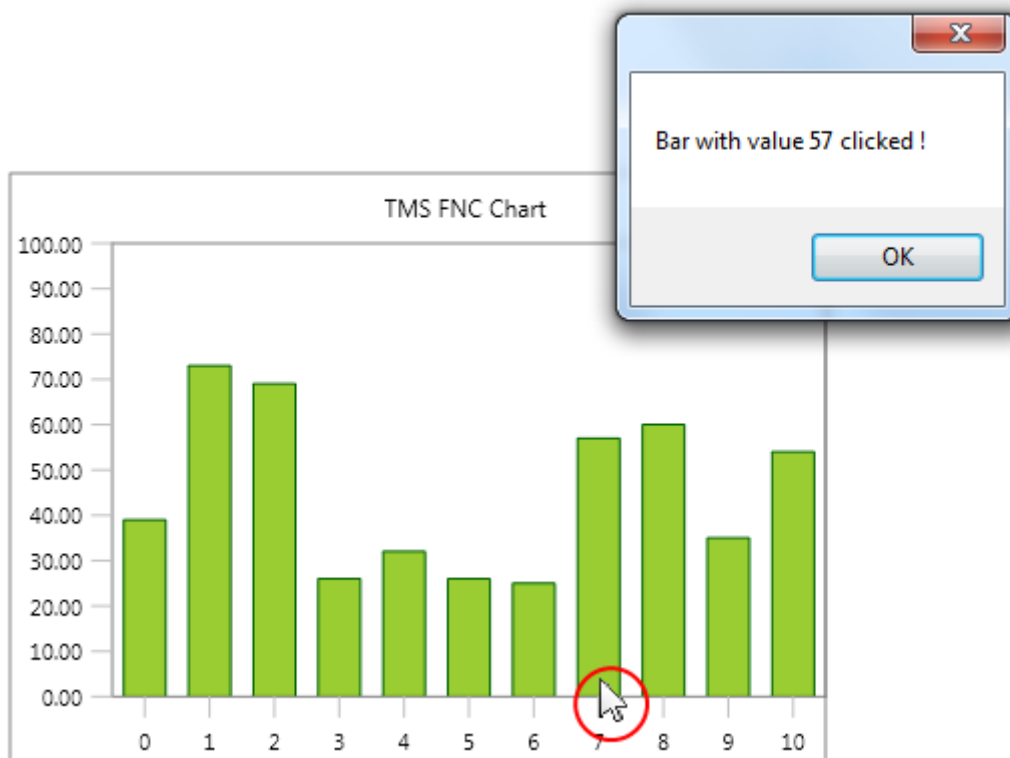
  TMSFNCChart1.EndUpdate;
end;
```



## Interaction

The Chart supports interaction, which is enabled by default. Interaction enables click detection on a point or a bar and triggers the appropriate event. Additionally, panning and scaling can be performed in X and Y direction, depending on the property values under `InteractionOptions`. Below is a sample that implements the `OnSerieBarClick` event.

```
procedure TForm1.TMSFNCChart1SerieBarClick(Sender: TObject;  
    APoint: TTMSFNCChartPoint);  
begin  
    ShowMessage('Bar with value ' + floattostr(APoint.YValue) + ' clicked  
    !');  
end;
```



The alternative for all other chart types (except for the pie variants) is the `OnSeriePointClick` event which detects clicks on a point that lies within the `ClickMargin` boundaries relative from the point X and Y value. The `ClickMargin` is a property on Chart level. For the pie variants, the `OnSerieSliceClick` is executed.