Popup-Box Component for Delphi
Version 1.0

Contents

---

1. Introduction

The PopupBox-Component is a component for Borlands Delphi and a replacement for the ComboBox with style csDropDownList. Because I am altogether dissatisfied with the combobox, it's look and functionality, I made the decision to write my own component. And here it is!
The PopupBox looks like an Edit- or Memofield, but it is not editable. If you click on it, a list opens up. You may select one of the list-entries by an other Mouseclick or by pressing the RETURN key. This entry will then become the current Text of the PopupBox-field. Optional the next window-control can be automatically focused. But if you don't like the way, PopupBox handles RETURN / MouseClicks, you can simply control this behaviour in Delphis Objectinspector with the properties *GoAwayOnClick* and *GoAwayOnReturn*. The PopupBox then will stay open until you move to an other window-control (by pressing the TAB key or clicking at an other field).
I guess it's a little bit difficult to understand how it works, but I think it's clear at once if you see the SAMPLE program, included in this package. Before you can compile and run this sample program you must install the PopupBox as a Delphi component. Look at section 3 - Installation.
PopupBox has a few settings, wich can change it's style. They are listed at section 5 - *PopupBox properties .*

The PopupBox is much more resource friendly than Delphis ComboBox. Every ComboBox in your application creates a (hidden) listwindow, wich is only unhidden, when the combolist is shown. These hidden windows let go down your rare windows system ressources. But the PopupBox creates only **ONE** listwindow. And the listwindow exists only as long as the list is shown. It will be destroyed as soon as the list is closed.

Using condition:
You can use the PopupBox-Component free, but won't see the source code, because it's not part of this package. If you need (or would like to have) the source-code of this component, you must register it. The registration fee is only $10.00 (or 15,00 DM).

How to register?
If you have a CompuServe account, you can register with GO SWREG . Look for #6032.

The other possibility is to send me the money cash to:

Jan - M. Strube
Breslauer Str. 19
70806 Kornwestheim
GERMANY

I will **e-mail** you immediately the source code of the latest version, so don't forget to tell me your e-mail address!

2. Files

The following 8 files are part of this package, called POPUPBOX.ZIP.

EXAMPLE.DFM	Example program (form)
EXAMPLE.DPR	Example program (project)
EXAMPLE1.DFM	Example program (form)
EXAMPLE1.PAS	Example program (example-code)
MAINCLSS.DCU	MainClass-Components
POPUPBOX.DCR	PopupBox-component icon
POPUPBOX.DCU	PopupBox-component
README.WRI	this README file (Write format)

3. Installation

Before you can try out the example program, you must install the PopupBox component into the Delphi palette and rebuild the component library.

I) Copy **MAINCLSS.DCU**, **POPUPBOX.DCR** and **POPUPBOX.DCU** to a directory, where your components reside, for instance C:\DELPHI\LIB .

II) From within Delphi, select from the main menu "**Options**" -> "**Install components...**". A Dialog "Install components" opens up.

III) Select "**Add**" from the available buttons at the right of the dialog, enter **POPUPBOX** and select the **OK** button.

IV) Before closing the "Install components" dialog, check if the path name, where POPUPBOX.DCR and POPUPBOX.DCU were copied to, is part of the "Search path" edit control.

V) Select **OK** to have Delphi compile and add the PopupBox component to the component library. When installation is complete, a new item "PopupBox" can be found at palette tab Standard.

4. Sample program

Within this package comes a sample program **EXAMPLE.DPR** . It shows, how to use the PopupBox and demonstrates it's main properties. Open the project EXAMPLE.DPR and select menu "**Run**" -> "**Run**" .

5. PopupBox properties

PopupBox is a two part component. There is:
A) the **PoupBox-Field** and
B) the **PopupBox-List** .

In Delphis Designmode only A) is visible. Whereas at runtime you can see A) and B).
Please note, that at runtime...
...you will not see B) until A) got the focus.
...B) is hidden (and destroyed) as soon as A) looses the focus.

Properties of A) PopupBox-Field:

Alignment	
	{taLeftJustify, taRightJustify, taCenter} controls the Textalignment of the field
AutoSize	
BorderStyle	
CharCase	
Color	
Ctl3D	
Cursor	
DragMode	
Enabled	
Font	
GoAwayOnClick	
	{True, False} **GoAwayOnClick = True:** the next window control is focused, when you click on a listentry **GoAwayOnClick = False:** the next window control will not be focused, when you click on a listentry
GoAwayOnReturn	
	{True, False} **GoAwayOnReturn = True:** the next window control is focused, when you press the Return-key **GoAwayOnReturn = False:** the next window control will not be focused, when you press the Return-key
Height	
HelpContext	
Hint	
Left	
MaxLength	
Name	
OEMConvert	
ParentColor	
ParentCtl3D	
ParentFont	

ParentShowHint
PopupMenu
ReadOnly
ShowHint
TabOrder
Tag
Text
Top
Visible
Width

Properties of B)   PopupBox-List:

ListAlignment

Type: TAlignment
{taLeftJustify, taRightJustify, taCenter}
controls the Textalignment of the Listentries

ListAutoPos

Type: Boolean
{True, False}

True: when the list is opened, that listentry is automatically selcted in the list, thats the current text of PopupBox-Field
False: when the list is opened, the first lisentry is selected

Tip: To select manually a specific listentry, make a call in the event OnEnter like: Popupbox1.List.ItemIndex:= 3;

ListBorderStyle

Type: TBorderStyle
{bsSingle, bsNone}
controls the Listborder

ListColor

Type: TColor
{0..$FFFFFF}
controls the Listcolor (backgound)

ListFont

Type: TFont
{FontColor, FontHeight, FontName, FontStyle}
controls the Listfont

Note: The ListItemHeight automatically resets, if ListFont changes! Thatswhy first select the font and *then* type in the ListItemHeight

ListHeight

Type: Integer
{0..32767}
is the ListHeight

ListHint

Type: String
controls the Hinttext, when the List is open. While the List is closed,

Hint from PopupBox-Field is the Hinttext

Note: You will see the ListHint only, if **ShowHint** from PopupBox-Field is set to True!

ListIntegralHeight

Type: Boolean
{True, False}
controls the way the list represents itself on the form. If ListIntegralHeight is True, the list shows only entries that fit completely in the vertical space, and the bottom of the list moves up to the bottom of the last completely drawn item in the list. If ListIntegralHeight is False, the bottom of the list is at the location determined by its ItemHeight property, and the bottom item visible in the list might not be complete.

ListItemHeight

Type: Integer
{0..32767}
is the height of an item in the list in pixels

ListItems

Type: TStringList
is the list of StringItems, you see in the list.
Note: All properties and methods of TStringList can be used. Look at Delphi's Online-Help under TStringList.

Examples:

```
   To add a new Item to the list call:
   PopupBox1.ListItems.Add('TestString');

   To remove all Items from the list call:
   PopupBox1.ListItems.Clear;

   To load a file into the list call:
   PopupBox1.ListItems.LoadfromFile('C:\WINDOWS\WIN.INI');

   To get the ListItems-count call:
   MessageDlg(IntToStr(PopupBox1.ListItems.Count),
      mtInformation,
      [mbOK], 0);
```

ListLeft

Type: Integer
{-32768..32767}
is the distance between PopupBox-Field.Left and the left side of the PopupBox-List

ListSel3D

Type: Boolean
{True, False}
if True the selected ListItem is painted in 3D-style

ListSelBGColor

Type: TColor
{0..$FFFFFF}

controls the backgroud color of the selected ListItem

ListSelColor

Type: TColor
{0..$FFFFFF}
controls the foreground(Text-) color of the selected ListItem

ListSorted

Type: Boolean
{True, False}
indicates whether the items in the list are arranged alphabetically. To sort the items, set the ListSorted value to True. If ListSorted is False, the items are unsorted.
If you add or insert items when ListSorted is True, PopupBox automatically places them in alphabetical order.

ListTop

Type: Integer
{-32768..32767}
is the distance between PopupBox-Field.Top and the Top side of the PopupBox-List

ListWidth

Type: Integer
{0..32767}
is the ListWidth

List.TopIndex

Type: Integer
{0..32767}
At Runtime only and the list must be open!
List.TopIndex property is the index number of the item that appears at the top of the list. You can use the TopIndex property to determine which item is the first item displayed at the top of the list and to set it to the item of your choosing.

Example:
To make the 6th listentry the first displayed item make a call in the event OnEnter like:
if Popupbox1.List <> nil then Popupbox1.List.TopIndex:= 5;

6. PopupBox variable

There is only one variable of interrest, the **List** of type TPopupBoxListBox.
List is available only at Runtime and while the list is open!
List is a descendant of TListBox. So you can use all properties and methods of TListBox.

Example: When the mouse moves about a list-entry a label shows the list number the mouse is over. Event is OnListMouseMove.

```
procedure TForm1.PopupBox1ListMouseMove(Sender: TObject;
 Shift: TShiftState; X, Y: Integer);
VAR
 ListIndex: LongInt;
 ExistEntry: Boolean;
```

```
begin
 if PopupBox1.List <> nil then begin
   ListIndex:= PopupBox1.List.ItemAtPos(
                    Point(X, Y), ExistEntry);
   if not ExistEntry then exit;
   Label1.Caption:= IntToStr(ListIndex + 1);
 end;
end;
```

An other call could be at event OnEnter of PopupBox1. This call sets the Height of the list to such a value, that all entries fit in. (no scrollbar necessary)

```
 with Popupbox1.List do
      SetBounds(Left, Top, Width,
                PopupBox1.ListItems.Count * ItemHeight);
```

Note: The Items of the List are not in "**PopupBox1.List.Items**". They are in "**PopupBox1.ListItems**" !

7. PopupBox events

PopupBox is a two part component. There is:

A) the **PoupBox-Field** and
B) the **PopupBox-List** .

Events refer to A)

OnChange	-> text value of PopupBox-Field changed
OnClick	-> click at PopupBox-Field
OnDblClick	-> double click at PopupBox-Field
OnEnter	Here is a good place to do initializations for the list!
OnExit	Here is a good place to do cleanups at the list!
OnMouseDown	-> mouse down at PopupBox-Field
OnMouseMove	-> mouse moved over PopupBox-Field
OnMouseUp	-> mouse up at PopupBox-Field

Events refer to B)

OnKeyDown	-> key down
OnKeyPress	-> key pressed
OnKeyUp	-> key up
OnListClick	-> clicks at the list
OnListMouseMove	-> mouse moved over the list
OnDrawListItem	-> necessary for owner draw PopupBoxes

Examples:

OnEnter:

```
procedure TForm1.PopupBox1Enter(Sender: TObject);
VAR
 y, month, day: Word;
begin
```

```
  Popupbox1.ListItems.Add('Value1');
  Popupbox1.ListItems.Add('Value2');
  Popupbox1.ListItems.Add('Value3');

  DecodeDate(Date, y, month, day);
  if (day = 24) and (month = 12) then
    Popupbox1.ListItems.Add('Happy christmas!');

  if (day = 31) and (month = 12) then
    Popupbox1.ListItems.Add('Happy New Year!');
end;
```

OnExit:

```
procedure TForm1.PopupBox1Exit(Sender: TObject);
begin
  Popupbox1.ListItems.Clear;
end;
```

Ownerdraw PopupBoxes

It's easy to make an ownerdraw PopupBox, because the most is automatically painted. Only the Items itself still are to paint. These paint calls must be placed at the event OnDrawListItem. The Popupbox becomes automatically an owner draw list, if something is placed within the event OnDrawListItem.

Here is a simple example, that allows to write multiple lines at one list entry. The property ListItemHeight must be high enough (something like 60 or so).

```
procedure TForm1.PopupBox1DrawListItem(Control: TWinControl;
  Index: Integer; Rect: TRect; State: TOwnerDrawState);
VAR
  TempBuf: Array[0..255] of Char;
begin
  (Control as TPopupBoxListBox).Canvas.FillRect(Rect);
  StrPCopy(TempBuf, PopupBox1.ListItems[Index]);
  DrawText((Control as TPopupBoxListBox).Canvas.Handle,
           TempBuf, -1, Rect, DT_WORDBREAK);
end;
```

Please tell me what you think about the PopupBox. Please tell me also if you find any bugs, so that I can hopefully remove them quickly. Thank you!

Jan - M. Strube
Breslauer Str. 19
70806 Kornwestheim
GERMANY

e-mail:

CompuServe:	Jan Strube 100333.2744
Internet:	664104@rz.fht-esslingen.de