

TreeMap Component

Reference

TTreeMap is a Delphi/VCL component that lays out and renders a list of items using the squarified treemap algorithm.

Copyright 2026

by Rezar Behzad and Ingo Jache

Download: <https://www.fe1.com/treemap/>

Contents

TreeMap Component for Delphi	3
Unit Treemap	6
Class TTreeMap	9
Class TTreeMapContainer	14
Class TTreeMapItem	18
All Units	21
Class Hierarchy	22
All Classes, Interfaces, Objects and Records	23
All Types	24
All Variables	25
All Constants	26
All Functions and Procedures	27
All Identifiers	28

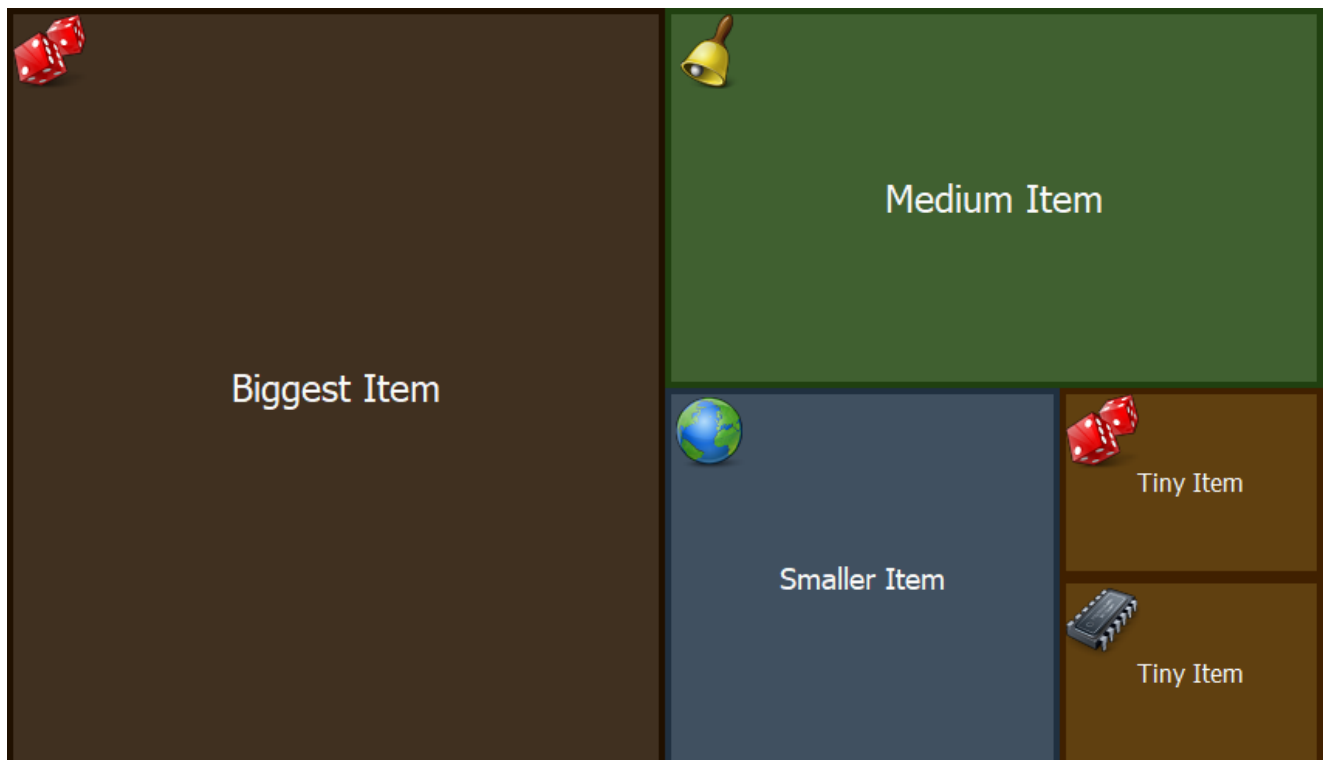
TreeMap Component for Delphi

by Rezar Behzad and Ingo Jache <https://www.fe1.com/treemap/>

Introduction

TTreeMap is a Delphi/VCL component that lays out and renders a list of items using the *squarified treemap* algorithm. The algorithm takes a list of items with associated weights and arranges them on a 2D surface as a set of rectangles: each rectangle's area reflects its item's weight relative to all items, while the layouter keeps the rectangles' aspect ratios as close to square as possible.

The component has decent performance, is customizable, and supports mouse interaction and selection. It is self-contained in a single Delphi unit and requires no third-party libraries.



The layouter is based on this paper: <https://vanwijk.win.tue.nl/stm.pdf>. It explains the idea and the individual steps well, but its pseudocode and equations — in particular the one for computing the squariness — appear to be broken or at least misleading; this component uses a corrected implementation.

Installation

There are two ways to install the component:

1. **Package-based:** double-click the corresponding .dpk file to open the package in Delphi, then right-click the package root and choose *Install*.
2. **No-install:** because the component is a single file, you can also drop it into your project folder or search path and use it from there.

Basic usage

If you installed the package, just drag the TreeMap component onto your form and you are ready to go. If you prefer to create it from code:

```
MyTreeMap := TTreeMap.Create(Self);
MyTreeMap.Parent := Self;
```

There is no item editor in the IDE, but adding items is simple:

```
var
  Item: TTreeMapItem;
begin
  Item := TTreeMapItem.Create(100, 'Item 1');
  MyTreeMap.Items.Add(Item);
end;
```

The constructor of [TTreeMapItem](#) has two mandatory parameters — the item's *Size* (its weight) and its *Caption* (the text shown on the tile) — and three optional ones: *BackgroundColor*, *ForegroundColor* and *Data* (an arbitrary pointer that links the item to your own data). A full example:

```
TTreeMapItem.Create(10, 'Biggest Item', $203040, $F8F8F8, nil);
```

When adding large numbers of items, enclose your Add/Remove calls between [TTreeMap.BeginUpdate](#) and [TTreeMap.EndUpdate](#) to suppress redraws until every item has been added — just as you would with [TListBox](#) and other standard components.

Reacting to input

Assign [TTreeMap.OnClickItem](#) or [TTreeMap.OnDbClickItem](#) to respond to clicks; both handlers receive the clicked [TTreeMapItem](#), the mouse button and the item's state. You can also look up the item under any point directly with [TTreeMap.GetItemAt](#).

Customization

The component has a default painter that styles each tile from a mix of the item's own properties and these component-level properties:

```
MyTreeMap.BorderSize := 4;
MyTreeMap.BackgroundColor := clBlue;
MyTreeMap.SelectionColor := clYellow;
```

Tiles can also show an icon: assign an image list to [TTreeMap.Images](#) and set each item's [TTreeMapItem.ImageIndex](#).

For full control, the TreeMap supports *custom drawing*, much like standard components such as [TListView](#). Assign a handler to the [TTreeMap.OnDrawItem](#) event:

```
MyTreeMap.OnDrawItem := Self.CustomDrawItem;
```

```

procedure TForm1.CustomDrawItem(
  Sender: TObject;           // the TTreeMap instance requesting the draw
  Canvas: TCanvas;           // the canvas to draw the item on
  Item: TTreeMapItem;        // the item to be drawn
  State: TTreeMapItemStates; // the item's state: hover, selected
  var Area: TRect             // the rectangle to draw the item in
);
begin
  // ...
end;

```

Note that Area is a var parameter. If your items are nested (top-level items have children), you may want to draw the parent as a thin title bar and let the children fill the remaining space. To have the children laid out and drawn, leave that remaining space in Area; to suppress them, set Area to Rect(0, 0, 0, 0):

```

procedure TForm1.CustomDrawItem(
  Sender: TObject;
  Canvas: TCanvas;
  Item: TTreeMapItem;
  State: TTreeMapItemStates;
  var Area: TRect
);
const
  TitleBarSize = 48;
var
  PaintArea: TRect;
begin
  if not Item.HasChildren then
    begin
      // no children: use the entire area
      PaintArea := Area;
      Area := Rect(0, 0, 0, 0);
    end
  else
    begin
      // children will be drawn: split the space
      PaintArea := Rect(Area.Left, Area.Top, Area.Right, Area.Bottom - TitleBarSize);
      Area := Rect(Area.Left, Area.Top + TitleBarSize, Area.Right, Area.Bottom);
    end;
  // ... paint the item into PaintArea ...
end;

```

Demos

- The *showcase* demo goes further: mouse interaction, nested treemaps and rendering to bitmaps.

Unit Treemap

Description

Squarified treemap component for the VCL.

This unit implements [TTreeMap](#), a `TGraphicControl` descendant that renders a set of weighted items as a *squarified* treemap: nested rectangles whose areas are proportional to each item's [TTreeMapItem.Size](#) and whose aspect ratios are kept as close to square as possible for readability.

The public model consists of three cooperating types:

- [TTreeMapItem](#) – a single node, carrying a size/weight, caption, colors and an optional list of child nodes.
- [TTreeMapContainer](#) – an owning, size-sorted list of items; the control's [TTreeMap.Items](#) property is such a container.
- [TTreeMap](#) – the visual control that lays the items out, paints them and tracks hovering, selection and clicks.

Custom rendering is possible through [TTreeMap.OnDrawItem](#); when no handler is assigned an internal default painter is used.

Overview

Classes, Interfaces, Objects and Records

NAME	DESCRIPTION
Class TTreeMap	A control that displays weighted items as a squarified treemap.
Class TTreeMapContainer	An owning, size-sorted list of TTreeMapItem instances.
Class TTreeMapItem	A single node in a treemap: a weighted, captioned rectangle.

Functions and Procedures

```
procedure Register;
```

Types

```
TTreeMapItemState = (...);
```

```
TTreeMapItemStates = set of TTreeMapItemState;
```

```
TOnDrawTreeMapItemEvent = procedure( Sender: TObject; Canvas: TCanvas; Item: TTreeMapItem;  
State: TTreeMapItemStates; var Area: TRect) of object;
```

```
TOnTreeMapItemClickEvent = procedure( Sender: TObject; Item: TTreeMapItem; Button: TMouseButton;  
State: TTreeMapItemStates ) of object;
```

Description

Functions and Procedures

procedure **Register**;

Registers [TTreeMap](#) on the "Fasko" component palette page.

Types

TTreeMapItemState = (...);

State flags describing how a [TTreeMapItem](#) is to be painted. They are passed as a set to draw handlers ([TOnDrawTreeMapItemEvent](#)) and click handlers ([TOnTreeMapItemClickEvent](#)).

Values

- `tmisHighlight`: The mouse pointer is currently hovering over the item.
- `tmisSelected`: The item is selected (see [TTreeMap.SetSelected](#)).
- `tmisLast`: Internal: marks the last item included in the current layout.

TTreeMapItemStates = set of [TTreeMapItemState](#);

A set of [TTreeMapItemState](#) flags describing an item's paint state.

TOnDrawTreeMapItemEvent = procedure(Sender: TObject; Canvas: TCanvas; Item: [TTreeMapItem](#); State: [TTreeMapItemStates](#); var Area: TRect) of object;

Custom-draw event for a single treemap item.

Assign a handler to [TTreeMap.OnDrawItem](#) to take over painting of items.

Parameters

Sender

The [TTreeMap](#) control requesting the draw.

Canvas

The canvas to paint onto.

Item

The item to be drawn.

State

The item's current [paint state](#).

Area

The rectangle the item must be drawn into. This is a var parameter: a handler may shrink it – for example to a small title bar – and return the remaining area, into which the item's children are then laid out and drawn.

```
TOnTreeMapItemClickEvent = procedure( Sender: TObject; Item: TTreeMapItem; Button: TMouseButton;  
State: TTreeMapItemStates ) of object;
```

Click and double-click event for a treemap item. Used by both [TTreeMap.OnClickItem](#) and [TTreeMap.OnDblClickItem](#).

Parameters

Sender

The [TTreeMap](#) control that raised the event.

Item

The item under the mouse pointer.

Button

The mouse button involved.

State

The item's current [paint state](#).

Author

- Rezar Behzad & Ingo Jache

Created

26.07.2026

Last Modified

26.07.2026

Generated by [PasDoc 1.0.4](#).

Class TTreeMap

Unit

Treemap

Declaration

```
type TTreeMap = class(TGraphicControl)
```

Description

A control that displays weighted items as a squarified treemap.

Populate the control through its [TTreeMap.Items](#) container, then let the control lay the items out and paint them. Items that have children are drawn as nested treemaps. Hovering and selection are tracked automatically and reported through [TTreeMap.OnClickItem](#) and [TTreeMap.OnDbClickItem](#); painting can be customized through [TTreeMap.OnDrawItem](#).

Wrap bulk changes to [TTreeMap.Items](#) in [TTreeMap.BeginUpdate](#)/[TTreeMap.EndUpdate](#) to avoid intermediate repaints.

Hierarchy

```
TGraphicControl
  TTreeMap
```

Overview

Methods

Public	constructor Create (aowner: TComponent); override;
Public	destructor Destroy ; override;
Public	procedure CalculateTreemap (aitems: TTreeMapContainer ; arect: TRect);
Public	procedure DrawTreemap (acanvas: TCanvas; arect: TRect; aitems: TTreeMapContainer);
Public	function GetItemAt (x,y: integer): TTreeMapItem ;
Public	procedure BeginUpdate ;
Public	procedure EndUpdate ;
Public	procedure ClearSelection ;
Public	procedure SetSelected (const Value: TTreeMapItem ; selected: boolean);
Public	procedure Repaint ; override;

Properties

Public	property Items : TTreeMapContainer read fItems;
Published	property MaxItems : integer read fMaxItems write SetMaxItems default 200;

Published	property MinArea : integer read fMinArea write SetMinArea default 64*64;
Published	property BackgroundColor : TColor read fBackgroundColor write fBackgroundColor default clBtnFace;
Published	property SelectionColor : TColor read fSelectionColor write SetSelectionColor default \$FFFFFF;
Published	property BorderSize : integer read fBorderSize write SetBorderSize default 1;
Published	property Font : TFont read fFont write SetFont;
Published	property Images : TImageList read fImages write fImages;
Published	property OnDrawItem : TOnDrawTreeMapItemEvent read fOnDrawItem write fOnDrawItem;
Published	property OnClickItem : TOnTreeMapItemClickEvent read fOnClickItem write fOnClickItem;
Published	property OnDbClickItem : TOnTreeMapItemClickEvent read fOnDbClickItem write fOnDbClickItem;

Description

Methods

Public	constructor Create (aowner: TComponent); override;
Creates the control, its default painter and an empty item container.	

Public	destructor Destroy ; override;
Destroys the control and frees the items it owns.	

Public	procedure CalculateTreemap (aitems: TTreeMapContainer; arect: TRect);
Recalculates the treemap layout for aitems within arect. Called internally when a relayout is required, but exposed so the same layout algorithm can be applied to arbitrary containers - for example to render a treemap onto another canvas together with TTreeMap.DrawTreemap.	
Parameters	
aitems The container whose items are laid out; each item's TTreeMapItem.Rect is updated in place.	
arect The rectangle the items are laid out within.	

Public procedure **DrawTreemap**(acanvas: TCanvas; arect: TRect; aitems: TTreeMapContainer);

Draws aitems as a treemap onto acanvas within arect.

This is the control's main drawing routine, but it can also render a treemap onto any canvas and for containers not managed by the control. The layout is (re)calculated first if required.

Parameters

acanvas

The target canvas.

arect

The rectangle to draw within.

aitems

The container of items to draw.

Public function **GetItemAt**(x,y: integer): TTreeMapItem;

Returns the item at control coordinates (x, y), searching nested children, or Nil if no item covers that point.

Parameters

x

Horizontal position in control coordinates.

y

Vertical position in control coordinates.

Returns

The item at the given position, or Nil.

Public procedure **BeginUpdate**;

Suspends repainting while TTreeMap.Items is being modified. Must be paired with a later TTreeMap.EndUpdate. Useful when adding or removing many items at once.

Public procedure **EndUpdate**;

Resumes repainting after TTreeMap.BeginUpdate, forces a relayout and repaints the control.

Public procedure **ClearSelection**;

Clears the selected state of every item, including nested children, and repaints the control.

Public procedure **SetSelected**(const Value: TTreeMapItem; selected: boolean);

Sets or clears the selected state of a single item, repainting if the state actually changed.

Parameters

Value

The item to update. Ignored when Nil.

selected

True to select the item, False to deselect it.

Public procedure **Repaint**; override;

Repaints the control immediately, re-rendering the offscreen surface.

Properties

Public property **Items**: [TTreeMapContainer](#) read fItems;

The items rendered by the control. The container may hold a flat list or a hierarchy (items with [TTreeMapItem.Children](#)). The control owns and frees the items it contains. Runtime-only and read-only, so it is not published.

Published property **MaxItems**: integer read fMaxItems write SetMaxItems default 200;

The maximum number of items laid out within a container. Items beyond this count are ignored by the layout. Changing it forces a relayout.

Published property **MinArea**: integer read fMinArea write SetMinArea default 64*64;

The smallest rectangle area, in square pixels, that is still laid out. Layout stops subdividing once the available area drops below this value. Changing it forces a relayout.

Published property **BackgroundColor**: TColor read fBackgroundColor write fBackgroundColor default clBtnFace;

Color used to clear the background behind the tiles.

Published property **SelectionColor**: TColor read fSelectionColor write SetSelectionColor default \$FFFFFF;

Border color drawn around a selected tile by the default painter.

Published property **BorderSize**: integer read fBorderSize write SetBorderSize default 1;

Tile border thickness, in pixels, used by the default painter.

Published property **Font**: TFont read fFont write SetFont;

Base font used by the default painter, which derives several sizes from its name.

Published property **Images**: TImageList read fImages write fImages;

Image list whose icons can be shown on tiles via [TTreeMapItem.ImageIndex](#).

Published property **OnDrawItem**: [TOnDrawTreeMapItemEvent](#) read fOnDrawItem write fOnDrawItem;

Handler invoked to paint each item. When unassigned, an internal default painter is used. See [TOnDrawTreeMapItemEvent](#).

Published property **OnClickItem**: [TOnTreeMapItemClickEvent](#) read fOnClickItem write fOnClickItem;

Handler invoked on a single click on an item. See [TOnTreeMapItemClickEvent](#).

Published

property **OnDbClickItem**: [TOnTreeMapItemClickEvent](#) read fOnDbClickItem write fOnDbClickItem;

Handler invoked on a double click on an item. See [TOnTreeMapItemClickEvent](#).

Generated by [PasDoc 1.0.4](#).

Class TTreeMapContainer

Unit

Treemap

Declaration

type TTreeMapContainer = class(TObject)

Description

An owning, size-sorted list of [TTreeMapItem](#) instances.

[TTreeMapContainer.Add](#), [TTreeMapContainer.Remove](#), [TTreeMapContainer.RemoveAt](#) and [TTreeMapContainer.Clear](#) keep the list sorted by [TTreeMapItem.Size](#) in descending order and keep [TTreeMapContainer.TotalSize](#) up to date on every mutation. The container **owns** its items: an item is freed as soon as it is removed, and when the container is cleared or destroyed.

For bulk population use [TTreeMapContainer.FastAdd](#) - which skips the duplicate check and does not preserve order - followed by a single [TTreeMapContainer.Sort](#).

Hierarchy

TObject
TTreeMapContainer

Overview

Methods

Public	constructor Create ;
Public	destructor Destroy ; override;
Public	function Add (aitem: TTreeMapItem): integer;
Public	function Remove (aitem: TTreeMapItem): boolean;
Public	function RemoveAt (index: integer): boolean;
Public	function IndexOf (aitem: TTreeMapItem): integer;
Public	function IndexOfData (adata: pointer): integer;
Public	function UpdateSizeAt (index: integer; newsize: int64): boolean;
Public	function UpdateSize (aitem: TTreeMapItem ; newsize: int64): boolean;
Public	procedure Clear ;
Public	procedure Sort ;
Public	procedure FastAdd (aitem: TTreeMapItem);

Properties

Public	property Count : integer read GetCount;
Public	property Items [index: integer]: TTreeMapItem read GetItem; default;
Public	property TotalSize : int64 read fTotalSize;

Description

Methods

Public	constructor Create ;
--------	-----------------------------

Creates an empty container.

Public	destructor Destroy ; override;
--------	---------------------------------------

Clears and frees all contained items, then destroys the container.

Public	function Add (aitem: TTreeMapItem): integer;
--------	---

Adds a single item, preserving descending size order, and updates TTreeMapContainer.TotalSize.

Parameters

aitem
The item to add; the container takes ownership of it.

Returns
The index at which the item was inserted. If the item is already present, its existing index is returned; returns -1 when aitem is Nil.

Public	function Remove (aitem: TTreeMapItem): boolean;
--------	--

Removes and frees an item, located by reference, and updates TTreeMapContainer.TotalSize.

Parameters

aitem
The item to remove.

Returns
True if the item was found and removed, False otherwise.

Public function **RemoveAt**(index: integer): boolean;

Removes and frees the item at the given index and updates [TTreeMapContainer.TotalSize](#).

Parameters

[index](#)

Zero-based index of the item to remove.

Returns

True if removed, False if index was out of range.

Public function **IndexOf**(aitem: [TTreeMapItem](#)): integer;

Returns the index of an item, located by reference.

Parameters

[aitem](#)

The item to look up.

Returns

The zero-based index, or -1 if not present.

Public function **IndexOfData**(adata: pointer): integer;

Returns the index of the first item whose [TTreeMapItem.Data](#) pointer equals adata.

Parameters

[adata](#)

The data pointer to match.

Returns

The zero-based index, or -1 if no item matches.

Public function **UpdateSizeAt**(index: integer; newsize: int64): boolean;

Changes the size of the item at index, moving it to its new sorted position and updating [TTreeMapContainer.TotalSize](#).

This is the only supported way to change an item's size, so that the list can stay sorted without a full re-sort.

Parameters

[index](#)

Zero-based index of the item to update.

[newsize](#)

The new size value.

Returns

True on success, False if index was out of range.

Public function **UpdateSize**(aitem: TTreeMapItem; newsize: int64): boolean;

Convenience wrapper around [TTreeMapContainer.UpdateSizeAt](#) that first looks the item up by reference.

Parameters

[aitem](#)

The item whose size to change.

[newsize](#)

The new size value.

Returns

True on success, False if the item was not found.

Public procedure **Clear**;

Removes and frees all items and resets [TTreeMapContainer.TotalSize](#) to zero.

Public procedure **Sort**;

Sorts the list by [TTreeMapItem.Size](#) in descending order. Use together with [TTreeMapContainer.FastAdd](#).

Public procedure **FastAdd**(aitem: TTreeMapItem);

Appends an item without checking for duplicates and without preserving sort order, updating [TTreeMapContainer.TotalSize](#). Intended for bulk population; call [TTreeMapContainer.Sort](#) once all items have been added.

Parameters

[aitem](#)

The item to append; the container takes ownership of it.

Properties

Public property **Count**: integer read GetCount;

The number of items currently in the container.

Public property **Items**[index: integer]: TTreeMapItem read GetItem; default;

Indexed, read-only access to the contained items. This is the default property, so `container[i]` is equivalent to `container.Items[i]`.

Public property **TotalSize**: int64 read fTotalSize;

The sum of the [TTreeMapItem.Size](#) values of all contained items.

Class TTreeMapItem

Unit

Treemap

Declaration

```
type TTreeMapItem = class(TObject)
```

Description

A single node in a treemap: a weighted, captioned rectangle.

The most important attribute is [TTreeMapItem.Size](#), the weight from which the item's area in the layout is derived. Items live inside a [TTreeMapContainer](#), which keeps them sorted by size and owns them. An item may itself own a container of child items ([TTreeMapItem.Children](#)), which are drawn as a nested treemap.

Hierarchy

```
TObject
  TTreeMapItem
```

Overview

Methods

Public	constructor Create (asize: int64; acaption: string; afill: TColor = \$203040; atext: TColor = \$f8f8f8; adata: pointer = nil);
Public	destructor Destroy ; override;
Public	function HasChildren : boolean;

Properties

Public	property Size : int64 read fSize;
Public	property Rect : TRect read fRect;
Public	property Caption : string read fCaption write fCaption;
Public	property Info : string read fInfo write fInfo;
Public	property BackgroundColor : TColor read fBackgroundColor write fBackgroundColor;
Public	property ForegroundColor : TColor read fForegroundColor write fForegroundColor;
Public	property Data : pointer read fData write fData;
Public	property Tag : integer read fTag write fTag;
Public	property ImageIndex : integer read fImageIndex write fImageIndex;
Public	property Children : TTreeMapContainer read GetChildren;

Public property **Selected**: boolean read fSelected;

Description

Methods

Public constructor **Create**(asize: int64; acaption: string; afill: TColor = \$203040; atext: TColor = \$f8f8f8; adata: pointer = nil);

Creates an item.

Parameters

asize

The item's size/weight, driving its area in the layout.

acaption

The caption text shown on the tile.

afill

The tile background/fill color.

atext

The caption/text color.

adata

An optional user pointer stored in [TTreeMapItem.Data](#).

Public destructor **Destroy**; override;

Destroys the item and frees its child container, if any.

Public function **HasChildren**: boolean;

Returns True if the item owns at least one child item (that is, when [TTreeMapItem.Children](#) is non-empty).

Properties

Public property **Size**: int64 read fSize;

The item's size/weight. Set at creation; change it through [TTreeMapContainer.UpdateSize](#) so the owning container stays sorted.

Public property **Rect**: TRect read fRect;

The rectangle assigned to the item by the layouter, in the coordinate space of the surface it was laid out on. Read-only; updated during layout.

Public property **Caption**: string read fCaption write fCaption;

The caption text drawn on the tile.

Public property **Info**: string read fInfo write fInfo;

Secondary information text drawn on the tile (for example a formatted size).

Public property **BackgroundColor**: TColor read fBackgroundColor write fBackgroundColor;

The tile fill color.

Public property **ForegroundColor**: TColor read fForegroundColor write fForegroundColor;

The caption/text color.

Public property **Data**: pointer read fData write fData;

An arbitrary user pointer, for linking the item to application data.

Public property **Tag**: integer read fTag write fTag;

An arbitrary user integer, free for application use.

Public property **ImageIndex**: integer read fImageIndex write fImageIndex;

Index into the control's [TTreeMap.Images](#) list of the icon shown on the tile, or -1 for none.

Public property **Children**: [TTreeMapContainer](#) read GetChildren;

Optional child items, drawn as a nested treemap inside this item's tile. Their sizes and [TTreeMapContainer.TotalSize](#) are independent of this item's own [TTreeMapItem.Size](#) and of the [TTreeMapContainer.TotalSize](#) of the container holding this item. Reading this property creates the child container on demand.

Public property **Selected**: boolean read fSelected;

True when the item is selected. Change it through [TTreeMap.SetSelected](#) or [TTreeMap.ClearSelection](#).

Generated by [PasDoc 1.0.4](#).

All Units

NAME	DESCRIPTION
Treemap	Squarified treemap component for the VCL.

Generated by [PasDoc 1.0.4](#).

Class Hierarchy

TGraphicControl

TTreeMap

TObject

TTreeMapContainer

TTreeMapItem

Generated by [PasDoc 1.0.4](#).

All Classes, Interfaces, Objects and Records

NAME	UNIT	DESCRIPTION
TTreeMap	Treemap	A control that displays weighted items as a squarified treemap.
TTreeMapContainer	Treemap	An owning, size-sorted list of TTreeMapItem instances.
TTreeMapItem	Treemap	A single node in a treemap: a weighted, captioned rectangle.

Generated by [PasDoc 1.0.4](#).

All Types

NAME	UNIT	DESCRIPTION
TOnDrawTreeMapItemEvent	Treemap	Custom-draw event for a single treemap item.
TOnTreeMapItemClickEvent	Treemap	Click and double-click event for a treemap item.
TTreeMapItemState	Treemap	State flags describing how a TTreeMapItem is to be painted.
TTreeMapItemStates	Treemap	A set of TTreeMapItemState flags describing an item's paint state.

Generated by [PasDoc 1.0.4](#).

All Variables

The units do not contain any variables.

Generated by [PasDoc 1.0.4](#).

All Constants

The units do not contain any constants.

Generated by [PasDoc 1.0.4](#).

All Functions and Procedures

NAME	UNIT	DESCRIPTION
Register	Treemap	Registers TTreeMap on the "Fasko" component palette page.

Generated by [PasDoc 1.0.4](#).

All Identifiers

NAME	UNIT	DESCRIPTION
Register	Treemap	Registers TTreeMap on the "Fasko" component palette page.
TOnDrawTreeMapItemEvent	Treemap	Custom-draw event for a single treemap item.
TOnTreeMapItemClickEvent	Treemap	Click and double-click event for a treemap item.
TTreeMap	Treemap	A control that displays weighted items as a squarified treemap.
TTreeMapContainer	Treemap	An owning, size-sorted list of TTreeMapItem instances.
TTreeMapItem	Treemap	A single node in a treemap: a weighted, captioned rectangle.
TTreeMapItemState	Treemap	State flags describing how a TTreeMapItem is to be painted.
TTreeMapItemStates	Treemap	A set of TTreeMapItemState flags describing an item's paint state.

Generated by [PasDoc 1.0.4](#).