TMS FNC Dashboard Controls
DEVELOPERS GUIDE

January 2022

Web: http://www.tmssoftware.com
Email: info@tmssoftware.com

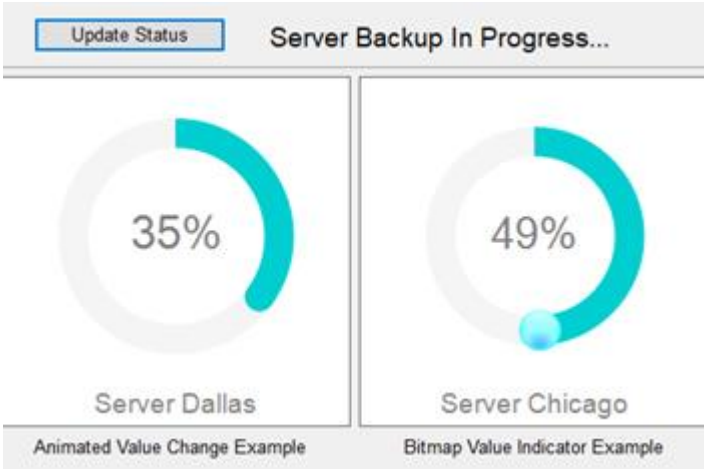
Index

Availability

Supported frameworks and platforms
- VCL Win32/Win64
- FMX Win32/Win64, macOS, iOS, Android, Linux
- LCL Win32/Win64, macOS, iOS, Android, numerous Linux variants including Raspbian
- WEB: Chrome, Edge, Firefox, …

Supported IDE's
- Delphi XE7 and C++ Builder XE7 or newer releases
- Lazarus 1.4.4 with FPC 2.6.4 or newer official releases
- TMS WEB Core for Visual Studio Code 1.3 or newer releases

TTMSFNCWidgetProgress



Introduction

TTMSFNCWidgetProgress displays the progress of a process graphically in the form of a circle. To use this control, just drop it on a form and use the Value property to show the progress. The above screenshot of the Demo shows two usage examples of this control.

Features

- Easy to use, just set the Value property to set the progress
- Option to animate the display of change in progress value
- Properties to customize the Value Indicator on the circle by a filled circle (dot) or a custom bitmap. The second control in the demo screenshot above shows a custom bitmap.
- Option to display a Caption text at the top or bottom
- Properties to customize the font and format of the displayed value
- Properties to customize the fill colors and thickness of the finished and unfinished portion of the progress circle
- Properties to change the circle extent by specifying the Start and End angles
- Properties to show tick marks around the circle with bigger starting and ending tick marks (similar to a meter) with their own captions

Demo

A Demo is included in the source in a subfolder “Demo\WidgetProgress.” A screenshot is shown at the top. In this Demo, two WidgetProgress controls display backup progress of two different servers. The first control has a default look whereas the second control shows a custom bitmap (a blue ball) on the circle at the value position. With each click of the button “Update Status,” the backup progresses and advances both the WidgetProgress controls. You will see that the first control advances to the next value position with an animation whereas the second control progresses immediately to the next position. It also displays a custom bitmap (a blue ball) on the circle at the value position as shown in the screenshot.

Properties

How to overview:

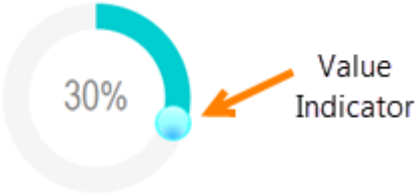
- To show a progress value, set it to the Value property. Control the display of the value text by using the properties ValueFont and ValueFormat.
- Control the display of actual position of the value on the circle by using the property group ValueIndicator. You can show a bigger Dot with a different fill or a custom Bitmap at the Value position. The second control in the demo screenshot above shows a custom bitmap.
- Normally when you set a new value, the control immediately advances the position to the new value. But you can make it advance with an animation by activating the property ValueAnimation.
- Adjust the thickness and color fill of the finished and unfinished portions of the progress circle by the property group CircleOptions.
- Show a Caption by the CaptionOptions property group. The caption can be placed at the top or at the bottom of the control.
- Advanced features consist of controlling the Start and End angle of the progress circle, showing tickmarks around the circle, showing captions at the Start, End and Center of the tickmarked circle.

Value: sets the value of the progress in percent and displays a circular progress arc accordingly.

ValueFormat: sets the format string to display the value text at the center of the circle. The default format "%g%%" displays the value with a percent symbol. Note that 2 percent symbols at the end of this default format display a single percent symbol.

ValueFont: sets the font to display the value text at the center of the circle.

ValueIndicator: sets the display of the actual position on the circle corresponding to the value.

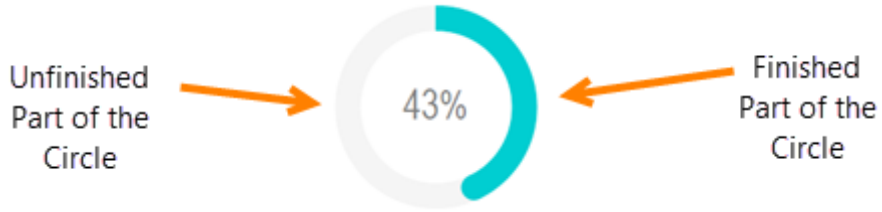


The settings for the above include:

- **Kind**: The choices are piDot, piBitmap and piNone. The default is piDot that displays a circle or Value Dot at the value position. If you select piBItmap, it allows you to display a custom bitmap assigned via the property BitmapName described below.
- **Size**: The diameter of the Value Dot if the Kind is piDot. The default size is same as the default thickness of the circle.
- **Fill**: sets the color of the Value Dot if the Kind is piDot.
- **BitmapName**: sets the name of a bitmap in a TMS BitmapContainer assigned to the WidgetProgress control. This bitmap is displayed as the Value Indicator if you set the kind above to piBitmap.

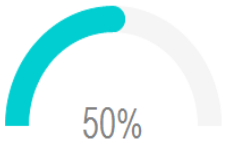
ValueAnimation: set this Active to let the circle progress to a new value with Animation.

CircleOptions: includes properties to control the display of the progress circle.



The settings for the above include:

- **Fill**: sets the color of the Finished Part of the Circle.
- **UnfinishedFill**: sets the color of the Unfinished Part of the Circle.
- **Thickness**: sets the thickness or width of the Finished Part of the Circle Arc.
- **UnfinishedThickness**: sets the thickness or width of the Unfinished Part of the Circle Arc.
- **Margin**: sets the space to leave on all the sides of the widget.
- **StartAngle**: sets the angle in degrees of the starting point of the progress (value 0). The default value is angle 0 (top).
- **EndAngle**: sets the angle in degrees of the ending point of the progress (value 100). The default value is angle 360 (ends at top).
  Example: if you set the StartAngle at -90 and EndAngle at 90 the control will appear as shown on the right.



Tickmarks: includes properties to display tick marks around the progress circle.

- **Count**: when set to a Non Zero value, shows that many tick marks around the progress circle.
- **Size**: sets the length of the tick marks.
- **Stroke**: sets the stroke to draw tickmarks
- **Gap**: sets the gap between the circle and the start of a tick mark.

- **Additional Tickmarks properties**

  CenterText: sets the text to show at the center of the tick mark circle below the Value text.

  Tickmarks Properties related to Start/End marks: If you set the earlier described properties CircleOptions.StartAngle and CircleOptions.EndAngle to a non-complete circle then you can use the following additional properties in TickMarks.

  StartEndMarkSize: sets a different size (length) of the starting and ending tick mark as

shown above. This is usually done with a Start and End angle that defines a non-complete circle.
StartEndMarkWidth: sets a different width of the starting and ending tick mark.
StartText: sets a text label to show near the starting mark.
EndText: sets a text label to show near the ending mark.

CaptionOptions: sets a caption to show on the widget. This includes the following settings:

- **Text**: sets the text for the caption
- **Font**: sets the font for the caption
- **Position**: sets the position for the caption to Top (cpTop) or Bottom (cpBottom) or cpNone to hide the caption even if the text is set.

When you set a caption, the size of the widget is reduced to make space for the caption.

Code Snippets

To set a progress value to show:

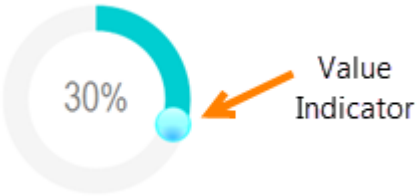
```
TMSFNCWidgetProgress1.Value := 45;
```

To set the caption:

```
TMSFNCWidgetProgress1.CaptionOptions.Text := 'Server Dallas';
```

To show a bitmap as the ValueIndicator: following code was used in the Demo above to show the Widget Progress on the right. The BitmapContainer was already added to the form in the Designer with the desired BlueBall named bitmap.

```
TMSFNCWidgetProgress2.BitmapContainer := TMSFNCBitmapContainer1;
TMSFNCWidgetProgress2.ValueIndicator.BitmapName := 'BlueBall';
TMSFNCWidgetProgress2.ValueIndicator.Kind := piBitmap;
```



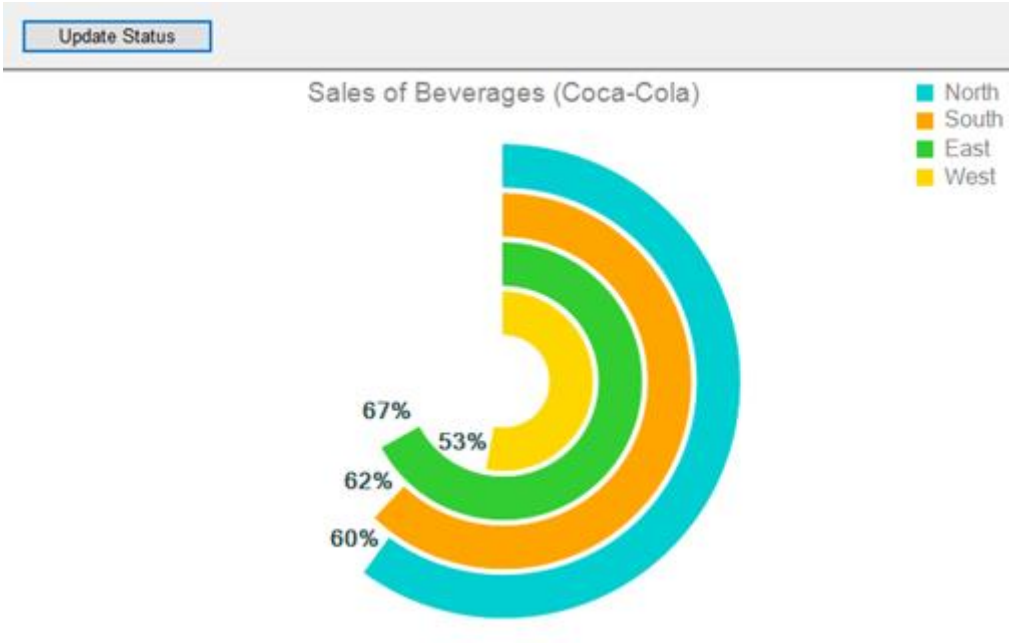
To change the color scheme completely with a dark gradient background and tick marks:

```
TMSFNCWidgetProgress1.Value := 30;
TMSFNCWidgetProgress1.Fill.Color := gcNavy;
TMSFNCWidgetProgress1.Fill.ColorTo := gcRoyalBlue;
TMSFNCWidgetProgress1.Fill.Kind := gfkGradient;
TMSFNCWidgetProgress1.CircleOptions.Fill.Color := gcWhite;
TMSFNCWidgetProgress1.CircleOptions.Fill.ColorTo := gcPowderBlue;
TMSFNCWidgetProgress1.CircleOptions.Fill.Kind := gfkGradient;
TMSFNCWidgetProgress1.CircleOptions.UnfinishedFill.Color := gcNavy;
TMSFNCWidgetProgress1.CircleOptions.Thickness := 5;
TMSFNCWidgetProgress1.CircleOptions.UnfinishedThickness := 5;
TMSFNCWidgetProgress1.ValueIndicator.Fill.Color := gcPowderBlue;
TMSFNCWidgetProgress1.ValueFont.Color := gcPowderBlue;
TMSFNCWidgetProgress1.Tickmarks.Count := 10;
TMSFNCWidgetProgress1.Tickmarks.Stroke.Color := gcPowderBlue;
TMSFNCWidgetProgress1.Tickmarks.Size := 12;
```

To make it a half circle as shown above, the following lines are added:

```
TMSFNCWidgetProgress1.CircleOptions.StartAngle := -90;
TMSFNCWidgetProgress1.CircleOptions.EndAngle := +90;
```

TTMSFNCWidgetMultiProgress



Introduction

TTMSFNCWidgetMultiProgress displays the progress of multiple processes graphically in concentric circles as shown above. To use this control, just drop it on a form and add Circle Items to it. Then set a Value for each item to show the progress in the Form Designer or by code. Later, you can set the Value for each item by code, quite similar to the WidgetProgress described earlier. A Legend can also be shown in various positions. In the screenshot, the Legend appears on the top right.

Features

- Easy to use, just add a circle item in the Object Inspector or by code. Set its Value and Caption to show the progress immediately. The circle gets a new color automatically.
- Properties to customize the fill colors and thickness of the finished and unfinished portion of any circle item.
- Option to display a Caption text at the top or bottom
- Show a Legend showing the circle item captions. The Legend can appear in 4 corners or on the circle itself. For example, the Legend appears at the top right in the above screenshot.

Demo

A Demo is included in the source in a subfolder “Demo\WidgetMultiProgress.” A screenshot is shown at the top. In this Demo, four progress circles display the sales figures of different regions. In the Demo, you can click on the button “Update Status” to advance all the progress circles by random values.

Properties

How to overview:

- Add a different circle item in the Designer or in code.
- Set a value to show the progress of an item.
- A new color is automatically assigned for a new circle item but you can change it by the Fill property of the item. Similarly, you can change the color fill of the unfinished part of the circle item.
- Adjust the thickness of the finished and unfinished portions of all the circles by the property group CircleOptions.
- Show a Caption by the CaptionOptions property group. The caption can be placed at the top or at the bottom of the control.
- Show a Legend describing the items at various positions. Note that a Legend will not be shown if the Captions of all the circle items are empty.

CircleItems: contains items to show progress of each circle. You can click the … after the property in the Object Inspector to manage the items, to add, reorder or delete items. You can also expand the property in the Structure window of Delphi IDE to inspect the circle items. Note that the outermost circle is the Item at index 0.

Each circle item includes the following properties:

- **Caption**: sets the caption for the circle item. This is shown in the Legend. For example, "North" is a caption describing the outermost circle in the above screenshot. The captions appear only in the Legend and the Legend is shown only if you set at least one item's Caption.
- **Value**: sets the value of the progress in percent and displays a circular progress arc for the item accordingly.
- **Fill**: sets the color of the Finished Part of the Circle.
- **UnfinishedFill**: sets the color of the Unfinished Part of the Circle. Note that by default, this widget does not show any unfinished part of the circle.

ValueFormat: sets the format string to display the progress value text at the end of a circle item. The default format "%g%%" displays the value with a percent symbol. Note that 2 percent symbols at the end of this default format display a single percent symbol.

ValueFont: sets the font to display the value text at the end of a circle item.

CircleOptions: includes common properties to control the display of the progress circle of all the circle items.

- **Thickness**: sets the thickness or width of the Finished Part of the Circle Arc.
- **UnfinishedThickness**: sets the thickness or width of the Unfinished Part of the Circle Arc. Note that by default, this widget does not show any unfinished part of the circle.
- **ValueGap**: sets the distance between the value text and the end of the circle.
- **Margin**: sets the space to leave on all the sides of the widget.
- **StartAngle**: sets the angle in degrees of the starting point of the progress (value 0). The default value is angle 0 (top).
- **EndAngle**: sets the angle in degrees of the ending point of the progress (value 100). The default value is angle 360 (ends at top).

Legend: includes properties to display the Legend
- **Border**: a stroke for Legend table's border
- **Fill**: a fill color for the Legend
- **Margin**: the margin around the Legend
- **Position**: The position at which the Legend appears

CaptionOptions: sets a caption to show on the widget. This includes the following settings:
- **Text**: sets the text for the caption
- **Font**: sets the font for the caption
- **Position**: sets the position for the caption to Top (cpTop) or Bottom (cpBottom) or cpNone to hide the caption even if the text is set.

When you set a caption, the size of the widget is reduced to make space for the caption.

Code Snippets

To set a caption and a position for Legend:
```
TMSFNCWidgetMultiProgress1.CaptionOptions.Text :=
                        'Sales of Beverages (Coca-Cola)';
 TMSFNCWidgetMultiProgress1.Legend.Position := lpTopRight;
 TMSFNCWidgetMultiProgress1.CaptionOptions.Position := cpTop;
```

To add circle items by code and initialize them:
```
 TMSFNCWidgetMultiProgress1.CircleItems.Clear;
 AnItem := TMSFNCWidgetMultiProgress1.CircleItems.Add;
 AnItem.Caption := 'North';
 AnItem.Value := 0;
 AnItem := TMSFNCWidgetMultiProgress1.CircleItems.Add;
 AnItem.Caption := 'South';
 AnItem.Value := 0
 AnItem := TMSFNCWidgetMultiProgress1.CircleItems.Add;
 AnItem.Caption := 'East';
 AnItem.Value := 0;
 AnItem := TMSFNCWidgetMultiProgress1.CircleItems.Add;
 AnItem.Caption := 'West';
 AnItem.Value := 0;
```

Then to set the value for South:
```
TMSFNCWidgetMultiProgress1.CircleItems.Items[1].Value := 24;
```

TTMSFNCWidgetMarqueeProgress

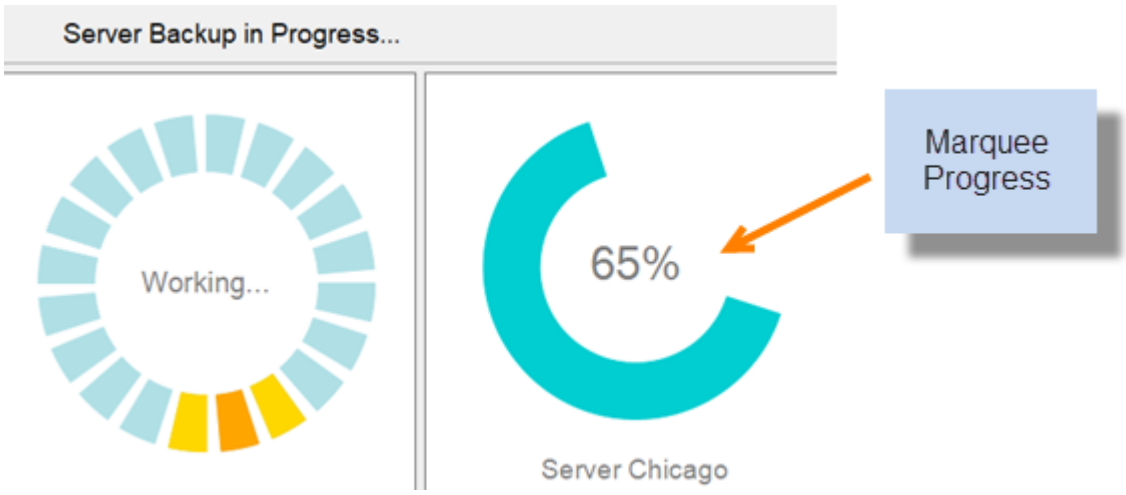
Introduction

TTMSFNCWidgetMarqueeProgress displays the progress of a process graphically in the form of a circle. In addition, this circle revolves like a marquee to indicate a running process. To use this control, just drop it on a form and use the Value property to show the progress. To make it revolve, you need to set its MarqueeAnimation property active.

Features

- Revolving progress circle Animation in addition to displaying the circular progress
- Properties to customize the font and format of the displayed value
- Option to display a Caption text at the top or bottom
- Properties to customize the fill colors and thickness of the finished and unfinished portion of the revolving progress circle

Demo



A Demo is included in the source in a subfolder “Demo\WidgetMarqueeProgress.” In this Demo, two types of Widgets are demonstrated as shown above. The one on the right is the TTMSFNCWidgetMarqueeProgress control. The demo runs by itself showing an increasing progress value till it finishes at 100%.

Properties

How to overview:

- To show a progress value, set it to the Value property. Control the display of the value text by using the properties ValueFont and ValueFormat.
- Make it revolve by activating the property MarqueeAnimation.
- Adjust the thickness of the revolving arc and the color fill of the finished and unfinished portions of the progress circle by the property group CircleOptions.
- Show a Caption by the CaptionOptions property group. The caption can be placed at the top or at the bottom of the control.

Value: sets the value of the progress in percent and displays a circular progress arc accordingly.

ValueFormat: sets the format string to display the value text at the center of the circle. The default format "%g%%" displays the value with a percent symbol. Note that 2 percent symbols at the end of this default format display a single percent symbol.

ValueFont: sets the font to display the value text at the center of the circle.

MarqueeAnimation: set this Active to let the circle revolve. To change the speed of revolution, try changing the Interval and Step properties in MarqueeAnimation.

CircleOptions: includes properties to control the display of the progress circle.

The settings for the above include:

- **Thickness**: sets the thickness of the circle, for both finished and unfinished parts.
- **Fill**: sets the color of the Finished Part of the Circle.
- **UnfinishedFill**: sets the color of the Unfinished Part of the Circle. By default, it is not used in this widget.
- **Margin**: sets the space to leave on all the sides of the widget.

CaptionOptions: sets a caption to show on the widget. This includes the following settings:

- **Text**: sets the text for the caption
- **Font**: sets the font for the caption
- **Position**: sets the position for the caption to Top (cpTop) or Bottom (cpBottom) or cpNone to hide the caption even if the text is set.

Code Snippets

To set a progress value to show:

```
TMSFNCWidgetMarqueeProgress1.Value := 45;
```

To set the caption:

```
TMSFNCWidgetMarqueeProgress1.CaptionOptions.Text := 'Server Chicago';
```

To start the animation:

```
TMSFNCWidgetMarqueeProgress1.MarqueeAnimation.Active := true;
```

TTMSFNCWidgetMarqueeContinuousProgress

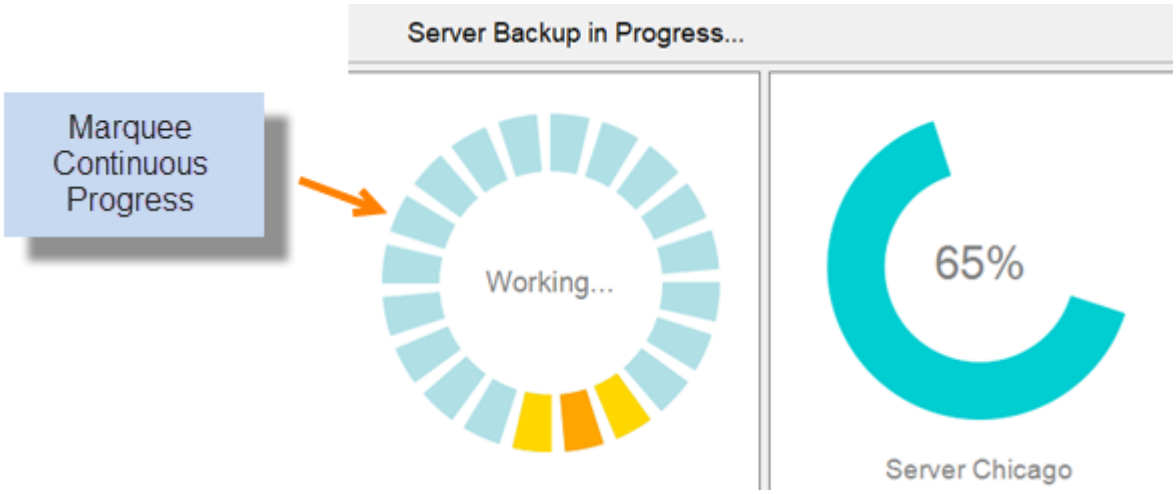
Introduction

TTMSFNCWidgetMarqueeContinousProgress displays a revolving animation to indicate that a process is going on. The animation consists of 3 highlighted sections that revolve around the circle as shown in the screenshot above. This widget does not show any value of the progress. To make it revolve, you need to set its MarqueeAnimation property active.

Features

- Eye-catching animation to show the status text of a running process with a set of revolving highlighted sections
- Properties to customize the number of sections, thickness and gap.
- Properties to customize the all the colors in the sections
- Option to display a Caption text at the top or bottom

Demo



A Demo is included in the source in a subfolder "Demo\WidgetMarqueeProgress." In this Demo, two types of Widgets are demonstrated as shown above. The one on the left is the WidgetMarquee Continuous Progress control. The demo runs by itself showing a working and finished state of the process via this widget's animation.

Properties

How to overview:

- Make it revolve by activating the property MarqueeAnimation.
- Set the CenterText to show a status text.
- Set the number of sections in the MarqueeAnimation property group. This property group also lets you set the colors of the highlighted sections that revolve.
- Adjust the colors of the regular non-highlighted sections by the property group CircleOptions which is similar to the earlier progress controls.
- Show a Caption by the CaptionOptions property group. The caption can be placed at the top or at the bottom of the control.

MarqueeAnimation: includes the following properties to control the animation:

- Active: sets the animation active that revolves around the circle.
- SectionCount: sets the number of sections to display in the circle.
- SectionGapAngle: sets the gap between the sections in terms of an angular difference.
- Fill: sets the color of the main animated, highlighted section.
- SideFill: sets the color of the 2 adjacent sections surrounding the main animated section.

  For example, in the above screenshot, Fill and SIdeFill colors are gcOrange and gcGold respectively. So the main Orange section is surrounded by 2 Gold sections. All these 3 sections revolve around the circle.

CircleOptions: includes properties to control the display of all the section arcs.

The settings for the above include:

- **Thickness**: sets the thickness of the circle that becomes the thickness of a section arc.
- **Fill**: sets the color of a regular non-highlighted section.
- **UnfinishedFill**: sets the color of the gap between the sections. By default, it is not used in this widget.
- **Margin**: sets the space to leave on all the sides of the widget.

CaptionOptions: sets a caption to show on the widget. This includes the following settings:

- **Text**: sets the text for the caption
- **Font**: sets the font for the caption
- **Position**: sets the position for the caption to Top (cpTop) or Bottom (cpBottom) or cpNone to hide the caption even if the text is set.

When you set a caption, the size of the widget is reduced to make space for the caption.

Code Snippets

To set a center text to show:

```
TMSFNCWidgetMarqueeContinuousProgress1.CenterText := 'Working...';
```

Or, later:

```
TMSFNCWidgetMarqueeContinuousProgress1.CenterText := 'Finished';
```

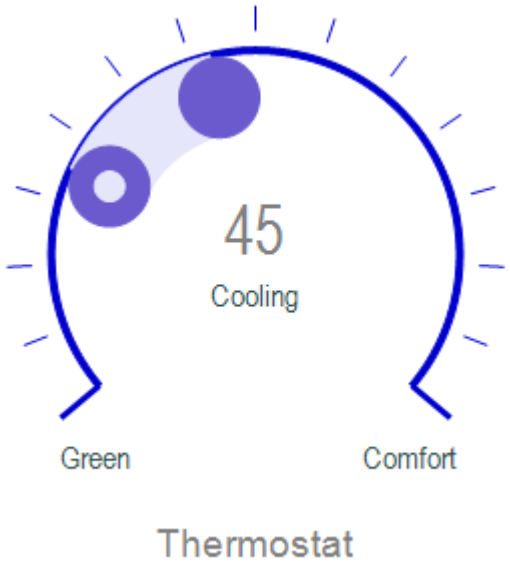
To start the animation:

```
TMSFNCWidgetMarqueeContinuousProgress1.MarqueeAnimation.Active := true;
```

To stop the animation:

```
TMSFNCWidgetMarqueeContinuousProgress1.MarqueeAnimation.Active := false;
```

TTMSFNCWidgetSetPoint



Introduction

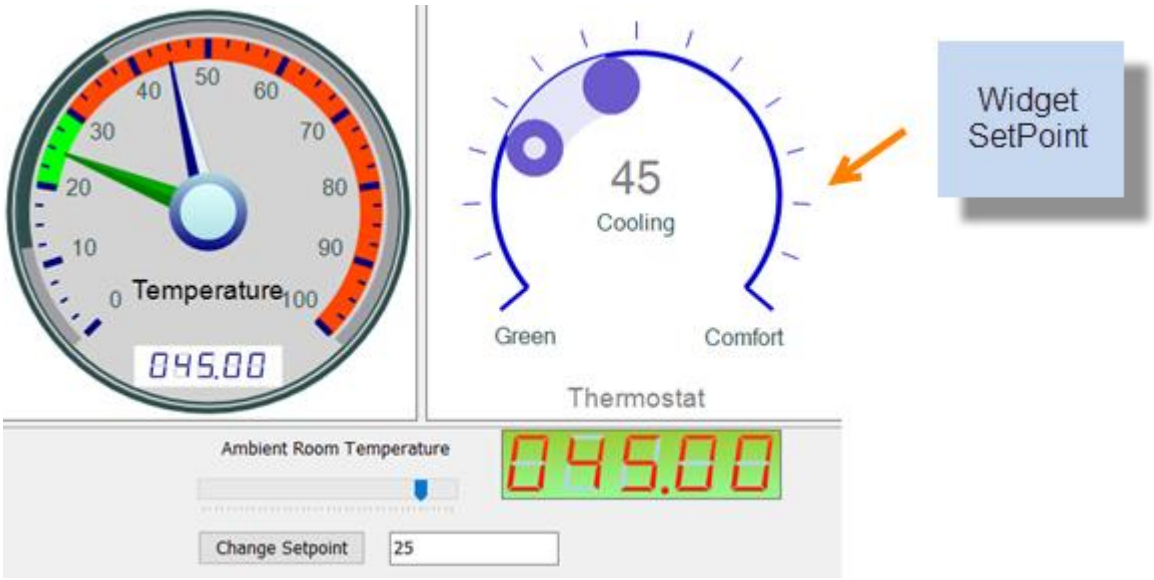
TTMSFNCWidgetSetPoint displays the Value of a parameter and a SetPoint (desired) value of the same parameter on a circular scale. The idea is to highlight the difference between the current Value and the SetPoint which is the desired value. At the same time, the application can use its own logic to display the center text and a starting and ending text based on the difference above.

For example, in the above example, the SetPoint Widget represents the status of a Thermostat where the parameter being displayed is the Temperature. The SetPoint is set at value 25 that is indicated by the double dot. The current temperature is 45, represented by the solid dot. The shaded area between these two is the difference that is shaded for a quick visual representation. At the same time, the status is indicated that the Thermostat is cooling because the current temperature is higher than the desired SetPoint.

Features

- Intuitive display of SetPoint Indicator and Value Indicator with a Strip joining them that is shaded to quickly see how far the Value is from SetPoint
- Properties to customize the colors of Value and SetPoint Indicators and the Strip
- Properties to customize the look of the circular track and tick marks
- Properties to customize the starting, ending and center text showing the status
- Option to display a Caption text at the top or bottom

Demo



A Demo is included in the source in a subfolder "Demo\WidgetsSetPointAndLcdLabel." In this Demo, the SetPoint Widget shows the status of a Thermostat. SetPoint value is set at 25. That is also indicated by the green needle of the Gauge widget on the left. In the screenshot, the current temperature is 45 and hence the Thermostat is cooling towards the SetPoint temperature. The strip between the SetPoint and the current Value gives a visual indication of the difference. The Demo allows changing the temperature by a track bar to see how it affects the Thermostat.

Properties

How to overview:

- Set desired value in SetPoint property.
- Set current value in Value property.
- Control the colors of Value and SetPoint dots by ValueIndicator and SetPointIndicator property groups respectively.
- Control the color of the difference strip from Value to SetPoint by SetPointStripFill property.
- Control the look of the circular track and tick marks with CircleOptions and Tickmarks property groups.
- Set the starting, ending and center text by the Tickmarks property group.
- Show a Caption by the CaptionOptions property group. The caption can be placed at the top or at the bottom of the control.

Value: sets the value that determines the position of the ValueDot according to ValueOptions below.

ValueOptions: includes the following settings:

- **Format**: sets the format string to display the value text at the center of the circle.
- **Font**: sets the font to display the value text at the center of the circle.
- **Min, Max**: These properties set the range of values to display according to which the positions of Value and SetPoint dots are determined.

ValueIndicator: sets the display of the Value Dot. This includes the following properties: properties

- **Kind**: sets the type of indicator which is piDot by default. If you select piBItmap, it allows you to display a custom bitmap assigned via the property BitmapName described below.
- **Size**: sets the diameter of the Value Dot.
- **Fill**: sets the color of the Value Dot.
- **BitmapName**: sets the name of a bitmap in a TMS BitmapContainer assigned to the WidgetProgress control. This bitmap is displayed as the Value Indicator if you set the kind above to piBitmap.

SetPointIndicator: sets the display of the SetPoint Dot. This uses similar properties described above for ValueIndicator. In addition, it uses the following properties to display the SetPoint Dot with an inner dot to distinguish it from the Value Dot.

- **InnerDotSize**: sets the diameter of the Inner Dot inside the SetPoint Dot.
- **InnerDotFill**: sets the color of the Inner Dot inside the SetPoint Dot.

CircleOptions: includes properties to control the display of the circular track. These are the same properties as described in WidgetProgress earlier. But WidgetSetPoint uses only a few of them as indicated below because it does not need to display a progress arc. You can ignore rest of the properties.

- **Fill**: sets the color of the circular track.
- **Thickness**: sets the thickness or width of the circular track.
- **Margin**: sets the space to leave on all the sides of the widget.
- **StartAngle**: sets the angle in degrees of the starting point of the circular track. The default value is angle 230.
- **EndAngle**: sets the angle in degrees of the ending point of the circular track. The default value is angle 130.

Tickmarks: includes properties to display tick marks around the circular track on the outside.

- **Count**: when set to a Non Zero value, shows that many tick marks around the progress circle. Default value is 15.
- **Size**: sets the length of the tick marks.
- **Stroke**: sets the stroke to draw tickmarks
- **Gap**: sets the gap between the circle and the start of a tick mark.
- **CenterText**: sets the text to show at the center of the tick mark circle below the Value Text at the center. For example, the screenshot above shows it as 'Cooling.'
- **StartEndMarkSize**: sets a different size (length) of the starting and ending tick mark.
- **StartEndMarkWidth**: sets a different width of the starting and ending tick mark.
  StartText: sets a text label to show near the starting mark. For example, the screenshot above shows it as 'Green.'
- **EndText**: sets a text label to show near the ending mark. For example, the screenshot above shows it as 'Comfort.'

SetPointStripFill: sets the color of the shaded difference strip joining the SetPoint and the Value Dots.

CaptionOptions: sets a caption to show on the widget. This includes the following settings:

- **Text**: sets the text for the caption
- **Font**: sets the font for the caption
- **Position**: sets the position for the caption to Top (cpTop) or Bottom (cpBottom) or cpNone to hide the caption even if the text is set.

When you set a caption, the size of the widget is reduced to make space for the caption.

Code Snippets

To set a SetPoint and a Value to show:

```
TMSFNCWidgetSetPoint1.SetPoint := 25;
TMSFNCWidgetSetPoint1.Value := 45;
```

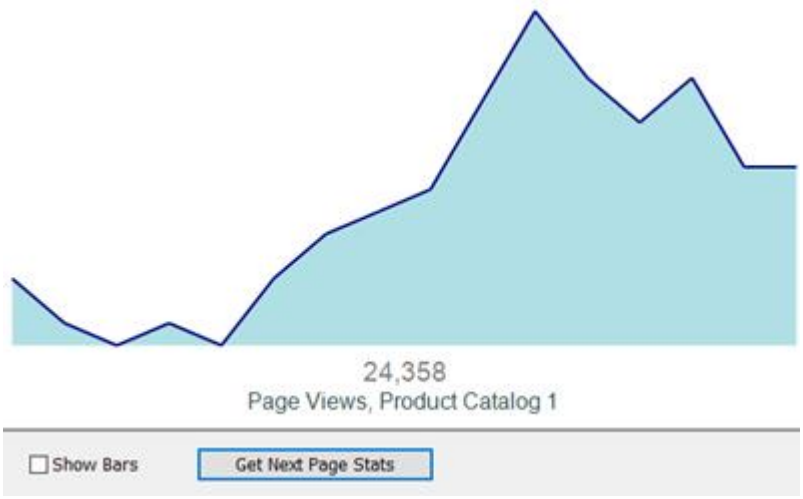
To set the caption:

```
TMSFNCWidgetSetPoint1.CaptionOptions.Text := 'Thermostat';
```

To change Center Text:

```
TMSFNCWidgetSetPoint1.Tickmarks.CenterText := 'Heating'
```

TTMSFNCWidgetTrendIndicator



Introduction

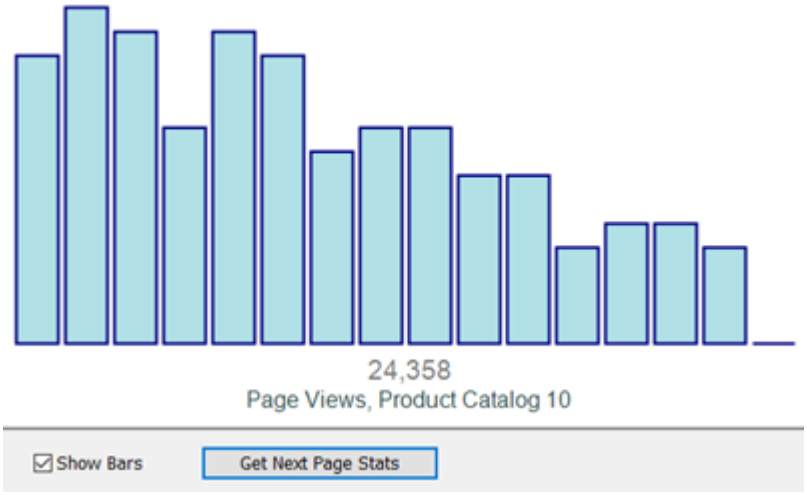
TTMSFNCWidgetTrendIndicator displays a graph of Values and can accommodate a varying number of values, plotting them evenly distributed along the width. Hence, it is very convenient to display a trend.

Features

- Flexible, can accommodate a varying count of values, automatically fitting them to the width
- Ability to add the values directly in the Form Designer and in code
- Option to display as a bar graph with a property to control the gap
- Option to display a summary Value at the bottom along with a description text
- Properties to customize the color and the stroke used to draw the graph or bars

Demo

A Demo is included in the source in a subfolder "Demo\WidgetTrendIndicator." A screenshot is shown at the top above. The Demo generates random values to draw the graph. With each click of the button "Get Next Page Stats," the values are changed randomly and the graph changes. Switch ON the "Show Bars" option to show a bar graph as in the screenshot.



Properties

How to overview:

- Add Values to draw the graph in the Designer or in code. The graph plots them in the available width automatically.
- Optionally display the graph as a bar graph where you control the gap between the bars.
- Display a summary Value at the bottom along with a Description Text.
- Customize the color used to draw the graph line and the fill color for the graph separately.

.
Values: contains the value items to plot the graph. Each item includes a Value property that you need to set. You can click the … for the property in the Object Inspector to manage the values, to add, reorder or delete values. You can also expand the property in the Structure window of Delphi IDE to inspect the values.

GraphType: sets the type of graph to show—**gtLine** or **gtBar**. The default value is gtLine.

GraphColor: sets the stroke for the graph line or the edges of the bars in case of bar graph.

GraphFill: sets the fill for the graph or the bars in case of bar graph.
Value: sets the summary Value shown at the bottom. Application uses its own criteria to set this summary value.

ValueFormat: sets the format string to display the summary Value text at the bottom.

ValueFont: sets the font to display the summary value text at the bottom.

DescriptionText: sets the Description Text shown below the summary value at the bottom.

DescriptionFont: sets the font to display the Description Text.

Code Snippets

To fill Values:

```
var
  v: TTMSFNCWidgetTrendValue;
begin
  ….
  TMSFNCWidgetTrendIndicator1.Values.BeginUpdate;
  try
    TMSFNCWidgetTrendIndicator1.Values.Clear;
    for I := 0 to 15 do
    begin
      v := TMSFNCWidgetTrendIndicator1.Values.Add;
      v.Value := ..avalue from application..
    end;
  finally
    TMSFNCWidgetTrendIndicator1.Values.EndUpdate;
  end;
  …
```

To set description text and a summary value at the bottom:

```
  TMSFNCWidgetTrendIndicator1.DescriptionText := 'Page Views';
  TMSFNCWidgetTrendIndicator1.Value := 24358;
```

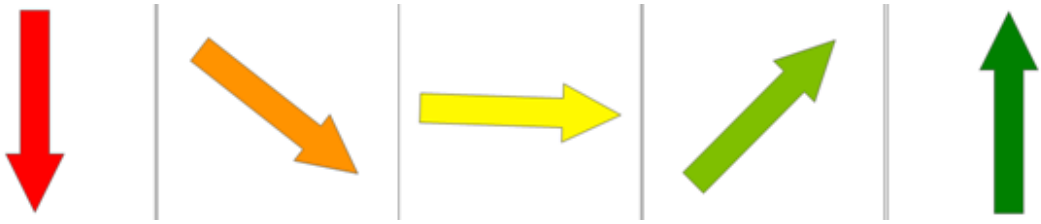
To change the type of the graph:

```
TMSFNCWidgetTrendIndicator1.GraphType := gtBar;
```

Or,

```
TMSFNCWidgetTrendIndicator1.GraphType := gtLine;
```

TTMSFNCWidgetArrow



Introduction

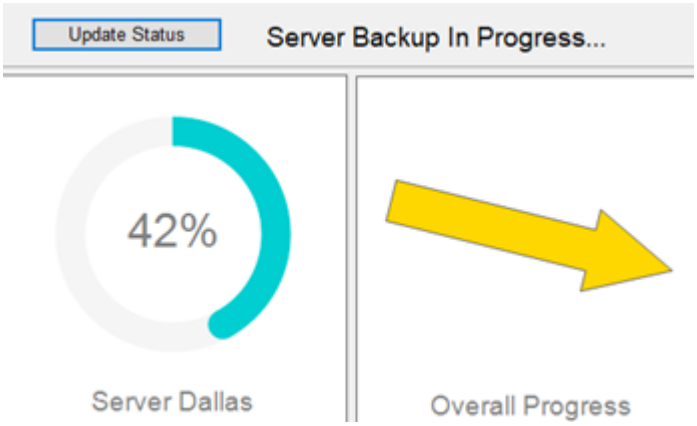
TTMSFNCWidgetArrow displays an arrow that changes color depending on its angle set in the **Value** property. The Value (angle) can be set to minimum -90 degrees and maximum +90 degrees. The minimum value corresponds to a Pointing Down arrow and maximum value corresponds to a Pointing Up arrow. As the Value (angle) changes from -90 to 0 to +90, the color of the arrow changes from Red to Yellow to Green. This is shown in the screenshots at various Values as the arrow rotates and changes color. This can be a useful visual indication of a progress from Red (starting) to Green (Finished).

Features

- Impressive display of status of a process with an arrow that goes from Down to Up direction with a color that changes continuously with the changing direction of the arrow
- Option to display either the angle or a status text by the side of the arrow
- Properties to customize the color of the arrow at the -90, 0 and +90 angle positions. The arrow automatically uses intermediate colors at intervening angle positions.
- Properties to customize the shape of the arrow by the ArrowOptions property group.
- Option to display a Caption text at the top or bottom

Demo

A Demo is included in the source in a subfolder “Demo\WidgetArrow.” This Demo uses the WidgetArrow to show the overall progress of a backup operation. With each click of the button “Update Status,” the backup progresses, and the arrow goes from the Down to Right, then to Up position, changing its color from Red to Yellow to Green.



Properties

How to overview:

- Set the Value (angle) in the range -90 to +90 to show the arrow at that angle with an intermediate color between red and green.
- The Value is displayed by the side of the arrow or can be switched off by making ValueFormat property empty.
- ValueFormat can be set to a fixed text to display the text by the side of the arrow.
- Customize the color of the arrow at the -90, 0 and +90 values by the ArrowOptions property group.
- Customize the shape of the arrow by the ArrowOptions property group.
- Show a Caption by the CaptionOptions property group. The caption can be placed at the top or at the bottom of the control.

Value: sets the Value that is an angle in the range -90 to +90.

ValueFont: sets the font to display the value.

ValueText: sets the text to display by the side of the arrow. By default is a Format '%g' to display the Value. But this can be a fixed text too that shows a changing status based on the requirements of the application. It can even be an empty string so that nothing is displayed.

ArrowOptions: include the properties to control the color and shape of the arrow.

- **ColorFrom**: sets the color of the arrow at Value -90 (Down)
- **ColorAtZero**: sets the color of the arrow at Value 0 (Right)
- **ColorTo**: sets the color of the arrow at Value +90 (Up)
- **Border**: sets the stroke for the border of the arrow
- **Width**: sets the width of the stem of the arrow
- **HeadLength**: sets the length of the head (along the direction of the arrow)
- **HeadWidth**: sets the width of the head where it starts

CaptionOptions: sets a caption to show on the widget. This includes the following settings:

- **Text**: sets the text for the caption
- **Font**: sets the font for the caption
- **Position**: sets the position for the caption to Top (cpTop) or Bottom (cpBottom) or cpNone to hide the caption even if the text is set.

When you set a caption, the size of the widget is reduced to make space for the caption.

Code Snippets

To set a Value (angle) based on a percent progress value:

```
TMSFNCWidgetArrow1.Value :=
              Round(GetArrowValue(TMSFNCWidgetProgress1.Value));
```

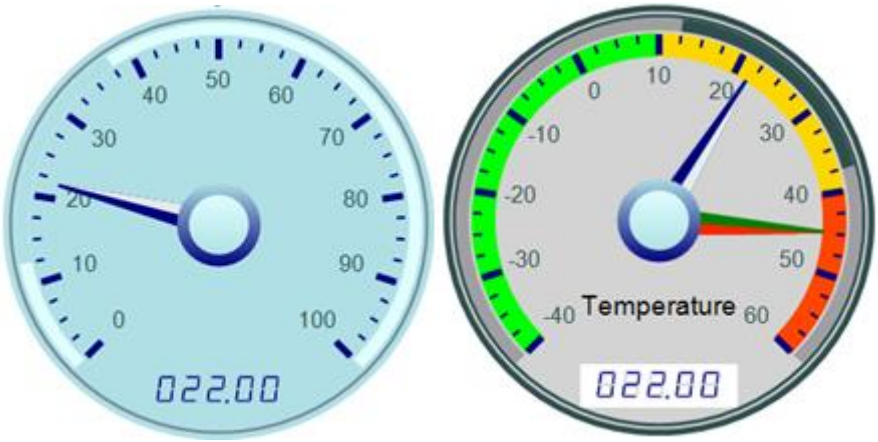
Where the function GetArrowValue is:

```
function GetArrowValue(APosition: Single): Single;
begin
  result := -90 + (APosition / 100.0) * 180;
end;
```

In the Demo, the following code changes the shape of the arrow which is different from the default shape:

```
TMSFNCWidgetArrow1.ArrowOptions.Margin := 10;
TMSFNCWidgetArrow1.ArrowOptions.Width := 30;
TMSFNCWidgetArrow1.ArrowOptions.HeadWidth := 60;
TMSFNCWidgetArrow1.ArrowOptions.HeadLength := 60;
```

TTMSFNCWidgetGauge



Introduction

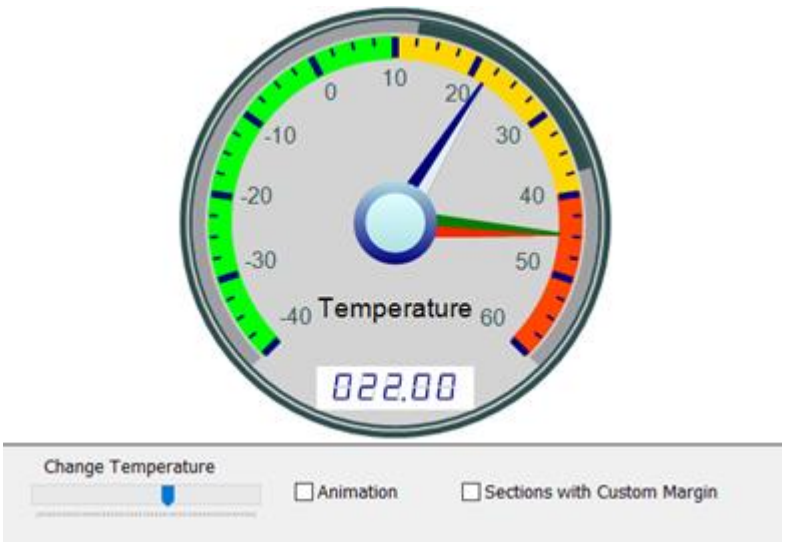
TTMSFNCWidgetGauge is an Instrumentation control to visualize data as a meter with optionally smooth animation.

Features

- Specify any range to show on the meter with **Divisions** and subdivisions.
- Set a **Value** for the main Needle.
- Specify a **Threshold** subrange to be shown on the rim.
- Specify additional ranges with colored **Sections** with custom margins.
- Add **Extra Needles** to show multiple values.
- Enable Animation to move needles in animated motion.
- Get an LCD display of the value at the bottom. Hide this display by the Digit property group.
- Set a Dial text for short description.
- Completely customize the color and size of various items like needles, arcs, marks, sections and more as shown in the screenshots on this page.

Demo

A Demo is included in the source in a subfolder "Demo\WidgetGauge." It shows a customized Gauge that shows Temperature. You can change the Temperature on the track bar to see how it works. Switch ON Animation to see how the change is animated. Also see how the feature to show Sections with custom margin works.



Properties

.
Value: sets the Value to show.

ValueFormat: sets the format to show on the LCD panel.

ValueFont: sets the font to display the values on the divisions.

DialText: sets the short description text on the dial.

DivisionCount: sets the number of divisions to show on the scale.
DivisionColor: sets the color of the division mark.
DivisionWidth: sets the width of the division mark.
SubDivisionCount: sets the number of subdivisions with divisions.
SubDivisionColor: sets the color of subdivision mark.
SubDivisionWidth: sets the width of the subdivision mark.

OuterCircle: sets the color and width of the outermost circle.
OuterRim: sets the color and width of the outer rim which is shown on top of the outer circle.
InnerCircle: sets the color and width of the innermost circle.

Arc: includes the following properties

- **Color**: sets the color of the Arc that is between the outer circle and inner circle.
- **Width**: sets the width of the Arc.
- **StartAngle**: sets the start angle of the Arc.
- **EndAngle**: sets the end angle of the Arc.
- **Threshold**: This is a Subrange that can be shown in the form of a Sub Arc in the above Arc. It includes many sub properties that allow to specify the Threshold Arc.

Digit: includes the properties to control the display of LCD digits.

- **Visible**: Shows or hides the LCD panel
- **Color**: sets the color of the digit
- **BackgroundColor**: sets the background color of the LCD panel. Make it same as InnerCircle to make it look transparent.

Needle: sets color of various parts of the needle.

Animation: when set to true makes the needle move to the next value in animation.

ExtraNeedles: allows to add more needles to the gauge as required either in Designer or by code. Each such extra needle has its own Value property and a set of color properties.

Sections: allows to divide the scale and dial into multiple sections (sub ranges) with their own color and margins. Please see the screenshot of the Demo above to where 3 sections are shown in different color. These sections can be added in Designer or by code.

Code Snippets

In the Demo, the following code was used to set up the Gauge for min, max values and with a different color scheme:

```
TMSFNCWidgetGauge1.Value := 0;
TMSFNCWidgetGauge1.MinimumValue := 0;
TMSFNCWidgetGauge1.MaximumValue := 100;
TMSFNCWidgetGauge1.OuterCircle.Color := gcDarkslategray;
TMSFNCWidgetGauge1.InnerCircle.Color := gcLightgray;
TMSFNCWidgetGauge1.OuterRim.Color := gcLightgray;
TMSFNCWidgetGauge1.Arc.Color := gcMedGray;
TMSFNCWidgetGauge1.Arc.Threshold.Color := gcDarkslategray;
TMSFNCWidgetGauge1.Digit.BackGroundColor := gcWhite;
TMSFNCWidgetGauge1.DialText := 'Temperature';
TMSFNCWidgetGauge1.Font.Color := gcBlack;
```

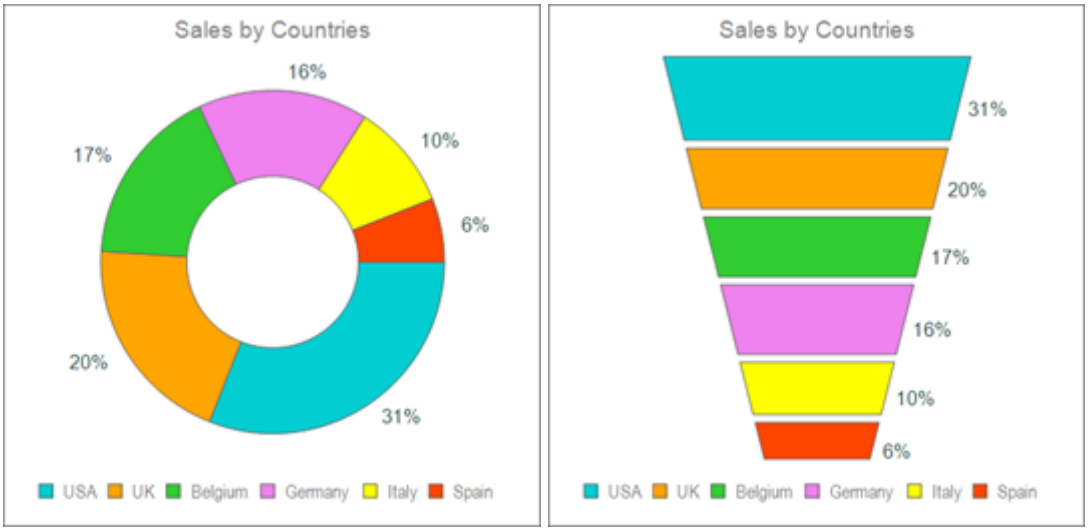
Add Sections to show different subranges:

```
TMSFNCWidgetGauge1.Sections.BeginUpdate;
try
  aSection := TMSFNCWidgetGauge1.Sections.Add;
  aSection.StartValue := 20;
  aSection.EndValue := 30;
  aSection.Color := gcLime;
  aSection.SectionType := stBorder;
  aSection := TMSFNCWidgetGauge1.Sections.Add;
  aSection.StartValue := 30;
  aSection.EndValue := 100;
  aSection.Color := gcOrangeRed;
  aSection.SectionType := stBorder;
finally
  TMSFNCWidgetGauge1.Sections.EndUpdate;
end;
```

Add an Extra Needle and set it to the same value as SetPoint in the Demo:

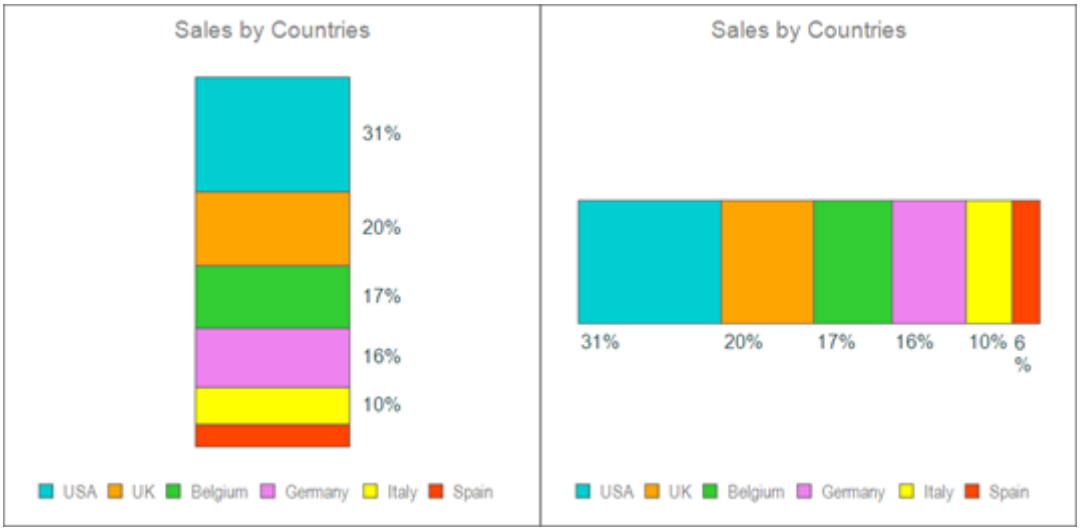
```
aNeedle := TMSFNCWidgetGauge1.ExtraNeedles.Add;
aNeedle.ShineColor := gcDarkGreen;
aNeedle.ShineColorTo := gcLime;
aNeedle.Value := 25;
```

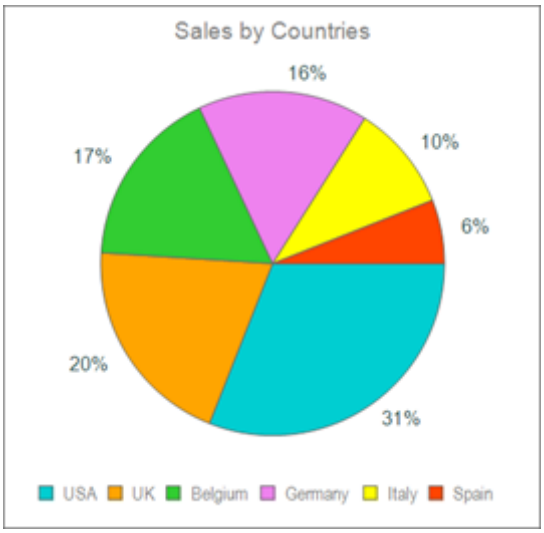
TTMSFNCWidgetDistributionIndicator



Introduction

TTMSFNCWidgetDistributionIndicator displays a distribution graph of Values and can accommodate a varying number of values. Moreover, it can display 5 types of graphs—Donut, Pie, Funnel, Horizontal bar and Vertical bar. The Donut and Funnel bar is already shown above. Here are the pictures of Horizontal bar, Vertical bar and Pie graphs.





Features

- Display the same distribution graph as a Donut, Funnel, Pie, Horizontal bar or Vertical bar
- Flexible, can accommodate a varying count of values, calculating the distribution automatically
- Ability to add the values directly in the Form Designer and in code.
- Option to show the values as Percentages or Absolute values
- Options to show a Header / Footer
- Properties to customize the colors of value items

Demo

A Demo is included in the source in a subfolder "Demo\WidgetDistributionIndicator." A drop down list allows you to switch between the 5 types of graphs. You can also choose to view either Percentage or Absolute values.

Properties

How to overview:

- Add Values to draw the graph in the Designer or in code. The graph calculates the percentages of the distribution automatically.
- Switch between above 5 types of graph views by the property DistributionType.
- Show the values on the graph in Percent or Absolute by the property ValueType.
- Optionally show a Header, Footer on the widget.

Values: contains the value items to plot the graph. Each item includes a Value property that you need to set. You can click the … for the property in the Object Inspector to manage the values, to add, reorder or delete values. You can also expand the property in the Structure window of Delphi IDE to inspect the values.

Each Value Item contains the following properties:

- **Value**: sets the value for the item
- **Text**: sets the text for the item
- **Fill**: sets the color fill for the item's graph
- **Stroke**: sets the stroke for the item's graph

DistributionType: sets the type of graph to show—d**tDonut, dtPie, dtFunnel, dtHorizontalBar, or dtVerticalBar**.

ValueType: sets the value type to show on the graph—**vtAbsolute** or **vtPercentage**. The format of the value display is controlled by the properties **ValueFormatAbsolute** and **ValueFormatPercentage** respectively.

ValueFont: sets the font to display the value text.

Header: sets the header to display on the widget. It includes the sub properties **Text**, **Font** and **Visible** (to show or hide the header).

Footer: sets the footer to display on the widget. It includes the sub properties **Text**, **Font** and **Visible** (to show or hide the footer).

Code Snippets

To fill Values (sample code from Demo):

```
 TMSFNCWidgetDistributionIndicator1.Values[0].Value := 31 * 100;
  TMSFNCWidgetDistributionIndicator1.Values[0].Text := 'USA';
  TMSFNCWidgetDistributionIndicator1.Values[1].Value := 20 * 100;
  TMSFNCWidgetDistributionIndicator1.Values[1].Text := 'UK';
  TMSFNCWidgetDistributionIndicator1.Values[2].Value :=  17 * 100;
  TMSFNCWidgetDistributionIndicator1.Values[2].Text := 'Belgium';
  TMSFNCWidgetDistributionIndicator1.Values[3].Value :=  16 * 100;
  TMSFNCWidgetDistributionIndicator1.Values[3].Text := 'Germany';
  TMSFNCWidgetDistributionIndicator1.Values[4].Value :=  10 * 100;
  TMSFNCWidgetDistributionIndicator1.Values[4].Text := 'Italy';
  TMSFNCWidgetDistributionIndicator1.Values[5].Value :=  6 * 100;
  TMSFNCWidgetDistributionIndicator1.Values[5].Text := 'Spain';
```

To change the graph type as Funnel:

```
  TMSFNCWidgetDistributionIndicator1.DistributionType := dtFunnel;
```

To set a Header:

```
  TMSFNCWidgetDistributionIndicator1.Header.Text := 'Sales by Countries';
```

To change to show Absolute values instead of percentages:

```
  TMSFNCWidgetDistributionIndicator1.ValueType := vtAbsolute;
```

TTMSFNCWidgetLCDLabel

Introduction

TTMSFNCWidgetLCDLabel displays values via 7-segment LCDs with the possibility to add complex gradients and textures.

Features

- Allows background fill, including gradient
- 7-Segment LCDs
- Can display Time too

Demo

There is no separate demo. You can see the demo for WidgetSetPoint to see an example of this widget that displays the temperature.

Properties

.
Fill: sets the background color of the label

Caption: includes the following settings

- **Value**: sets the value to display
- **ValueType**: sets the type of the value. Choices are vtNormal or vtDateTime.
- **Format**: sets the format to display the value in case of vtNormal.
- **TimeFormat**: sets the format to display the value in case of vtDateTime.
- **Fill**: sets the color of the ON segments
- **FillOff**: sets the color of the OFF segments

Code Snippets

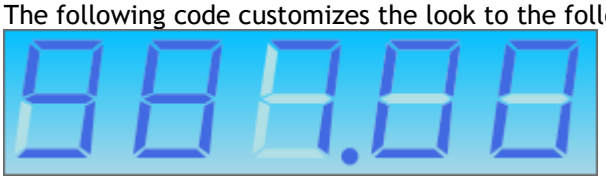
Setting the value:

```
TMSFNCWidgetLCDLabel1.Caption.Value := 419.15;
```

The following code customizes the look to the following:

```
TMSFNCWidgetLCDLabel2.Fill.Color := gcYellowgreen;
TMSFNCWidgetLCDLabel2.Fill.ColorTo := gcPalegreen;
TMSFNCWidgetLCDLabel2.Fill.Kind := gfkGradient;
TMSFNCWidgetLCDLabel2.Caption.FillOff.Color := gcLightBlue;
TMSFNCWidgetLCDLabel2.Caption.FillOff.Kind := gfkSolid;
TMSFNCWidgetLCDLabel2.Caption.Value := -20.45;
```

The following code customizes the look to the following:



```
TMSFNCWidgetLCDLabel3.Fill.Color := gcDeepskyblue;
TMSFNCWidgetLCDLabel3.Fill.ColorTo := gcLightBlue;
TMSFNCWidgetLCDLabel3.Fill.Kind := gfkGradient;
TMSFNCWidgetLCDLabel3.Caption.Fill.Color := gcRoyalblue;
TMSFNCWidgetLCDLabel3.Caption.Fill.Kind := gfkSolid;
TMSFNCWidgetLCDLabel3.Caption.FillOff.Color := gcPowderblue;
TMSFNCWidgetLCDLabel3.Caption.FillOff.Kind := gfkSolid;
TMSFNCWidgetLCDLabel3.Caption.Value := 987;
```