TurboDelphi for Win32

Titel: How to install and use components in Turbo Delphi (Win32)

How to install and use components in Turbo Delphi (Win32)

A few days ago, Borland/DTG published TurboDelphi, and just like me, many people downloaded and installed it. The biggest deficiency of the free Explorer version is that it does now allow to install design time packages (which integrate into the IDE and can be dragged onto a form). However, it is possible to integrate design time packages into the IDE, and this tutorial shows how to do it. Special thanks to Elvis, without his statements the second part of the tutorial would not be possible.

I'm using the components "TScrollingCredits" (which you can find in the rar archive too) and TGauge, which is already included in TurboDelphi.

1. How to use components non-visual

In general we can't integrate components into the IDE and work with them. But who cares? We can still create the components at run-time directly in the code.

1. Most components are in directly pascal sourcecode, which means we have the *.pas-file, like in our example TScrollingCredits. Copy the unit *credits.pas* to the TurboDelphi-library folder (for example X:\Program Files\Borland\BDS\4.0\lib) or just create a new library path in the Delphi options.

2. In the next step start Delphi and create a new VCL-Forms-Application. Delphi doesn't know in which unit our component is, thus we have to take the unit *credits.pas* into the uses-list. Here's an example:

Delphi-Quellcode: markieren

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, Credits;
```

3. Every component is just an instance of a class, in our example it's the class "TScrollingCredits", which you can find in the unit *credits.pas*. So we have to - in case we want to create a new component - derive a new instance from the TScrollingCredits - class. For that go to the OnCreate event of the main form, so that our component will be created after the mainform was created. To derive a new component just handle them like a

variable, that means just create a new instance of the class TScrollingCredits.

Delphi-Quellcode: markieren

```
procedure TForm1.FormCreate(Sender: TObject);
var
  MyCredits : TScrollingCredits; //Componentname & the class
/ type
begin

end;
```

4. Every component has the .*Create* method, which we have to call. It must be considered that we don't call the .Create method of our component MyCredits, because MyCredits doesn't exists in this time. We have to call the .Create method of the class TScrollingCredits, only this will create the component. Possibly some component expects types as parameter a *AOwner* as delivery value. This isn't important in this short tutorial, just use the parameter *self* or another control of TComponent.

Delphi-Quellcode: markieren

```
MyCredits := TScrollingCredits.Create(self);
```

5. When you compile and build the project, you won't see our component. Thats because Delphi doesn't know how and where it has to show the component. Quick & Dirty solution:

Delphi-Quellcode: markieren

```
with MyCredits do
  begin
    Parent := Form1; //Set parent
    Height := 100; //Set other important properties
    Width := 200;
    Top := 20;
    Left := 20;
  end;
```

6. By creating and compiling the project we're almost at the end, our component is lying good on the form and works perfect. Of course, more adjustments are possible via source, options can be done and everything should be like normally in the designer. "Everything is possible", comes to my mind. All in all, you can say that we replaced the objectinspector by the codeeditor.
Here the complete sourcecode of the project, for all of you being too lazy to write or read.

Delphi-Quellcode: aufklappen | markieren

```
unit Unit1;

interface

```

```
uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, Credits;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
var
  MyCredits : TScrollingCredits;
begin
  MyCredits := TScrollingCredits.Create(self);
  with MyCredits do
  begin
    Parent := Form1;
    Height := 100;
    Width := 200;
    Top := 20;
    Left := 20;
  end;
end;

end.
```

2. Integrate Components with the IDE

The headline already suggests it - though it should not be possible in the free version of Turbo Delphi, it is possible.
I do not know if Elvis found it himself, but I am thankful he notified us today. 🙂

1. So let's go: As I had this problem myself, we'll care for the component TGauge to be

added into the IDE, which gets shipped with Turbo Delphi, but strangely does not get integrated with the IDE. Fact is: Turbo Delphi does not accept 3rd party design packages and will deny their installation. But there is a package - which has been there going along with us since the old days - which we are (technically) allowed to edit and install. It is the package **dclusr.dpk**, which you will find in *X:\Program Files\Borland\BDS\4.0\lib* per default. So let's fire up TurboDelphi and open said dclusr.dpk.

2. Now we focus the "project management" window, where we have to add the respective unit to the package. In order to accomplish this, we right-click **dclusr100.bpl** and choose "Add...". In the window opening subsequently we click "Browse...", choose the respective unit (for our behalves gauges.pas, located in *X:\Program Files\Borland\BDS\4.0\source\Win32\Samples\Source*) and click "Ok". The unit will be added to the Package and we can see it in the *"Contains"* key.

3. Click through the folders in the *"Contains"* key until you can see the unit and open it with a double click. It will open into the code editor and we can see the source code. In our example there is no register-procedure, if you have one, skip this step! As there is none in gauges.pas, we have to create our own. In order to do so, we declare the procedure

Delphi-Quellcode: markieren

```
procedure Register;
```

after all other declarations in the interface section. In the implementation section we add this:

Delphi-Quellcode: markieren

```
procedure Register;
begin
RegisterComponents('Samples', [TGauge]);
end;
```

The string "Samples" passes the name of the tab/section the compoent will be placed later-on. If it does not exist just yet, it will be added autmatically. In the array ([TGauge]) you have to pass the classes of all components in the units you want to register. Another example: [TScrollingCredits].
The head of gauges.pas should look like this by now:

⊞**Delphi-Quellcode:** aufklappen | markieren

```
unit Gauges;

interface

uses SysUtils, Windows, Messages, Classes, Graphics,
Controls, Forms, StdCtrls;

type

  TGaugeKind = (gkText, gkHorizontalBar, gkVerticalBar,
gkPie, gkNeedle);

  TGauge = class(TGraphicControl)
  private
    FMinValue: Longint;
    FMaxValue: Longint;
    FCurValue: Longint;
    FKind: TGaugeKind;
    FShowText: Boolean;
    FBorderStyle: TBorderStyle;
    FForeColor: TColor;
    FBackColor: TColor;
    procedure PaintBackground(AnImage: TBitmap);
    procedure PaintAsText(AnImage: TBitmap; PaintRect:
TRect);
    procedure PaintAsNothing(AnImage: TBitmap;
PaintRect: TRect);
    procedure PaintAsBar(AnImage: TBitmap; PaintRect:
TRect);
    procedure PaintAsPie(AnImage: TBitmap; PaintRect:
TRect);
    procedure PaintAsNeedle(AnImage: TBitmap; PaintRect:
TRect);
    procedure SetGaugeKind(Value: TGaugeKind);
    procedure SetShowText(Value: Boolean);
    procedure SetBorderStyle(Value: TBorderStyle);
    procedure SetForeColor(Value: TColor);
    procedure SetBackColor(Value: TColor);
    procedure SetMinValue(Value: Longint);
    procedure SetMaxValue(Value: Longint);
    procedure SetProgress(Value: Longint);
    function GetPercentDone: Longint;
  protected
    procedure Paint; override;
  public
    constructor Create(AOwner: TComponent); override;
    procedure AddProgress(Value: Longint);
    property PercentDone: Longint read GetPercentDone;
  published
    property Align;
    property Anchors;
    property BackColor: TColor read FBackColor write
SetBackColor default clWhite;
    property BorderStyle: TBorderStyle read FBorderStyle
write SetBorderStyle default bsSingle;
    property Color;
    property Constraints;
    property Enabled;
    property ForeColor: TColor read FForeColor write
SetForeColor default clBlack;
    property Font;
    property Kind: TGaugeKind read FKind write
SetGaugeKind default gkHorizontalBar;
    property MinValue: Longint read FMinValue write
SetMinValue default 0;
    property MaxValue: Longint read FMaxValue write
SetMaxValue default 100;
    property ParentColor;
    property ParentFont;
    property ParentShowHint;
    property PopupMenu;
    property Progress: Longint read FCurValue write
SetProgress;
    property ShowHint;
    property ShowText: Boolean read FShowText write
SetShowText default True;
    property Visible;
  end;

procedure Register; //Create!

implementation

uses Consts;

//Create!
procedure Register;
begin
RegisterComponents('Samples', [TGauge]);
end;

type
  TBltBitmap = class(TBitmap)
    procedure MakeLike(ATemplate: TBitmap);
  end;

{ TBltBitmap }

procedure TBltBitmap.MakeLike(ATemplate: TBitmap);
begin
  Width := ATemplate.Width;
  Height := ATemplate.Height;
  Canvas.Brush.Color := clWindowFrame;
  Canvas.Brush.Style := bsSolid;
  Canvas.FillRect(Rect(0, 0, Width, Height));
end;
```

4. Via the Ctrl+S shortcut we save the changes we made. We're not far from the goal anymore, just change back into the project management window and right-click **dclusr100.bpl**. Now we choose *"Compile"* and then, after successful compilation, *"Install"*. A messagebox should now appear notifying us of all the new components installed. After restarting Turbo Delphi you should now find the "3rd party component" inside your IDE, patiently waiting for its deployment.

Important Notes:
- It has not yet been tested how Delphi will behave if there are a lot of components installed into dclusr.dpk, so issues may arise. The author has been integrating 7 Components so far for testing purposes and nothing went wrong up until now. If you have any experience about how the package does when there are a lot of components installed, please contact the author.
- If you get messages like "the package could not be deinstalled" dont be worry, just ignore them and after a restart of TurboDelphi the component would be installed successfull to the IDE

I hope this tutorial is helpful for you in any way. 😉

Please note: Against all denying voices the second part of this tutorial does not - according to the author's opinion - in any way violate the license agreements of Turbo Delphi, as a) no protection mechanisms get disabled, broken, or circumvented and b) no packages are installed, as this is forbidden. The author does not do this but install the compoeenents into an existing package.

Tutorial: How to install and use components in Turbo Delphi (Win32)
Author: Pierre (alias "Balu der Bär")
Version: 1.0b (10.09.2006)
Contact: turbodelphi@list.ru
Special thanks to Daniel G(ilbert)
Translation of 2nd part: Lukas Erlacher