productivity software building blocks

tmssoftware.com

TMS Advanced Toolbars
& Menus
DEVELOPERS GUIDE

Index

Availability

TMS Advanced ToolBars & Menus is available as VCL components set for Win32 application development.

VCL versions:

TMS Advanced ToolBars & Menus is available for Delphi 5,6,7,2005,2006,2007,2009,2010,XE & C++Builder 5,6,2006,2007,2009,2010,XE.*

*In Delphi 5 & C++Builder 5, Office 2010 ribbons are not supported.

Description

TMS Advanced ToolBars & Menus comprises a VCL component set to create toolbars & menus as found in 3 generations of Microsoft Office and Windows 7. This consists of the classic toolbars & menus of Office 2003, the fluent ribbon UI as found in Office 2007, the Windows 7 scenic ribbon and the Office 2010 style ribbon and application menu. While the components have all built-in styles to emulate the various Office & Windows user interfaces, the colors of the controls can be fully adapted to create custom styles.

Office 2003 toolbar
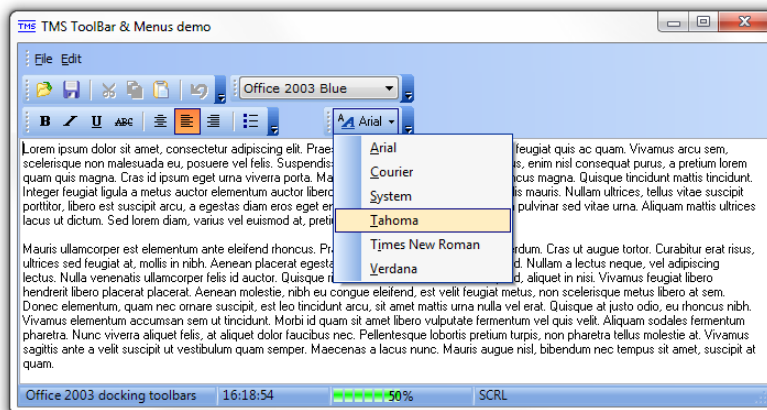
Office 2007 ribbon
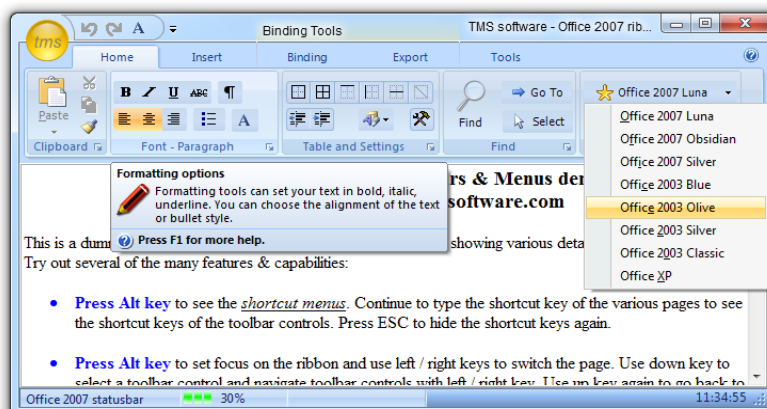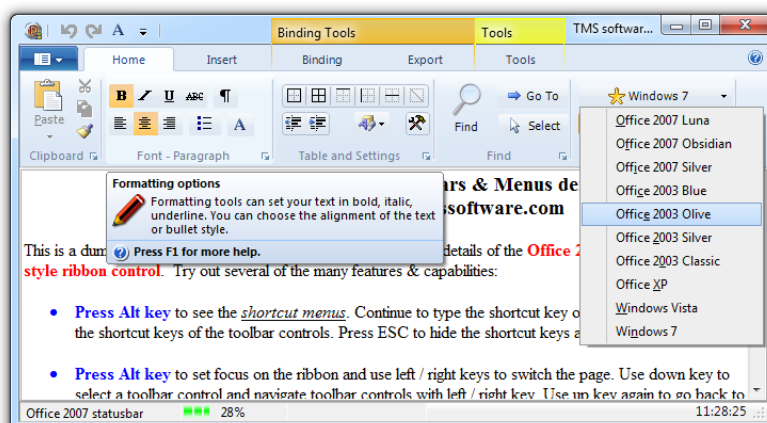
Office 2010 ribbon

Feature overview

TMS Advanced ToolBars & Menus includes following components

- TAdvToolBar: Office 2003 toolbar, Office 2007, Office 2010 & Windows 7 scenic ribbon toolbar
- TAdvToolBarPager: Office 2007, Office 2010, Windows 7 scenic ribbon base container class
- TAdvToolBarButton: Office 2003 toolbar button
- TAdvGlowButton: Office 2007, Office 2010 toolbar button
- TAdvOfficeStatusBar: Office 2003, Office 2007 & Office 2010 style statusbar
- TAdvPreviewMenu: Office 2007 ribbon, Windows 7 scenic ribbon application menu
- TAdvShapeButton: Office 2007 ribbon, Office 2010 ribbon & Windows 7 scenic ribbon application menu button
- TAdvMainMenu: Office 2003 style menu
- TAdvPopupMenu: Office 2003 popup menu
- TAdvStickyPopupMenu: Popup menu that remain visible
- TAdvVerticalPolyList: Office 2010 style application menu part component

Features of TMS Advanced ToolBars & Menus

- Office 2003, Visual Studio style docking toolbars & menus
- Office 2007, Office 2010 ribbon & Windows 7 scenic ribbon
- Licensed to Microsoft Office 2007 UI guidelines
- Automatic theme color adaption on Windows XP
- Allows to drop any control on the toolbar
- Automatic persistence of toolbar positions
- Top, bottom, right & left docking toolbar panels
- Imagelist images, bitmaps and picture support for buttons
- ActionList support
- Comes with Office 2003, Office 2007, Office 2010, Windows Vista, Windows 7 menu & toolbar styles and different other custom color schemes.
- Runtime toolbar configuration
- Terminal style for color reduction on terminals

Licensing

The TMS Advanced ToolBars & Menus are conform with the Microsoft Office 2007 Fluent UI licensing requirements and TMS Advanced ToolBars & Menus are as such licensed by Microsoft. For creating applications using the Office 2007 ribbon, it is required to sign the royalty free license agreement with Microsoft. The license is available for applications on any platform, except for applications that compete directly with the five Office applications that currently have the new UI (Microsoft Word, Excel, PowerPoint, Outlook, and Access). This license limitation does not apply to the Windows 7 scenic ribbon style.

TMS Advanced Menus

TMS Advanced Menus offers replacements for the standard TMainMenu and TPopupMenu. The equivalent components are TAdvMainMenu and TAdvPopupMenu. The TAdvMainMenu, TAdvPopupMenu use the same design time menu editor and the same TMenuItem class. Using TAdvMainMenu and TAdvPopupMenu is as such very similar to using a TMainMenu and TPopupMenu. An additional component TAdvStickyPopupMenu is a separate menu popup menu component that can remain visible while performing selections. TAdvStickyPopupMenu is separately discussed in this chapter.

Key differences between the TMainMenu/TPopupMenu and TAdvMainMenu/TMainMenu are:

- TAdvMainMenu/TAdvPopupMenu have the capability to have a DisabledImages imagelist to set a separate imagelist for disabled images
- TAdvMainMenu/TAdvPopupMenu have a styler component that can be assigned that configures all colors of the menu
- TAdvMainMenu/TAdvPopupMenu can show a sidebar, this is a bar left from the menu that can have extra information
- TAdvMainMenu/TAdvPopupMenu can show separate description text for menu items
- TAdvMainMenu/TAdvPopupMenu can have customizable checkbox or radiobutton glyphs

All visual settings of the menus are offered through the TAdvMenuOfficeStyler. Assign this styler component to TAdvMainMenu.MenuStyler and TAdvPopupMenu.MenuStyler. The specific color settings can be choosen for Office 2003, Office 2007, Office 2010, Windows XP, Windows Vista and Windows 7.

Example of changing the menu styler style via code:

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  case (Sender as TButton).Tag of
  1: AdvMenuOfficeStyler1.Style := osOffice2003Blue;
  2: AdvMenuOfficeStyler1.Style := osOffice2010Blue;
  3: AdvMenuOfficeStyler1.Style := osWindows7;
  end;
end;
```

Properties of the menu styler:

AntiAlias	Controls the anti aliasing technique used to draw menu item text
AutoThemeAdapt	When true, the menu color automatically adapts to the Office 2003 color style selection
Background	Sets a background solid color, gradient color or image for the menu
ButtonAppearance	Controls the appearance of menu buttons on a toolbar from where the menu is invoked (if the menu is used on a toolbar)
Glyphs	Holds the glyphs that are used to draw a checkbox, radiobutton and submenu triangle on the menu
IconBar	Holds the color properties for the iconbar on the menu, including the background color for checkboxes & radiobuttons

MenuBorderColor	Sets the border color of the menu
NotesFont	sets the font of description text for a menu item
RootItem	Holds the various color & font settings for menu root items
SelectedItem	Holds the various color & font settings for a selected menu item
Separator	Sets the color or gradient colors and gradient type for menu separator linesSideBar: holds the color & text properties for an extra sidebar on the left of the menu. The sidebar can have a gradient or image background, an icon and text
Style	Sets one of the predefined styles of the menu
UseSystemFont	Default true to use the system defined menu font. Set to false to use a custom font for the menu

Using TAdvMainMenu / TAdvPopopMenu at runtime

For programmatically inserting and updating menu items, it is required to do this between calling menu.BeginUpdate / menu.EndUpdate. This is a requirement due to an optimization for visual updates when the menu stylers are used. Following sample code snippet shows how to add a new menu item at runtime:

```
var
  mnu: TMenuItem;
begin
  AdvMainMenu1.BeginUpdate;
  mnu := TMenuItem.Create(self);
  mnu.Caption := 'New item';
  AdvMainMenu1.Items[1].Insert(0,mnu);
  AdvMainMenu1.EndUpdate;
end;
```

Using Notes in menu items

TAdvMainMenu and TAdvPopupMenu have the capability to show description text for each menu item. This description text is set by adding additional lines of text to the MenuItem.Caption property. Additional lines are separated with the '\n' character sequence. The description text is displayed when TAdvMainMenu.ShowNotes is set to true. The font for the description text is controlled by the setting TAdvMenuOfficeStyler.NotesFont. Defining a menu item with a description is as such straightforward:

```
var
  mnu: TMenuItem;
begin
  AdvMainMenu1.BeginUpdate;
  mnu := TMenuItem.Create(self);
  mnu.Caption := 'MenuCaption\nLine 1 of the notes text\nLine 2 of the
notes text';
  AdvMainMenu1.Items[1].Insert(0,mnu);
```

```
    AdvMainMenu1.EndUpdate;
end;
```

Note that the height of the menu item will automatically adapt to fit the text of the description.
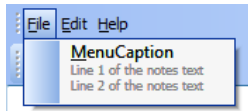
Using the menu sidebar

Via the menu styler, it is possible to enable a sidebar in the menu. This sidebar is displayed left from the icon bar. In the sidebar, optional text or an image can be added. This simple configuration in code shows a sidebar in the menu for which the styler is set to the AdvMenuOfficeStyler with sidebar set visible:

```
begin
  AdvMenuOfficeStyler1.SideBar.Caption := 'File menu';
  AdvMenuOfficeStyler1.SideBar.Font.Name := 'Tahoma';
  AdvMenuOfficeStyler1.SideBar.Font.Color := clWhite;
  AdvMenuOfficeStyler1.SideBar.Background.Color := $0086D5FB;
  AdvMenuOfficeStyler1.SideBar.Background.ColorTo := $000073E6;
  AdvMenuOfficeStyler1.SideBar.Background.GradientDirection :=
gdHorizontal;
  AdvMenuOfficeStyler1.SideBar.Visible := true;
end;
```

Result:

Using custom fonts in menus

By default, the system fonts are used in the menus. This guarantees that the menu font is consistent with all application menus run on the operating system. It is possible to use custom fonts for the menus though. To do this, set the property TAdvMenuOfficeStyler.UseSystemFont to false for the elements of the menu where a custom font should be used and select a custom font with the appropriate Font property.

Using the TAdvMainMenu on a TAdvToolBar

When the TAdvMainMenu is used on a TAdvToolBar, ie. by assigning it to AdvToolBar.Menu, the root items are not really displayed as regular root items directly on a form menu. The root items

become buttons on the TAdvToolBar from where the submenus are displayed. The color settings for the toolbar buttons are set via TAdvMenuOfficeStyler.ButtonAppearance. It is also important that when making changes to root items at runtime to the menu that the toolbar should be instructed to update. This is done by calling AdvToolBar.UpdateMenu. Example:

```
var
  mnu: TMenuItem;
begin
  mnu := TMenuItem.Create(self);
  mnu.Caption := 'Toolbar root item';
  AdvMainMenu1.Items.Add(mnu);
  AdvToolBar1.UpdateMenu;
end;
```
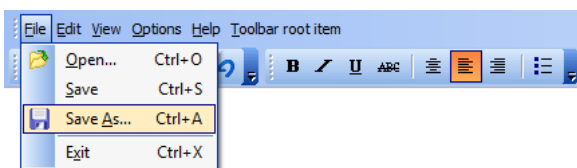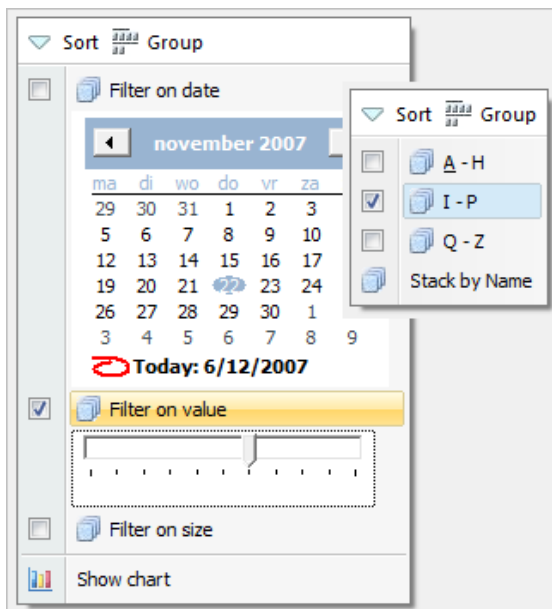

Using the TAdvStickyPopupMenu

In some cases it is not desirable and even annoying that a menu immediately disappears when a selection is made. To address such situations, the component TAdvStickyPopupMenu was created.

The TAdvStickyPopupMenu component was modelled after the new menu that can be found in the Windows Vista File Explorer.

The TAdvStickyPopupMenu consists of a button bar (top area of the menu) and menu items. The styles of the TAdvStickyPopupMenu are identical to the styles of a TAdvPopupMenu or TAdvMainMenu and are controlled by the same styler (TAdvMenuOfficeStyler) This way, the TAdvStickyPopupMenu can be set in Office 2003, Office 2007, Visual Studio .NET, Windows XP or Windows Vista styles.

To show the menu, call TAdvStickyPopupMenu.ShowMenu(x,y: integer); with X,Y being either screen coordinates or form coordinates for the position where the menu should appear. It is controlled by the property TAdvStickyPopupMenu.DisplayRelative drScreen or drForm what type of coordinates is used as X,Y parameters.

The TAdvStickyPopupMenu ButtonBar
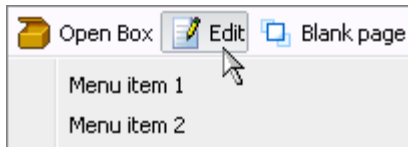
On top of the menu, it is possible to have an optional buttonbar. When TAdvStickyPopupMenu.ShowButtonsBar is true and the TAdvStickyPopupMenu.ButtonBar collection contains buttons, this will be shown. Note that the height of the buttonbar is controlled by the styler with AdvMenuOfficeStyler.SideBar.Size.
Each button in the ButtonBar can have a caption & image from the imagelist set with TAdvStickyPopupMenu.ButtonImages. A click on a ButtonBar button can either automatically hide the menu (like a classical popupmenu) when property TButtonBarItem.HideOnClick = true or the menu remains visible when clicked when property TButtonBarItem.HideOnClick = false. The click on the ButtonBar button can be handled by event TButtonBarItem.OnClick or via TAdvStickyPopupMenu.OnButtonClick.

The TAdvStickyPopupMenu menu items
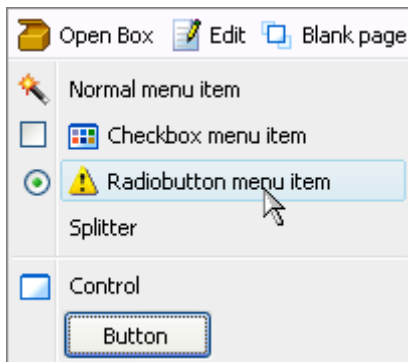
Several types of menu items are possible:

- Normal menu item: this is a menu item that can be enabled or disabled and that has a caption and an imagelist image

- A CheckBox menu item: this is a menu item with a checkbox, optionally an imagelist image and a caption. When AutoCheck = true, the checkbox automatically toggles on a click, otherwise the property TStickyMenuItem.Checked should be used.

- A Radiobutton menu item: this is a menu item with a radiobutton, optionally an imagelist image and a caption

- A splitter menu item: when MenuItem.Style = isSplitter, the item appears as just a splitter line

In addition to these 4 types, all types can be combined with a control (or container control like TPanel with more controls on it) and the control appears under the menu item. To add a control to a menu item, simply drop the control on the form and assign it to MenuItem.Control.

Just like with a ButtonBar button, a click on a menu item can either automatically hide the menu (like a classical popupmenu) when property TStickyMenuItem.HideOnClick = true or the menu remains visible when clicked when property TStickyMenuItem.HideOnClick = false. The click on the menu item can be handled by event TStickyMenuItem.OnClick or via TAdvStickyPopupMenu.OnItemClick.

In addition to this event, the OnCheckClick is triggered when a checkbox item is clicked or OnRadioClick is triggered when a radio menu item is clicked.

OwnerDraw menu items

Finally, it is possible to have custom drawing of the menu item image or the full menu item. To use this, each TStickyMenuItem has two events, OnDrawImage and OnDrawItem.

This code draws a simple cross as menu item image:

```
procedure
TForm1.AdvStickyPopupMenu1MenuItems2DrawImage(Sender: TObject;
 Canvas: TCanvas; ARect: TRect; Selected: Boolean);
begin
 Canvas.MoveTo(ARect.Left, ARect.Top);
 Canvas.LineTo(ARect.Right, ARect.Bottom);
 Canvas.MoveTo(ARect.Right, ARect.Top);
 Canvas.LineTo(ARect.Left, ARect.Bottom);
end;
```

This code draws the menu item text with an italic font:

```
procedure
TForm1.AdvStickyPopupMenu1MenuItems2DrawItem(Sender: TObject;
 Canvas: TCanvas; ARect: TRect; Selected: Boolean);
begin
 Canvas.Font.Style := [fsItalic];
 Canvas.TextOut(ARect.Left, ARect.Top, (Sender as
TStickyMenuItem).Caption);
end;
```

TMS Advanced ToolBar : Office 2003 docking toolbar mode

Getting started using docking toolbars

The TAdvDockPanel and TAdvToolBar together provide the capability to create user interfaces as present in Office 2003. A dockpanel is the container for one or more toolbars and the dockpanel can be positioned on each of the 4 sides of a form. Start by dropping a TAdvDockPanel on the form. By default it is set to the top position (property Align is set to daTop). Change the Align property if the TAdvDockPanel should be positioned on a different side of the toolbar. To add a toolbar to the dockpanel, right-click the dockpanel at design time and from the context menu, choose "Add toolbar". The toolbar has by default a grip on the left side from where the toolbar can be dragged. On the right side of the toolbar is by default a handle from where an option menu can be chosen. Controls can be added to the toolbar in two ways: right-click on the toolbar at design time to choose any of the predefined toolbar controls or drag & drop a component from the component palette on the toolbar. Note that by default, buttons are always left aligned on the toolbar.

The controls that are available from the context menu for a classic Office 2003 style toolbar are:
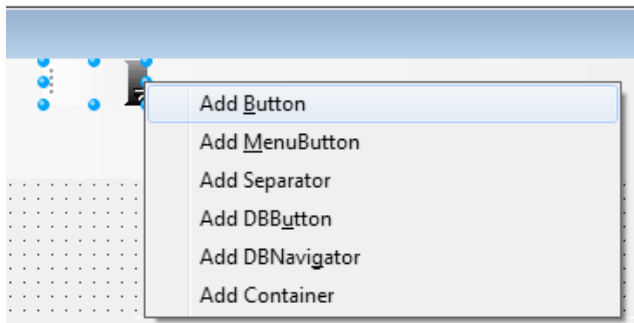
- TAdvToolBarButton : standard buton
- TAdvToolBarMenuButton : button with associated menu
- TAdvToolBarSeparator : separator
- TDBAdvToolBarButton : DB-function button
- TDBAdvToolBarNavigator : DB navigator
- TAdvToolBarContainer : container for other controls

Following code snippet demonstrates how to do the equivalent at runtime in code:

```
var
  adp: TAdvDockPanel;
  atb: TAdvToolBar;
  btn: TAdvToolBarButton;
begin
  adp := TAdvDockPanel.Create(self);
  adp.Parent := self;
  atb := TAdvToolBar.Create(adp);
  atb.Parent := adp;
  btn := TAdvToolBarButton.Create(self);
  btn.Caption := '1';
  btn.ShowCaption := true;
  atb.AddToolBarControl(btn);
  btn := TAdvToolBarButton.Create(self);
  btn.Caption := '2';
```

```
  btn.ShowCaption := true;
  atb.AddToolBarControl(btn);
end;
```

Buttons on the toolbar can work in a couple of different modes:

- Regular button: by default, the button on the toolbar operates as a regular button. It triggers the OnClick event when clicked.
- Radio buttons: All buttons for which the GroupIndex property is set to the same value different from zero will operate as radiobuttons. The checked radiobutton can be found by checking the Down property
- Checkbox button: this button has a down and normal state that toggles each time to the button is clicked. Set the Button.Style property to tasCheck and the button will behave as a checkbox. The Down property can be set to force the checked state.

This sample code creates a toolbar with 3 groups. The first group contains a regular button, the second group a checkbox button and the last group radiobuttons. Between these groups are separators:

```
begin
  // insert regular button
  btn := TAdvToolBarButton.Create(self);
  btn.Caption := 'Button';
  btn.ShowCaption := true;
  atb.AddToolBarControl(btn);
  sep := TAdvToolBarSeparator.Create(self);
  atb.AddToolBarControl(sep);

  // insert check button
  btn := TAdvToolBarButton.Create(self);
  btn.Caption := 'Check';
  btn.Style := tasCheck;
  btn.ShowCaption := true;
  atb.AddToolBarControl(btn);

  sep := TAdvToolBarSeparator.Create(self);
  atb.AddToolBarControl(sep);

  // insert radio button group
  btn := TAdvToolBarButton.Create(self);
  btn.Caption := 'Radio 1';
  btn.GroupIndex := 1;
  btn.ShowCaption := true;
  atb.AddToolBarControl(btn);

  btn := TAdvToolBarButton.Create(self);
  btn.Caption := 'Radio 2';
  btn.GroupIndex := 1;
  btn.ShowCaption := true;
  atb.AddToolBarControl(btn);

  // initialize last button as checked button on the toolbar
  btn := TAdvToolBarButton.Create(self);
  btn.Caption := 'Radio 3';
  btn.GroupIndex := 1;
  btn.Down := true;
  btn.ShowCaption := true;
  atb.AddToolBarControl(btn);
```
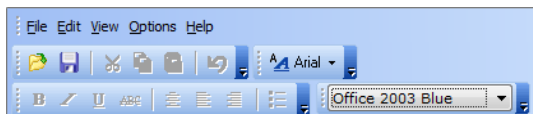
```
end;
```

To use a dockable TAdvMainMenu with TAdvDockPanel, add the TAdvDockPanel on the form and add a TAdvToolBar. Then add a TAdvMainMenu, use the design time menu editor to create the menu items and assign the TAdvMainMenu to the AdvToolBar.Menu property. Do not forget to remove the reference to TAdvMainMenu from the Form.Menu property. Set the AdvToolBar.FullSize property to true. This ensures that the menu is top aligned and uses the full width of the TAdvDockPanel. Finally, to apply a specific Office or Windows color style to the menu and toolbars, drop a TAdvMenuOfficeStyler and TAdvToolBarOfficeStyler component on the form. Assign the TAdvToolBarOfficeStyler to AdvDockPanel.ToolBarStyler and assign TAdvMenuOfficeStyler to AdvToolBarOfficeStyler.AdvMenuStyler.

TAdvDockPanel & TAdvToolBar property overview

TAdvDockPanel specific properties

Align	Sets the alignment of the dockpanel to left, right, top or bottom side of the form
AlignWithMargins	Take margins of the container (form) in account to do alignment
LockHeight	When true, the height of the TAdvDockPanel does not change. When false, the height of the dockpanel automatically adapts to the number of toolbars in the dockpanel.
Persistence	Specifies the location (registry or ini file) where to persist the states of the toolbars on the dockpanel(s)
ToolBarStyler	Sets the styler of the dockpanel and any toolbar on the dockpanel
UseRunTimeHeight	When true, the height of the dockpanel at design time is equal to the height at runtime. This is by default false. When false, the dockpanel is slightly larger than the toolbars on the dockpanel. This is to make sure that at design time, the component context menu will continue to work to insert more toolbars.

TAdvToolBar specific properties

This lists the properties that are relevant for a toolbar on a dockpanel. There are additional properties that are only used when the toolbar is on a ribbon page. This is discussed in the next chapter.

AllowFloating	When false, the toolbar cannot be dragged from the dockpanel to make it floating
AntiAlias	Specifies the font rendering used on the toolbar
AutoArrangeButtons	When true, buttons are automatically left aligned on the toolbar
AutoDockOnClose	When true, if a floating toolbar is closed, it is automatically docked again in the dockpanel
AutoOptionMenu	When true, the toolbar automatically shows a popup menu from the right-side handle from where toolbar buttons can be set hidden or visible
Caption	Sets the caption of the toolbar as shown on the floating window when the toolbar is in floating state
CaptionAlignment	Sets the alignment of the caption. On a floating window, this is always left aligned. Right alignment only applies for a toolbar on a ribbon.
CaptionFont	Sets the font of the floating window caption. When a toolbar styler is

	assigned, this is controlled by AdvToolbarOfficeStyler.CaptionFont
CaptionHeight	Sets the height of the floating window caption
DisabledImages	Imagelist holding images for disabled buttons on the toolbar. By default, disabled images are automatically generated from the normal state images
DockableTo	This set property defines to what specific dockpanels the toolbar can be docked. By default, the toolbar can be docked to all sides of the form (when a dockpanel is present on that side)
DockMode	When set to dmParentDockPanelOnly, the toolbar can only be docked back to its original parent dockpanel
FullSize	When true, the toolbar uses the full width (height) of the dockpanel. FullSize is typically set to true for toolbars holding the main application menu.
HintCloseButton	Sets the hint displayed on the close button of the floating toolbar window
HintOptionButton	Sets the hint on the option button on the right handle of the toolbar
Images	Holds the images used on toolbar buttons on the toolbar
Locked	The toolbar cannot be moved by dragging with the mouse
Menu	Sets the menu to be used with the toolbar
OfficeHint	Holds the title, notes, image for the Office 2007 style hint that can be shown on the toolbar. To have Office 2007 style hints, it is required to drop one instance of TAdvOfficeHint on the form
OptionMenu	Sets the menu that is to be displayed on click on the right handle
ParentStyler	Default true, the toolbar uses the styler as set for its parent, ie. the dockpanel
ShowOptionIndicator	When true, shows an indicator for clicking and opening an option menu from the right handle on the toolbar.
ShowRightHandle	When true, shows a handle from where an option menu can be displayed on the right side of the toolbar.
TextAutoOptionMenu	Sets the menu item caption for the menu item from where toolbar customization can be started
ToolBarStyler	Sets the styler for the toolbar. If the toolbar is on a dockpanel, it is recommended to use the dockpanel.ToolBarStyler property

TAdvToolBarButton specific properties

AllowAllUp	If AllowAllUp is true, all of the buttons in a group can be unselected at the same time. See also GroupIndex
Appearance	Sets the colors for the various button states. When a toolbar styler is used, the appearance is controlled by the styler.
AutoSize	the size of the button automatically adapts to the toolbar button glyph and/or caption
Down	For a button with Style set to tasCheck or used in a radiogroup, this property can be used to get and set the down state of the button
DropDownButton	When true, the button has a small dropdown button area on the right side
DropDownMenu	Assigns the menu that is shown when the dropdown button is clicked
DropDownSplit	When true, the button has two areas. The regular button area that generates the OnClick event when clicked and the dropdown button part that generates the OnDropDown event or shows the assigned dropdown menu
Glyph	Sets the button bitmap glyph in normal button state
GlyphChecked	Sets the button bitmap glyph in checked button state. When not specified, the normal glyph is used
GlyphDisabled	Sets the button bitmap glyph in disabled button state. When not specified, the button displays a grayed version of the normal glyph
GlyphDown	Sets the bitmap glyph used in down button state. When not specified, the normal glyph is used

GlyphHot	Sets the bitmap glyph used in hot button state. When not specified, the normal glyph is used
GlyphPosition	Sets the position of the glyph wrt the button caption
GroupIndex	Sets the index of the radio button group. When zero, the button is not grouped. All buttons with the same GroupIndex are treated as a radiobutton group
ImageIndex	Set the index of the image to use for the button in case an imagelist is assigned
OfficeHint	Sets the Title, Notes, Image for the optional Office 2007 / Office 2010 style hint that can be shown for the button.
Picture	Sets the picture used for the button normal state. This supports the picture formats available in the VCL
PictureDisabled	Sets the picture used for the button disabled state. This supports the picture formats available in the VCL
Position	Sets the position of the button's toolbar. When Position is set to daLeft or daRight, a button caption will be rotated (when a TrueType font is used)
Shaded	When true, automatically applies a shade on the button glyph in the down state
ShowCaption	When true, the button caption is displayed on the button
Style	Can be set to tasButton for regular button behavior or tasCheck for checkbox (toggle Down state) behavior

TAdvToolBarSeparator specific properties

LineColor	Sets the line color of the separator that sits between buttons on the toolbar
SeparatorStyle	Selects between 2 styles: ssBlank: blank gap between buttons ssOffice2003: line with shadow

TDBAdvToolBarButton specific properties

ConfirmAction	When true, a message prompts to confirm the click and allow the operation on the databound dataset.
ConfirmActionString	Text to show in the messagebox prompt to allow the action
DataSource	DataSource to which the button is bound
DBButtonType	Action the button performs on the dataset dbtPrior: move to previous record dbtNext: move to next record dbtFirst: move to first record dbtLast: move to last record dbtEdit: put dataset in edit mode dbtPost: post changes to dataset dbtCancel: cancel edits and put dataset back in browse mode dbtDelete: delete record dbtAppend: append new record dbtInsert: insert new record dbtCustom: custom action on datasource, can be executed in the OnBeforeAction event

TDBAdvToolBarNavigator

This inserts the TDBAdvToolBarButton controls that form together a DB navigator

TAdvToolBarContainer specific properties

A container can be used when it is need to position controls absolute on the toolbar within the container. Otherwise, controls are always automatically left aligned. Within the container, the controls can have any position.

Color	Sets the background color of the container
Line3D	When true, the line around the container is drawn with a 3D effect
LineColor	Sets the line color of the container border
OfficeHint	Sets the Title, Notes, Image for the optional Office 2007 / Office 2010 style hint that can be shown for the button.

Toolbar customization

TAdvToolBar comes with a companion component TAdvToolBarCustomizer. This is a component that encapsulates a dialog for editing the sequence and visibility of buttons on the toolbar. To start using the component, drop a TAdvToolBarCustomizer on the form. Assign the toolbar to be edited to AdvToolBarCustomizer.AdvToolBar. The toolbar customizer dialog will be shown when choosing "Customize toolbar" from the contect menu on the right handle of the toolbar. In this dialog, a series of predefined commands can be specified via the AdvToolBarCustomizer.Commands collection. This sets the caption, imageindex and optionally the action of commands at the disposal of the user to put on the toolbar. The user settings of the toolbar can be persisted in a file defined by AdvToolBarCustomizer.FileName.

Example:

Using the TAdvToolBarCustomizer and an ActionList, it is possible without writing any line of code to offer the user the capability to fully customize the actions available on the toolbar. To do this, drop a TAdvToolBar, TAdvToolBarCustomizer, TActionList and TImageList on the form. Fill the actionlist with actions that can be available on the toolbar. Assign the AdvToolBar instance to TAdvToolBarCustomizer.AdvToolBar. Open the Commands collection property and add for each action an item. Run the application and click the right-handle of the toolbar to start the configuration dialog. Add the desired actions to the left and these actions will appear as buttons on the toolbar. Via the INI file configured via TAdvToolBarCustomizer.FileName, the configuration of the toolbar is automatically persisted.
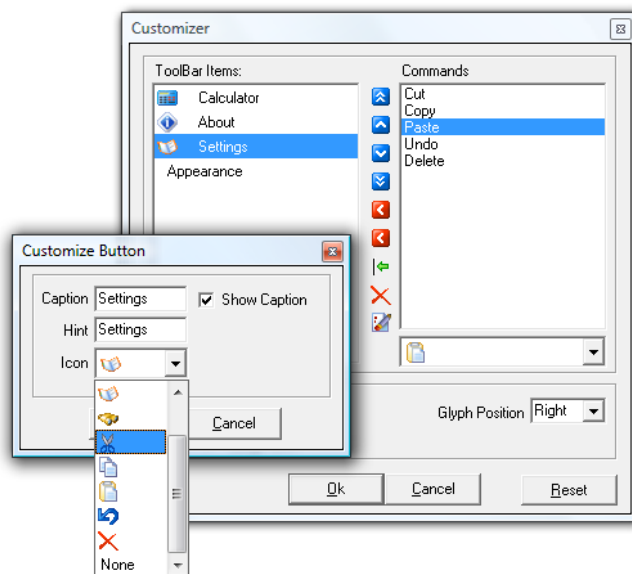
Toolbar persistence

As the user can move toolbars, dock and undock toolbars, it is often desirable that the exact state of the various toolbars is persisted and restored when the user restarts the application. This is very easy to do with TAdvDockPanel. Set AdvDockPanel.Persistence.Enabled = true and select the storage between INI file and registry. When the INI file is chosen, set the name of the INI file via AdvDockPanel.Persistence.Key and the section within the INI file via AdvDockPanel.Persistence.Section. When the registry is choosen, the AdvDockPanel.Persistence.Key property sets the registry key name and the property AdvDockPanel.Persistence.Section the value name. The dockpanel will now automatically persist the latest state of the toolbars when the application closes and restore it upon opening of the form. If it is not desirable that the component does this automatically for you, set AdvDockPanel.Persistence.Enabled to false and call AdvDockPanel.SaveToolBarsPosition to store the settings and call AdvDockPanel.LoadToolBarsPosition to restore the states.

# TMS Advanced ToolBar : Office 2007 / 2010 / Windows 7 scenic ribbon mode

Getting started with ribbons

In the IDE, choose File, New, Other and under "TMS Wizard", pick "TMS ToolBar Application Wizard" or "TMS Office 2010 ToolBar Application Wizard". This will create a new application for you with a TAdvToolBarPager, TAdvPage, TAdvToolBar, TAdvPreviewMenu, TAdvShapeButton. This is generated in a form descending from TAdvToolBarForm that takes care of border size, painting, and caption. In the case of an Office 2010 application wizard, this will also generate a frame that contains the skeleton of the application menu built from PolyList controls.

To start an Office 2007 or Office 2010 ribbon style toolbar application manually, drop the TAdvToolBarPager container and TAdvToolBarOfficeStyler component on the form. Assign the TAdvToolBarOfficeStyler to TAdvToolBarPager.ToolBarStyler. Right-click on the TAdvToolBarPager and choose "New Page" to insert pages in the pager. On the page, right-click and select "Add toolbar" to insert a toolbar.
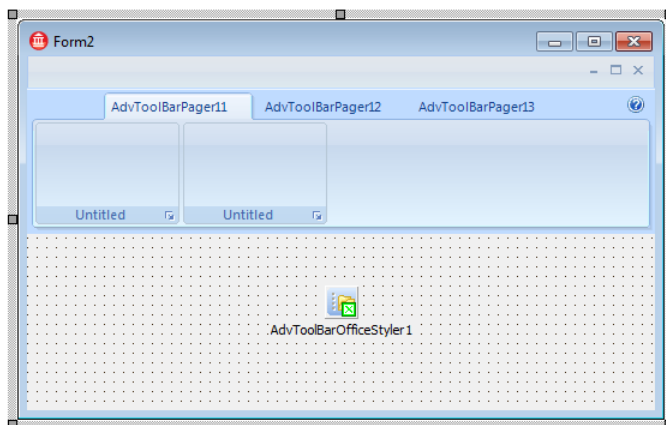
Additionally, the quick access toolbar (QAT) can be added by right-click on the TAdvToolBarPager and choosing "Add quick access toolbar" and the application menu button can be added from the context menu as well with "Add application menu button".
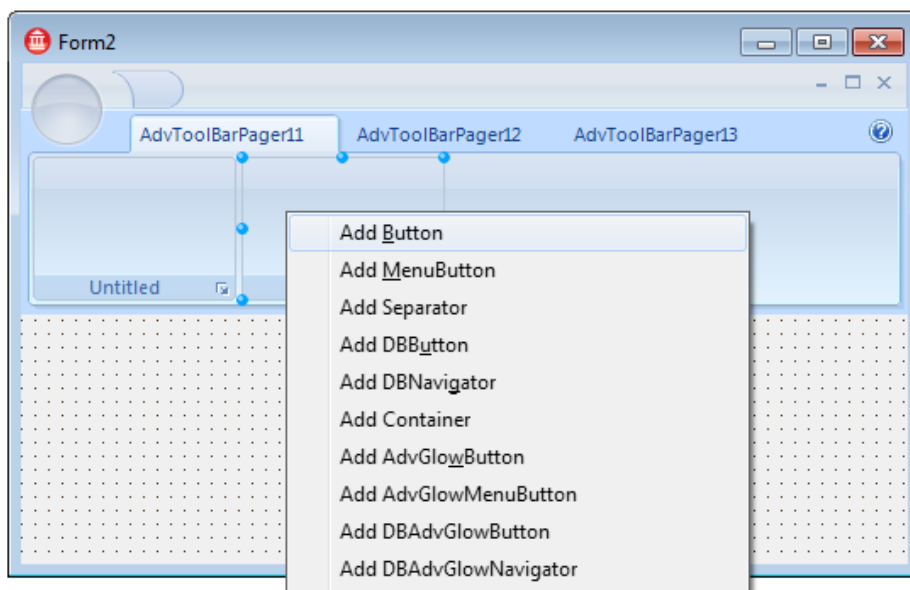
The TAdvToolBarPager consists mainly of three parts: a caption, an area for page tabs and pages. The caption can be used to host other controls (like shortcut buttons, referred to further as QAT: quick access toolbar), to show a caption text and will also show grouped page tabs. The caption can be optionally hidden. Set TAdvToolBarPager.Caption.Visible = false to hide the caption area. The TAdvToolBarPager is further very similar to a standard TPageControl. Pages can be inserted, an active page can be set by the property TAdvToolBarPager.ActivePage, a page can be temporarily hidden by setting TAdvPage.TabVisible to false.

Programmatically creating pages, toolbars, changing toolbars etc... is made easy with a couple of TAdvToolBarPager methods. This code snippet programmatically inserts a new page and creates a new toolbar and sets the toolbar caption in the inserted page:

```
begin
  with AdvToolBarPager1 do
  begin
    AddAdvPage('New page');
    ActivePage := AdvPages[AdvPageCount - 1];
    AdvPages[AdvPageCount - 1].CreateAdvToolBar;
    AdvPages[AdvPageCount - 1].AdvToolBars[0].Caption := 'New toolbar';
  end;
end;
```

TAdvToolBarPager properties overview

ActivePage	Sets which page is active in the pager
AntiAlias	Sets the font rendering mode to be used
AutoMDIButtons	When true, the window close, minimize, maximize buttons are automatically displayed in the upper right corner of the TAdvToolBarPager
Caption	Sets the caption text, caption height, indent from left of the caption text of the TAdvToolBarPager
CaptionButtons	Sets what window buttons should be displayed in the caption. By default these are the default window close, minimize, maximize buttons
DisabledImages	Sets the imagelist to be used for images in disabled tabs

EnableWheel	When true, when the mousewheel is used when the mouse is over the TAdvToolBarPager, this will do the active page selection. For Office 2007 UI compliance, this should be set to true.
Expanded	Public boolean property to set the TAdvToolBarPager in expanded or collapsed state.
HidePagesOnDblClick	When true, a double click on the page tabs will hide/show the pager. For Office 2007 UI compliance, this should be set to true
Images	Sets the imagelist to be used for images in tabs
OfficeHint	Sets the Title, Notes, Image for the optional Office 2007 / Office 2010 style hint that can be shown for the button.
OptionDisabledPicture	Sets the picture to be used in the bottom left corner of a disabled toolbar from where an options dialog can be shown
OptionPicture	Sets the picture to be used in the bottom left corner of a toolbar from where an options dialog can be shown
PageLeftMargin	Sets the margin from left for the first page in the TAdvToolBarPager
PageRightMargin	Sets the margin from right for the last page in the TAdvToolBarPager
Persistence	Holds the settings for persisting the state of the toolbars in either an INI file or the registry
ShowExpandCollapsButton	When true, a small button is shown in the top left corner of the TAdvToolbarPager to expand or collaps the pages
ShowHelpButton	When true, a small button is shown in the top left corner of the TAdvToolbarPager from where help can be shown
ShowNonSelectedTabs	When true, the tab border is drawn for tabs that are not selected. This is controlled by the TAdvToolBarOfficeStyler to ensure the look is identical to the Office 2007 / Office 2010 ribbon
ShowQATBelow	Public Boolean property to set the QAT position below the TAdvToolBarPager instead of in the caption.
ShowTabHint	When true, a hint can be shown when the mouse is over a page tab
TabGroups	Collection of groups. This sets: tab group caption, appearance of tabgroup and index of first page in the tabgroup and last page in the tabgroup
TabSettings	Holds the settings for displaying the page tabs. This includes spacing between tabs, margin from left for the first tab, tab height, margin from left and right between the tab text and tab border
ToolBarStyler	Sets the styler of the TAdvToolBarPager. TAdvToolBars on the pages of a TAdvToolBarPager will use this style as well as toolbar button controls on the toolbars
Win7ScenicRibbon	When true, the toolbarpager will have the behavior & look of a Windows 7 scenic ribbon. Make sure to set the TAdvToolBarOfficeStyler.Style also to tsWindows7

TAdvToolBarPager methods

AddAdvPage	Adds a new page to the TAdvToolbarPager
RemoveAdvPage	Removes a page from the TAdvToolbarPager
MoveAdvPage	Moves the position of a page in the TAdvToolbarPager from current index to new index
FindNextPage	Finds next accessible page
SelectNextPage	Selects next accessible page
NextActivePage	Selects the next active page
PrevActivePage	Selects the previous active page
IndexOfPage	Retrieves the index of a page
Expand	Sets the TAdvToolbarPager in expanded state

Collaps	Sets the TAdvToolbarPager in collapsed state

TAdvQuickAccessToolBar specific properties overview

CustomizeHint	this sets the hint to be displayed when the mouse hovers the button on the right side of the QAT. This button is visible when ShowCustomizeButton is set to true
DisabledImages	Sets the imagelist to be used for images in disabled buttons on the QAT
DropDownPopupMenu	Sets the menu to be displayes when the right side button is clicked
Images	Sets the imagelist to be used for images in buttons on the QAT
ShowCustomizeOption	When true, a small extra button is displayed on the right side of the QAT from where options can be configured

TAdvShapeButton : application menu button specific properties overview

Action	Sets the optional action for click on the application menu button
AdvPreviewMenu	Sets the application menu instance to display when the button is clicked. This is the Office 2007 style application menu instance
AntiAlias	Sets the font rendering mode on the button
Appearance	Sets the colors for the button in its various states
OfficeHint	Sets the Title, Notes, Image for the optional Office 2007 / Office 2010 style hint that can be shown for the button
Picture	Sets the picture for the button in normal state
PictureDisabled	Sets the picture for the button in disabled state
PictureDown	Sets the picture for the button in down state
PictureHot	Sets the picture for the button in hot state
ShortCutHint	Shortcut value that is shown in a small hint window when Alt key is pressed. The shortcut will perform the button click
ShortCutHintPos	Sets the position of the hint wrt the button

TAdvToolBarPage specific properties

AutoPosition	When true, the toolbars on the page are automatically positioned next to each other starting from left
Caption	Sets the caption for the page
ImageIndex	Sets the index of an image in the imagelist to show along the caption in the page tab
OfficeHint	Sets the Title, Notes, Image for the optional Office 2007 / Office 2010 style hint that can be shown for the page tab
PageIndex	Index of the page within the TAdvToolBarPager
ShortCutHint	Shortcut value that is shown in a small hint window on the tab when Alt key is pressed. The shortcut will activate the tab
ShortCutHintPos	Sets the position of the hint wrt the tab
TabEnabled	When true, the tab is enabled and can be clicked to change the active tab
TabVisible	When true, the tab is visible. When false, the tab is not visible but the page still exists but is not visible

TAdvGlowButton specific properties

Action	Sets the action associated with the button
AllowAllUp	If AllowAllUp is true, all of the buttons in a group can be unselected at the same time. See also GroupIndex
AntiAlias	Sets the font rendering mode on the button
Appearance	Sets the colors for the button in its various states
AutoSize	When true, the size of the button is automatically changed to fit

	image and caption
Caption	Sets the caption of the button
DisabledImages	Sets the imagelist to be used for images in disabled buttons
DisabledPicture	Sets the picture for the button in disabled state
Down	For a button with Style set to bsCheck or used in a radiogroup, this property can be used to get and set the down state of the button
DropDownButton	When true, the button has a small dropdown button area on the right side or at the bottom
DropDownDirection	The dropdown button can be displayed either on the right side or at the bottom of the button.
DropDownMenu	Assigns the menu that is shown when the dropdown button is clicked.
DropDownSplit	When true, the button has two areas. The regular button area that generates the OnClick event when clicked and the dropdown button part that generates the OnDropDown event or shows the assigned dropdown menu.
FocusType	Defines how the button focus is displayed on the button.
GroupIndex	Sets the index of the radio button group. When zero, the button is not grouped. All buttons with the same GroupIndex are treated as a radiobutton group
HotImages	Sets the imagelist to be used for images in hot buttons
HotPicture	Sets the picture for the button in hot state
ImageIndex	Set the index of the image to use for the button in case an imagelist is assigned
Images	Sets the imagelist to be used for images in buttons
InitRepeatPause	Sets the delay in milliseconds before repeated OnClick events are triggered when the mouse is continously pressed down on the button
Layout	Sets the position of the image on the button wrt the caption
MarginHorz	Sets the horizontal margin in pixels between the image on the button and the caption
MarginVert	Sets the vertical margin in pixels between the image on the button and the caption
MaxButtonSizeState	Defines the largest state the button can have when the toolbar containing the button resizes
MinButtonSizeState	Defines the smallest state the button can have when the toolbar containing the button resizes
Notes	Sets the description text that can be displayed on the button in addition to the caption
NotesFont	Sets the font of the description text that can be displayed on the button in addition to the caption
OfficeHint	Sets the Title, Notes, Image for the optional Office 2007 / Office 2010 style hint that can be shown for the button
Picture	Sets the picture for the button
Position	Sets the position of the button as standalone button or button position within a visual group of buttons
RepeatClick	When true, the button will trigger repeated OnClick events when the mouse is continously pressed down on the button
RepeatPause	Sets the delay in milliseconds between repeated clicks
Rounded	When true, the button is displayed with rounded corners
ShortCutHint	Shortcut value that is shown in a small hint window on the tab when Alt key is pressed. The shortcut will activate the button's OnClick event
ShortCutHintPos	Sets the position of the hint wrt the button
ShowCaption	When true, the caption is displayed along the image on the button
Style	Can be set to bsButton for regular button behavior or bsCheck for checkbox (toggle Down state) behavior

Transparent	When true, the button is transparent in normal state
WordWrap	When true, the caption on the button is displayed with wordwrapping

TDBAdvGlowButton specific properties

ConfirmAction	When true, a message prompts to confirm the click and allow the operation on the databound dataset.
ConfirmActionString	Text to show in the messagebox prompt to allow the action
DataSource	DataSource to which the button is bound
DBButtonType	Action the button performs on the dataset
	dbtPrior: move to previous record
	dbtNext: move to next record
	dbtFirst: move to first record
	dbtLast: move to last record
	dbtEdit: put dataset in edit mode
	dbtPost: post changes to dataset
	dbtCancel: cancel edits and put dataset back in browse mode
	dbtDelete: delete record
	dbtAppend: append new record
	dbtInsert: insert new record
	dbtCustom: custom action on datasource, can be executed in the OnBeforeAction event

TDBAdvGlowNavigator

This inserts the TDBAdvGlowButton controls that form together a DB navigator

Using the TAdvToolBarPager to move, close, minimize and maximize a captionless form

The TAdvToolBarPager can be used to replace a regular form caption, ie. it allows to move the form, close, minimize or maximize it. To achieve this, set TAdvToolBarPager.CanMove = true and enable the buttons that should be visible in the caption under TAdvToolBarPager.CaptionButtons. To have a rounded border, descend your form from TAdvToolBarForm.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, AdvGlowButton, AdvShapeButton, AdvToolBar, AdvToolBarStylers;

type
  TForm2 = class(TAdvToolBarForm)
    AdvToolBarPager1: TAdvToolBarPager;
    AdvToolBarPager11: TAdvPage;
    AdvToolBarPager12: TAdvPage;
    AdvToolBarPager13: TAdvPage;
    AdvToolBar1: TAdvToolBar;
    AdvToolBar2: TAdvToolBar;
    AdvToolBarOfficeStyler1: TAdvToolBarOfficeStyler;
    AdvQuickAccessToolBar1: TAdvQuickAccessToolBar;
    AdvShapeButton1: TAdvShapeButton;
    DBAdvGlowButton1: TDBAdvGlowButton;
    DBAdvGlowButton2: TDBAdvGlowButton;
```

Appearance on Windows XP:
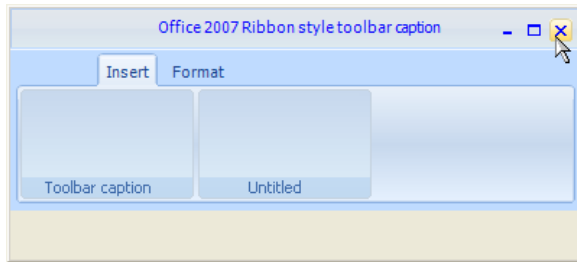
Appearance on Windows Vista / Windows 7:
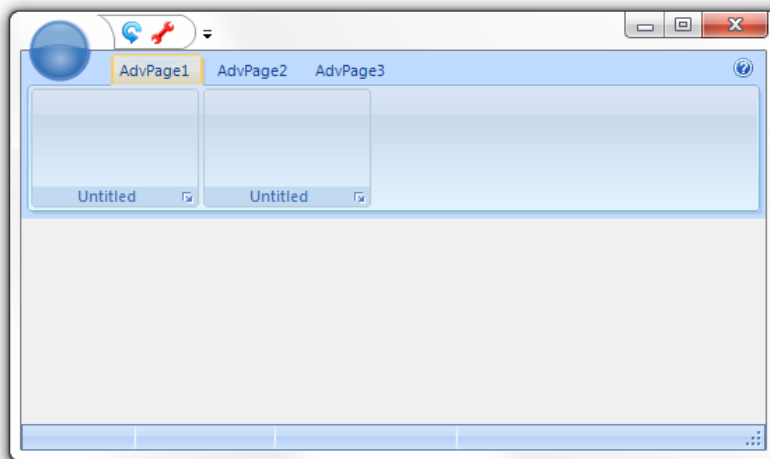
Note that the wizard to create an Office 2007 ribbon or Office 2010 ribbon skeleton application automatically creates a form descending from TAdvToolBarForm.

Adding controls

TAdvToolBar supports a number of built-in controls that can be added on the toolbar. The Office 2003 toolbar controls can be used but for closer Office 2007 appearance, it is recommended to use the new TAdvGlowButton, TAdvGlowMenuButton, TDBAdvGlowButton controls. Inserting controls is done by right-click on the toolbar and choosing "Add AdvGlowButton", "Add AdvGlowMenuButton", ... The position of the controls in the TAdvToolBar are by default automatically controlled by the TAdvToolBar itself and will be inserted starting from left in the toolbar. To disable this automatic position control, set TAdvToolBar.AutoPositionControls = false. In this mode, the controls can be positioned at exact coordinates in the toolbar.

Example: adding buttons programmatically to the toolbar:

```
var
  agb: TAdvGlowButton;
begin
  agb := TAdvGlowButton.Create(self);
  agb.Caption := 'New button';
  agb.Transparent := true;
  AdvToolBar1.AddToolBarControl(agb);
end;
```

The TAdvGlowButton or TAdvGlowMenuButton are two control types that can be added to the TAdvToolBar and that can be used in different ways:
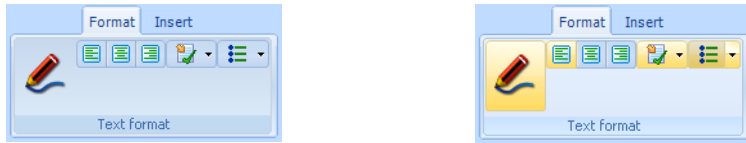
In the screenshot above, the first TAdvGlowButton (with pen picture) is used in transparent mode. The button will only highlight when the mouse hovers or clicks the button. The align buttons are default TAdvGlowButton controls but to make the three align buttons appear as a group, the first left align button Position property was set to bpLeft, the middle center alignment button Positon was set to bpMiddle, the right align button Position was set to bpRight. The GroupIndex of the three buttons was set to 1 and this way the buttons automatically behave as radiobuttons. Finally, two TAdvGlowButton controls were added with DropDownButton = true. For the first button, DropDownSplit is set to false to let the component behave as a single button. For the second button, DropDownSplit is set to true. The button acts as a dual button control, one for the main button and one for the dropdown part. OnClick is triggered for a click on the main button, OnDropDown is triggered for a click on the dropdown part.

Visual button groups

As can be seen in Office 2007 or Office 2010, some buttons that have related functionality are visually grouped, ie. the outer borders of the buttons are rounded and between the buttons there is just a single line. It is possible with TAdvGlowButton instances on the toolbar to create such button groups. This is done by setting the property AdvGlowButton.Position. By default, AdvGlowButton.Position is bpStandalone. This means that the button has a border on each side. Other positions are: bpLeft, bpMiddle, bpRight. To create a visual group of buttons, set the leftmost button Position to bpLeft, the rightmost button Position to bpRight and all other buttons as bpMiddle.

Example:

```
procedure AddButton(ACaption: string; APosition: TButtonPosition);
var
  agb: TAdvGlowButton;
begin
  agb := TAdvGlowButton.Create(self);
  agb.Caption := ACaption;
  agb.Position := APosition;
  agb.Width := 24;
  agb.Height := 24;
  AdvToolBar1.AddToolBarControl(agb);
end;

begin
  AddButton('1',bpLeft);
  AddButton('2',bpMiddle);
  AddButton('3',bpMiddle);
  AddButton('4',bpRight);
end;
```

Result:
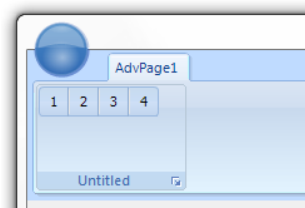
Tab groups

Toolbars that are in multiple pages and that functionally belong together can be grouped with tabgroups. Tabgroups are managed on the level of the TAdvToolBarPager via the TabGroups property.
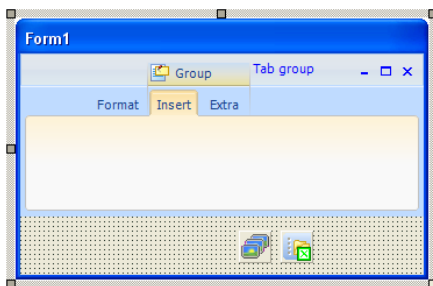

A TabGroupItem can be inserted in the TAdvToolBarPager.TabGroups property and this TabGroupItem controls which tabs belong to the same group. This is set with the property TabGroupItem.TabIndexStart (zero based index of the first tab that belongs to the group) and TabGroupItem.TabIndexEnd. In the example image, the TabIndexStart is set to 1, the TabIndexEnd is set to 2. A group indicator will be displayed on top of the tabs that belong to the same group with a caption text set by TabGroupItem.Caption and optionally an image set by TabGroupItem.ImageIndex. By default, the tabs that belong to a tabgroup are displayed with the TAdvToolBarOfficeStyler.GroupAppearance.TabAppearance settings. In the sample screenshot, the tab of a group is shown in light orange while other tabs and displayed with a soft blue. The same applies for the TAdvPage background. When there is a need to have different colors for different groups, the colors for each group can be customized under TabGroupItem.GroupAppearance.

Compact mode toolbars

By default, a toolbar on a TAdvPage shows all controls on the toolbar. However, when a window gets too small to show all toolbars with all controls, a toolbar switches to compact mode. In this compact mode, a toolbar reverts to a single AdvGlowButton from where a click displays the full toolbar. The screenshots below show this in action. The first window is sufficiently wide to show all toolbars.
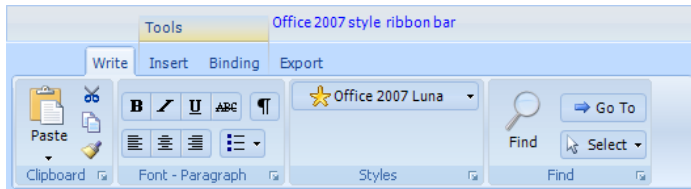
When the window becomes smaller, the Find toolbars reverts to compact mode:
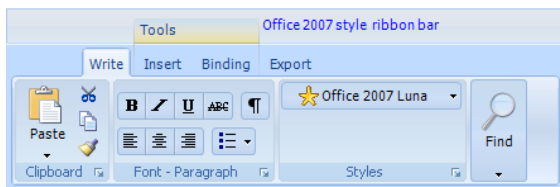
Finally, for a very small window, all toolbars have reverted to compact mode and the Clipboard compact toolbar button is clicked for access to the full toolbar:
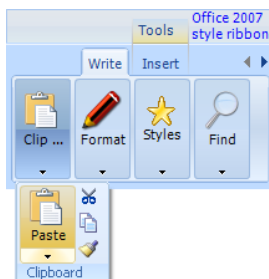
Using compact toolbars is easy as everything is done automatically in TAdvToolBar. All that is required is to set the property TAdvToolBar.CompactCaption and TAdvToolBar.CompactPicture. This is the caption and picture that is used for the dropdown button from where the full toolbar is displayed when the toolbar is in compact mode.

Office 2007 / Office 2010 hints

All controls that are part of the advanced toolbars & menus feature an extended Office 2007 / Office 2010 style hint. Office 2007 hints are more sophisticated than regular Delphi hints. An Office 2007 / Office 2010 hint has a title, an optional picture, help text and an optional help line.
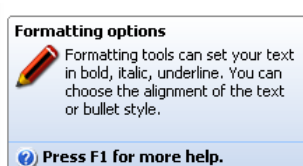
This title, help text, picture and optional help line is set through the property OfficeHint which has subproperties:

- OfficeHint.Notes: stringlist containing help text
- OfficeHint.Picture: BMP, ICO, GIF, JPG or PNG image for the hint. Preferred image type if PNG as the alpha channel is used for transparency
- OfficeHint.ShowHelp: When true, an extra line is shown in the hint to suggest to press F1 for help
- OfficeHint.Title: sets the title of the hint

To make use of the new hint style, drop the component TAdvOfficeHint on the main form and use the OfficeHint property of the components to set hint subject, picture, notes and help line. The regular Hint property is no longer used when a control has an OfficeHint property. Set ShowHint to true when the hint is enabled for a control.

Any other control can be extended with Office 2007 / Office 2010 hints as well. It is sufficient for this to add the property OfficeHint:TAdvHintInfo to the control and TAdvOfficeHint will automatically start using this instead of the regular Hint property.

This code snippet shows the code that was added to extend a custom control with the Office 2007 hint type. Note that the unit AdvHintInfo is added to the uses hint as the TAdvHintInfo class is defined in this unit.

```pascal
interface

uses
  Classes, AdvHintInfo;

type

TMyCustomControl = class(TCustomControl)
private
  FOfficeHint: TAdvHintInfo;
  procedure SetOfficeHint(const Value: TAdvHintInfo);
public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
published
  property OfficeHint: TAdvHintInfo read FOfficeHint write SetOfficeHint;
end;

implementation
{ TMyCustomControl }

constructor TMyCustomControl.Create(AOwner: TComponent);
begin
  inherited;
  FOfficeHint := TAdvHintInfo.Create;
end;

destructor TMyCustomControl.Destroy;
begin
  FOfficeHint.Free;
  inherited;
end;

procedure TMyCustomControl.SetOfficeHint(const Value: TAdvHintInfo);
```
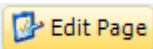
```
begin
  FOfficeHint.Assign(Value);
end;
```

Controlling toolbar button sizing

The Office 2007 / Office 2010 ribbon GUI has sophisticated capabilities to resize controls depending on the size of the window & its toolbars. A button can have 3 different size states:

- glyph size (bsGlyph size) :

- label size (bsLabel size) :

- large size (bsLarge size) :

During resizing of the form that contains a toolbar with buttons, the buttons can automatically change their size state. This can be seen below where toolbar buttons change the size state in order to always try to fit all available buttons:

Toolbar is at maximum size. All buttons are in bsLarge size state

Toolbar shrinks. Half of the buttons shrink to bsLabel size state

Next step in shrinking process is that all buttons are now in the bsLabel size state

Half of the buttons shrink to bsGlyph size state

All buttons are in mimimum size type: bsGlyph size state

This is the default sizing behavior that the toolbar automatically performs when it shrinks or expands. This sizing behavior can be controlled on the level of a toolbar button through the properties:

- MinButtonSizeState : minimum size the button can be set to
- MaxButtonSizeState : maximum size the button can be set to

In the above example, all buttons on the toolbar have the MinButtonSizeState set to bsGlyph and MaxButtonSizeState set to bsLarge.

It is not always desirable that all buttons can have a varying size state between bsGlyph and bsLarge. For the clipboard toolbar for example, following layout is achieved by setting:

- Paste button: MinButtonSizeState is bsLarge, MaxButtonSizeState is bsLarge
- Cut, Copy, Format button: MinButtonSizeState is bsGlyph, MaxButtonSizeState is bsGlyph

This guarantees that irrespective of the toolbar size, the buttons have always the same appearance:
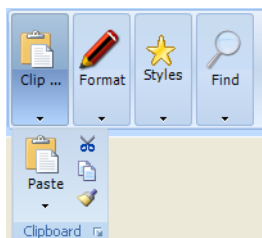
Using the TAdvShapeButton & TAdvPreviewMenu as Office 2007 style application menu

In the Office 2007 ribbon a round button that sits in the top right corner of the window from where an application menu can be shown that resembles the old File menu.
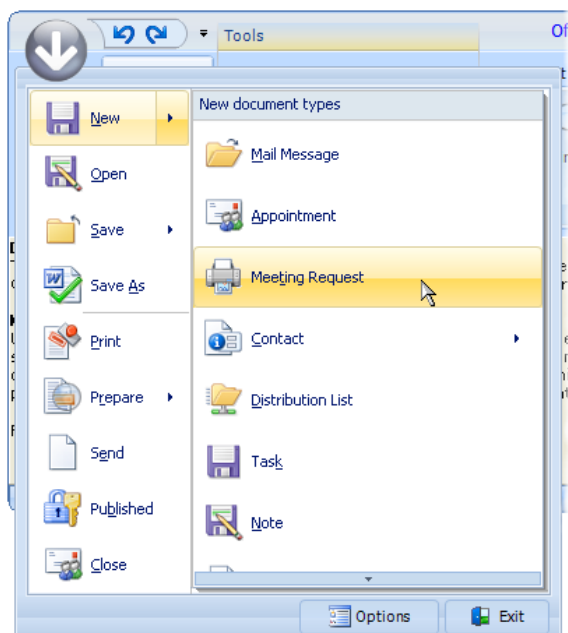
What makes this menu a little different from a regular menu is that it has a frame with 2 panes. On the left pane, the menu items are displayed. On the right pane, either submenu items for left menu items are shown or a fixed set of items to select from. TAdvShapeButton provides such typical round button and TAdvPreviewMenu provides the application menu in Office 2007 style.

The TAdvShapeButton component is by default a round button that can display the value set with the Text property but it can also display images in PNG format that have a transparency on the TAdvToolBarPager. Make sure that the pager tabs do not overlap the button. The indent of the first pager tab can be set with the property AdvToolBarPager.TabSettings.StartMargin. In the demo, a round PNG image was used but nothing prevents you from using more irregular shapes. The TAdvShapeButton supports 4 images: Picture, PictureHot, PictureDisabled, PictureDown which set images to display for normal, hot, disabled and down state respectively. The TAdvShapeButton can be used as standalone button but more commonly will be used with a TAdvPreviewMenu and as such, it has a property AdvPreviewMenu with which the TAdvPreviewMenu component can be set that should be displayed when it is clicked.

The TAdvPreviewMenu is a non visual component at design time. It can be displayed at runtime by calling AdvPreviewMenu.ShowMenu(X,Y). When assigned to a TAdvShapeButton, it will be displayed automatically at the correct position by the TAdvShapeButton. The TAdvPreviewMenu has a frame with 2 panes. The left pane is being used to show MenuItems and the right pane is used to show SubMenuItems. The MenuItems are defined through the property TAdvPreviewMenu.MenuItems which is a collection of TAdvPreviewMenuItem instances. This class has following properties:

Action	Sets the optional action associated with the menu item
CanSelect	When true, the item can be selected (clicked) otherwise, it is only used to show SubMenuItems that can be selected
Caption	Sets the caption of the menu item DisabledPicture: sets the picture for the disabled state of the menu item
Enabled	When true, the menuitem is enabled
ImageIndex	Sets the picture for a menu item through an ImageList assigned to TAdvPreviewMenu.MenuImages
OfficeHint	Sets the Office 2007 style hint that is displayed when the mouse hovers the menu item
Picture	Sets the picture for the normal state of the menu item

Separator	When true, the menu item is just a separator, ie. small horizontal line between menu items
ShortCutHint	This sets the shortcut hint text that is shown when the menu is opened through its shortcut on the TAdvToolBarPagerShortCutSubItemsHint: this sets the shortcut hint text used to open the subitems
SubItems	This is a collection of subitems that can be displayed for a MenuItem. When items are available in the SubItems collection, a little arrow on the right side of the menu item indicates this.
SubMenuCaption	This sets the caption text of the submenu, shown at the top of the right pane
SubMenuItemSpacing	This sets the spacing between items in the submenu

When a menuitem is clicked, it triggers the event OnMenuItemClick with ItemIndex parameter indicating the menu item index that was clicked.

The right pane shows submenu items. These submenu items can either be defined globally for the TAdvPreviewMenu (and will be displayed for all MenuItems that have no SubItems defined) or can set per MenuItem. The global defined submenu items are set through TAdvPreviewMenu.SubMenuItems while the subitems per menu are set through TAdvPreviewMenu.MenuItems[index].SubMenuItems. A submenu item has following properties:

Action	Sets the optional action associated with the submenu item
DisabledPicture	Sets the picture for the disabled state of the submenu item
Enabled	When true, the menuitem is enabled
ImageIndex	Sets the picture for a submenu item through an ImageList assigned to TAdvPreviewMenu.SubMenuImages
Notes	This is a stringlist that can hold a description text for a submenu item
OfficeHint	Sets the Office 2007 style hint that is displayed when the mouse hovers the submenu item
Picture	Sets the picture for the normal state of the submenu item
ShortCutHint	This sets the shortcut hint text that is shown when the submenu is opened through its shortcut of the menu item

When a submenu item is clicked, it triggers the event OnSubMenuItemClick with ItemIndex parameter indicating the menu item index the submenu belongs to and SubItemIndex the submenu item index.

Finally, the TAdvPreviewMenu also has the capability to have one or more buttons at the bottom of the frame. The properties of the buttons are set through the Buttons collection. Each button is defined by a TButtonCollectionItem class with following properties:

Action	Sets the optional action associated with the button
Caption	Sets the caption of the button
DisabledPicture	Sets the picture for the disabled state of the button
Enabled	When true, the button is enabled
Font	Sets the font of the button
ImageIndex	Sets the picture for a menu item through an ImageList assigned to TAdvPreviewMenu.ButtonImages
OfficeHint	Sets the Office 2007 style hint that is displayed when the mouse hovers the button
Picture	Sets the picture for the normal state of the button
Visible	Sets the visibility of the button
Width	Sets the width of the button

When a button is clicked, it triggers the event OnButtonClick. This event returns the parameter ButtonIndex to indicate what button was pressed. Note that the TAdvPreviewMenu will not

automatically close when a button is clicked. The menu can be programmatically closed by calling:
AdvPreviewMenu.HideMenu

## Choosing a style with the TAdvPreviewMenuOfficeStyler

TAdvPreviewMenu also has a lot of color settings to customize every aspect of the control. Just like with the other controls that emulate the Office 2007 or Office 2010, predefined colors have been embedded in a separate styler component: TAdvPreviewMenuOfficeStyler. The component can be dropped on the form and assigned to TAdvPreviewMenu.Styler. In this styler, appearance for buttons, menuitem and frame can be set or more conveniently, a style can be chosen by setting the Style property. TAdvPreviewMenuOfficeStyler is also TAdvFormStyler, TAdvAppStyler compatible to make it easy to change the style of a complete form or application with a single property.

The Office 2007 / Windows 7 scenic ribbon and Office 2010 modes

The TAdvToolBarOfficeStyler will control via its style property the 3 Office 2007 modes, the Windows 7 scenic ribbon mode and the 3 Office 2010 modes. As can be seen in the screen shots, one of the major differences is the application menu button (TAdvShapeButton). The round button of the Office 2007 ribbon is replaced by a rounded rectangular button in the Windows 7 scenic ribbon and Office 2010 ribbon. In the Windows 7 scenic ribbon, this button has an image specific for this mode and in Office 2010 mode, this button can have a caption. This means that to properly let an application switch at runtime between these 3 ribbon modes, the TAdvShapeButton caption will have to be set accordingly as in the sample code here:

```
procedure TTMSForm3.AdvComboBox1Change(Sender: TObject);
begin
  case AdvComboBox1.ItemIndex of
  0:
    begin
      AdvToolBarofficeStyler1.Style := bsOffice2007Luna;
      AdvShapeButton1.Text := 'TMS';
    end;
  1:
    begin
      AdvToolBarofficeStyler1.Style := bsWindows7;
      AdvShapeButton1.Text := '';
    end;
  2:
    begin
      AdvToolBarofficeStyler1.Style := bsOffice2010Blue;
      AdvShapeButton1.Text := 'File';
    end;
  end;
end;
```
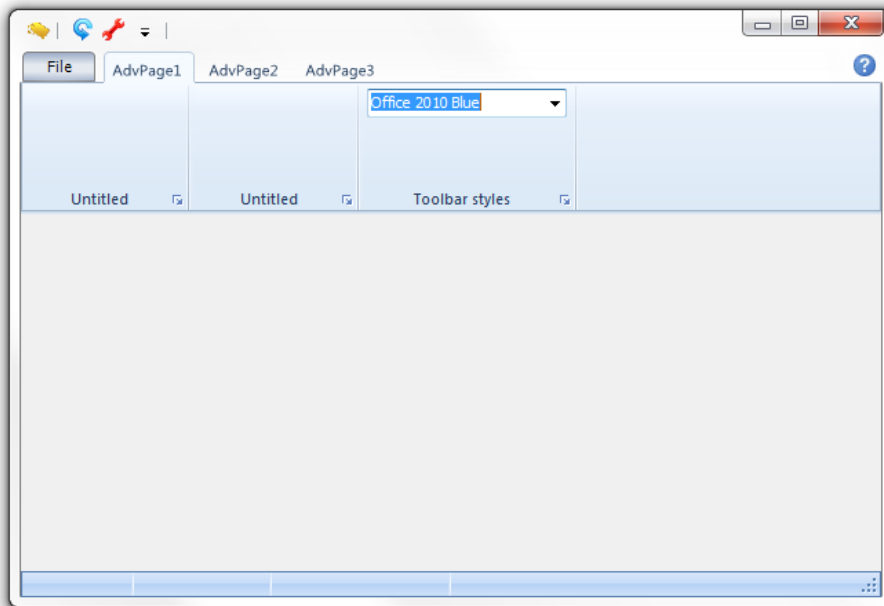
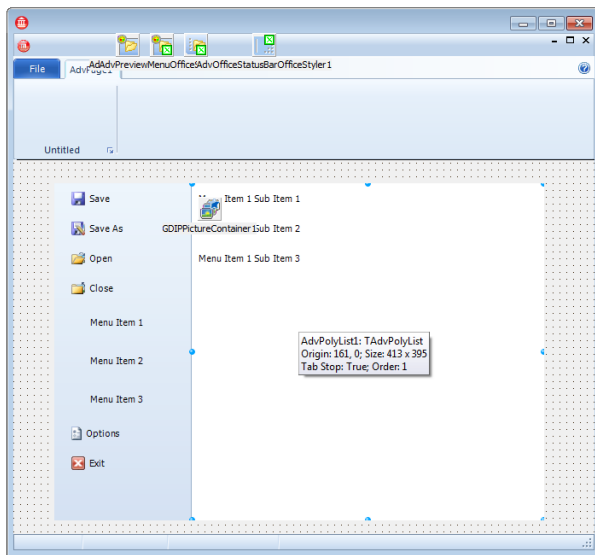Office 2007 ribbon
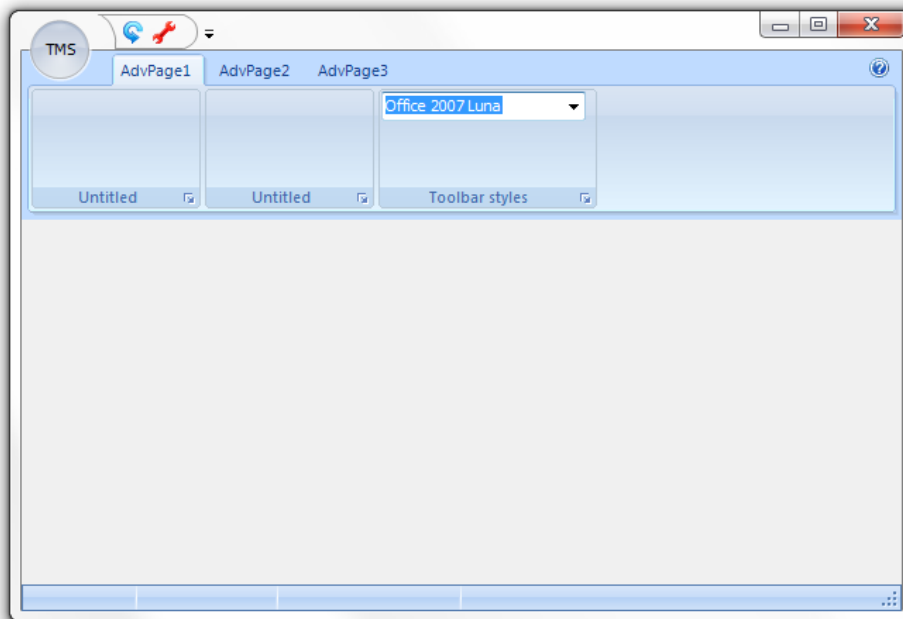
Windows 7 scenic ribbon
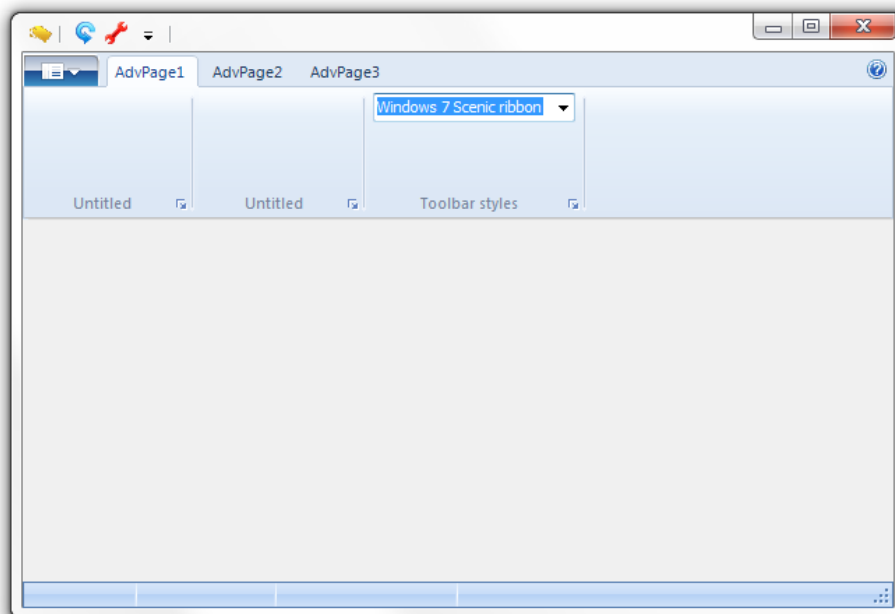
Office 2010 ribbon

**Using the new TAdvShapeButton & Poly List controls for Office 2010 style application menu**

The Office 2010 application menu is even more sophisticated than the Office 2007 one. It offers a complex hierarchy of menu and menu options. To meet the features of these various menus, we have created the Poly List controls. These controls can be considered as lists of polymorph items that can represent and do various actions. To create an Office 2010 application menu, a frame is created (this is done automatically if you use the wizard to create a new Office 2010 ribbon application) and this frame is assigned to the TAdvShapeButton. In Office 2010 mode, a click on the TAdvShapeButton will show the frame in the full form. The frame typically consists of 2 or more TAdvVerticalPolyList controls that contain all menu items and menu options. The wizard that creates an Office 2010 ribbon application automatically creates this frame with TAdvPolyMenu controls.
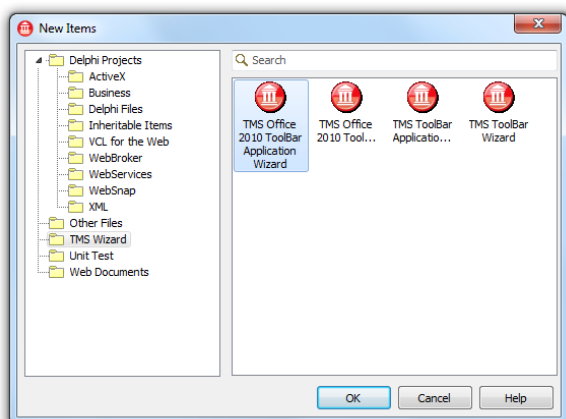
Selecting TMS Office 2010 ToolBar Application Wizard and clicking OK generates a complete application with Application Menu and the Office 2010 Blue style.

Starting the application and clicking the Application Menu Button, shows the Office 2010 style application menu that demonstrates how to switch pages:
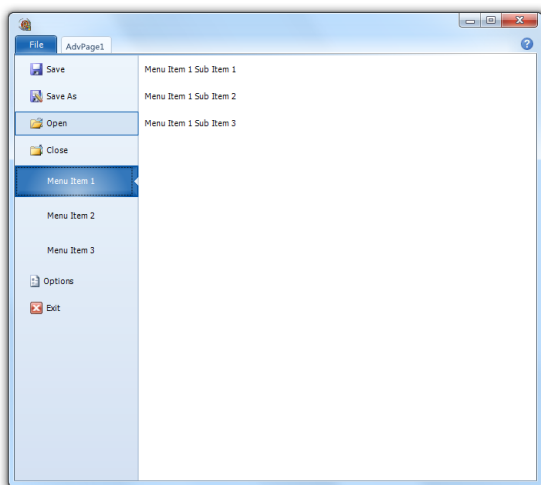
It is also possible to create the Office 2010 ribbon with application menu manually.

Add a new Frame to the application:
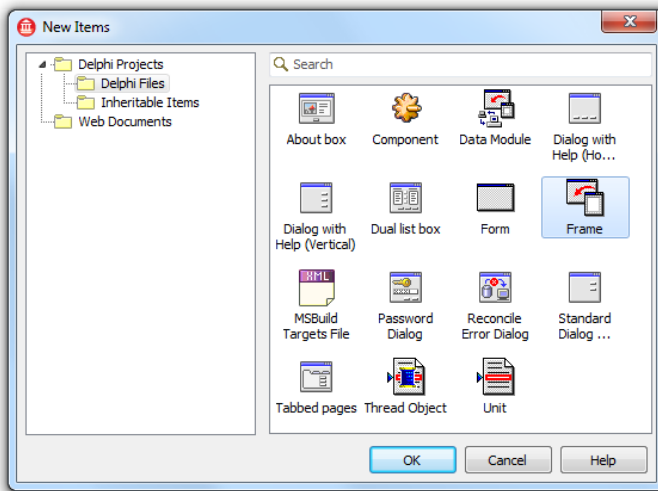
Add the frame to the main form and link the frame to the Application Menu Button:
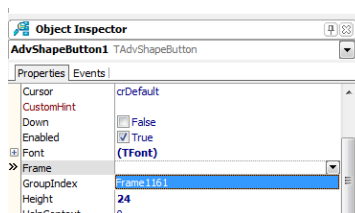
Starting the application and clicking the menu button will show an empty menu with the frame embedded. You will also notice a gradient line that is placed below the Application Menu button to provide a smoother look and feel.
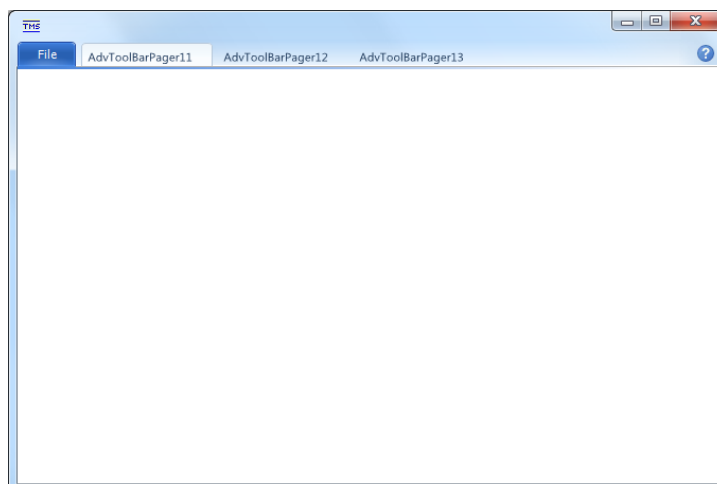
Designing the Application Menu can be done with standard VCL controls, but to provide a look and feel that is similar to the Office 2010 Application Menu, we have especially created the TMS Advanced Poly List controls.
One of the controls that can be used to create an Office 2010 style Application Menu is the TAdvPolyMenu component. This control presets some properties to be recognized as an Application Menu. When the property IsMainMenu is true the poly list will automatically adapt to the selected Office 2010 style.
Drop a TAdvPolyMenu control on the Frame, and set alignment to alLeft.
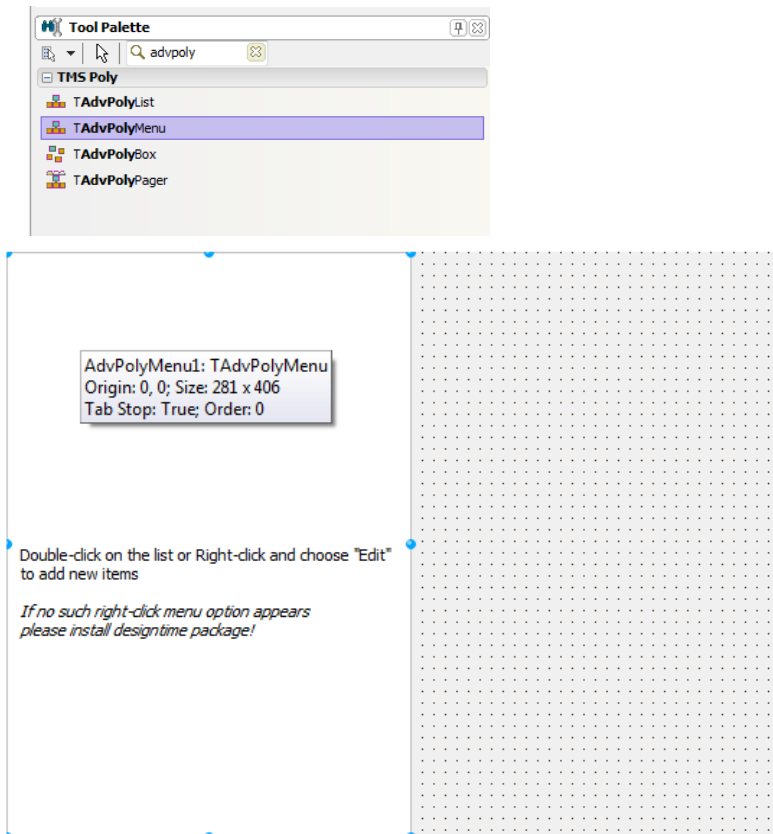
As the TAdvPolyMenu shows at designtime, double-click on the list to add new items. For this sample we have added 3 "normal text and image" items. For more details about all items that can be added to the Poly List controls, please see the TMS Advanced Poly List documentation. Starting the application and clicking on the Application Menu button shows the menu with the AdvPolyMenu control embedded.
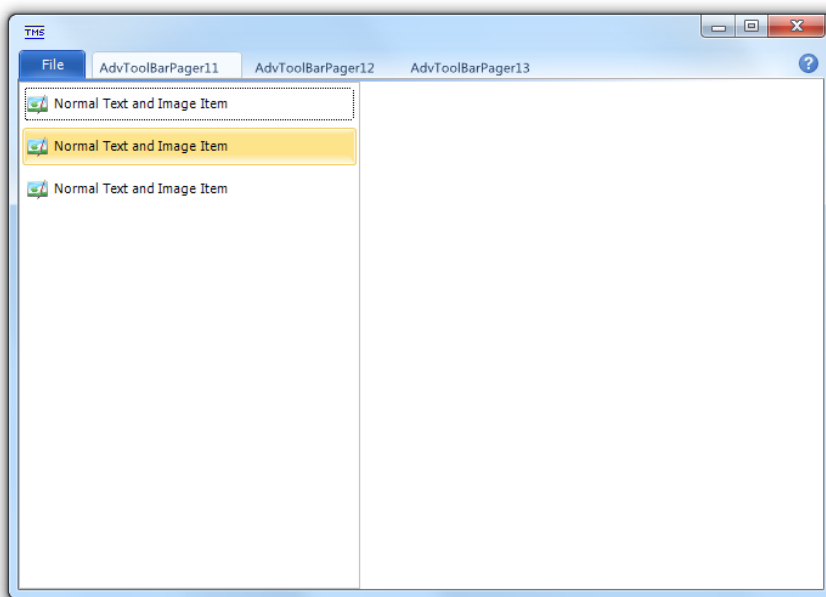

The empty space on the right can be used to display the content when clicking on the items. This content can again be any standard VCL control but is in most case more sophisticated. The TMS

Advanced Poly List controls contain 5 other lists that can be used to display such content:

- TAdvVerticalPolyList
- TAdvHorizontalPolyList
- TAdvPolyList
- TAdvPolyBox
- TAdvPolyPager

For this sample we have added 3 new TAdvVerticalPolyList controls on the frame (which always displays the items vertically). Also, we have set the AdvVerticalPolyList2 and AdvVerticalPolyList3 visible property to false. This will be explained below.
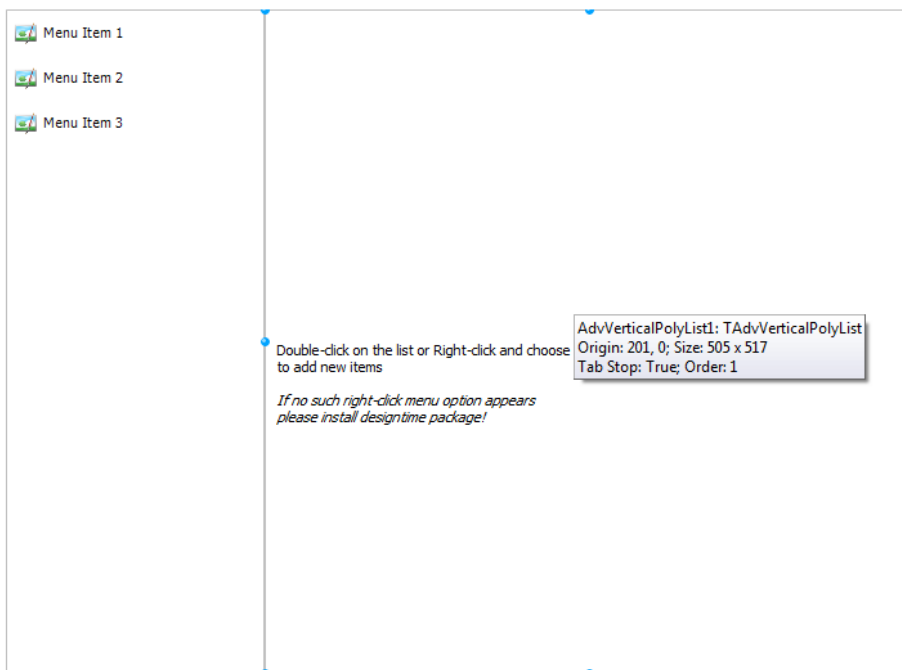
Displaying the pages can be done when selecting an item. Click on the TAdvPolyMenu control and implement the OnItemSelect event:
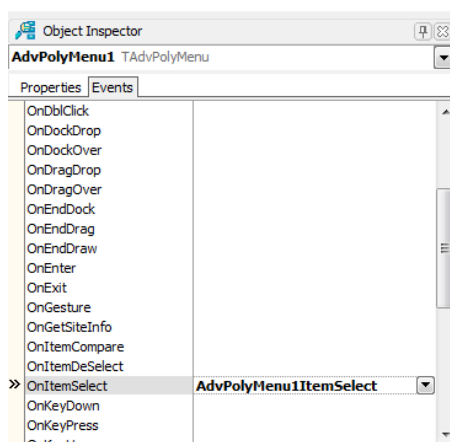

```
procedure TFrame116.AdvPolyMenu1ItemSelect(Sender: TObject; Item:
TCustomItem;
  var Allow: Boolean);
```

```
begin
  ChangePage(Item.Index);
end;
```

The ChangePage procedure changes the correct page to match the item selection:

```
procedure TFrame116.ChangePage(AIndex: Integer);
begin
  case AIndex of
    0:
    begin
      AdvVerticalPolyList1.Visible := True;
      AdvVerticalPolyList2.Visible := False;
      AdvVerticalPolyList3.Visible := False;
      AdvVerticalPolyList3.BringToFront;
    end;
    1:
    begin
      AdvVerticalPolyList1.Visible := False;
      AdvVerticalPolyList2.Visible := True;
      AdvVerticalPolyList3.Visible := False;
      AdvVerticalPolyList2.BringToFront;
    end;
    2:
    begin
      AdvVerticalPolyList1.Visible := False;
      AdvVerticalPolyList2.Visible := False;
      AdvVerticalPolyList3.Visible := True;
      AdvVerticalPolyList3.BringToFront;
    end;
  end;
end;
```

Now, more items can be added to the different lists in the frame. For list one we have added three checkitems, for the second list three buttonitems and three radioitems for the third list.
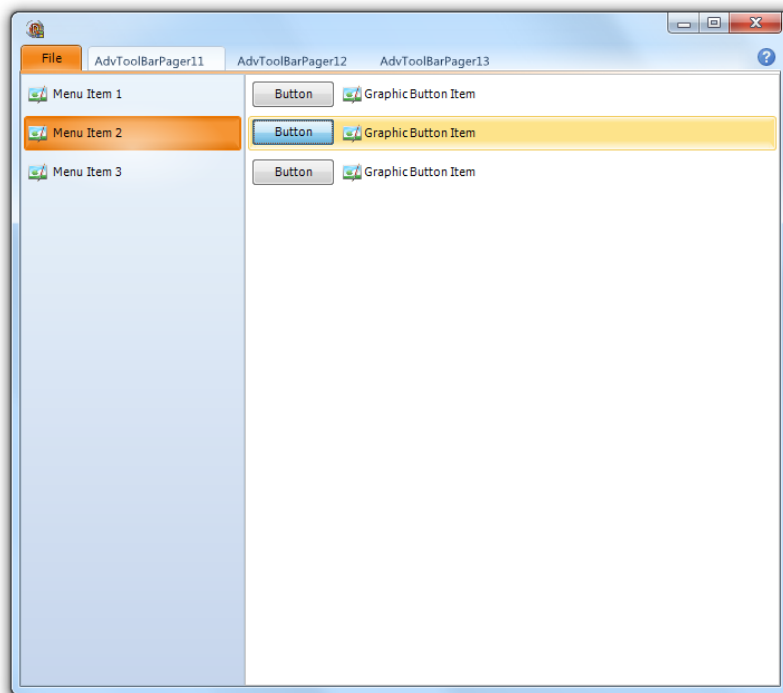
Selecting a different item will show the corresponding page. You can add as many controls as you wish.

Closing the menu at runtime can be done by clicking the Application Menu button, pressing Escape or clicking one of the pages in the toolbar. The Application Menu can also be closed in code. In this sample we have added an extra "normal text" item to the AdvPolyMenu which is non-selectable and closes the Application Menu.
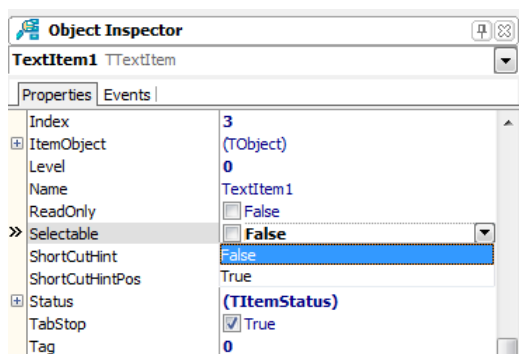
Selecting the item in the main form and implementing the OnItemClick event allows you to close the Application Menu with the hideframe procedure.
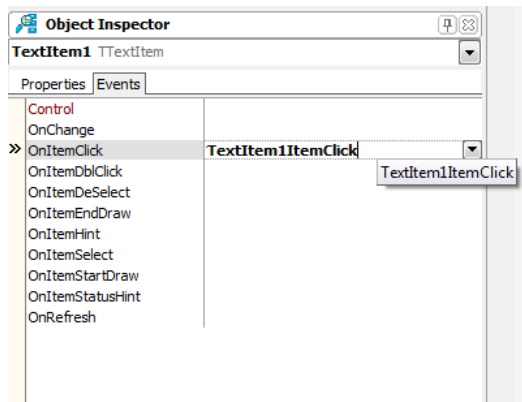
```
procedure TForm115.Frame1161TextItem1ItemClick(Sender: TObject;
  Item: TCustomItem);
begin
  AdvShapeButton1.HideFrame;
end;
```

Choosing the application menu color

In Office 2010, the different applications have a different application menu button color and left menu color for easy recognition what application is used (Orange for Outlook, green for Excel, blue for Word). TMS Advanced ToolBars & Menus also makes it easy to choose a color for application menu button and application menu. To change to color you can use the built-in color picker from the application menu button context menu by choosing "Change Color".
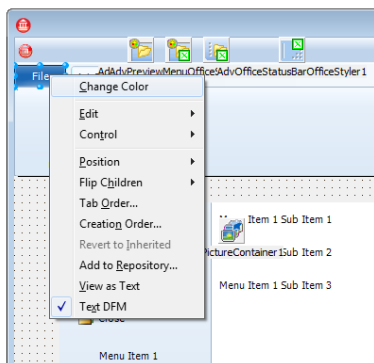
Setting a different color changes the menu item appearance to match the color of the button.
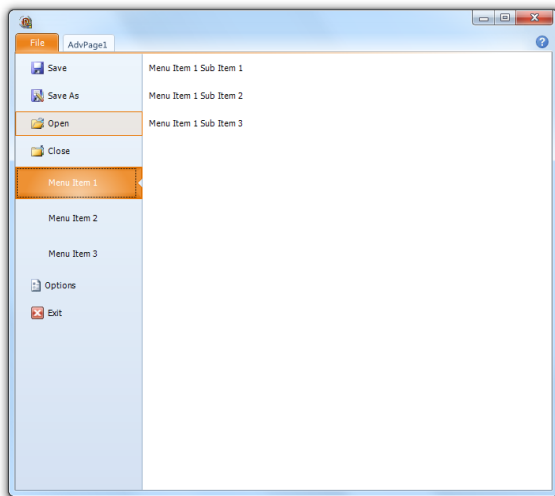
Changing the color can also be done in code, the global appearance of the button can be changed with the Color property if the property UseGlobalColor is true.

```
AdvShapeButton1.UseGlobalColor := True;
AdvShapeButton1.Appearance.Color := clGreen;
```

When the frame is assigned with the AdvPolyMenu implemented, the application will automatically update the item appearance to match the color of the menu button: