Tips & Tricks

QuickReports Images And Dots

QuickReports lets you drop a Delphi standard `TImage` on a band and then draw on it, once the `QRPrinter` object is initialized. The most peculiar thing I found about using such an image was that on my dot matrix printer it looked fine, but on my customer's color laser printer it caught a case of pink chicken pox, despite explicitly setting the color of the `brush` and doing everything I could think of, initially, to ensure I had a white background. The solution turned out to be building a device independent bitmap first:

```
mybmp := tbitmap.create;
with mybmp do begin
  height := ImageOnReport.height;
  width := ImageOnReport.width;
  monochrome := true;
end;
ImageOnReport.picture.bitmap := mybmp;
with ImageOnReport.canvas do begin
  {... do all the drawing ...}
end;
```

Contributed by Brandon Smith, synature@aol.com

QuickReports Generalized Report Control

I found I was replicating code on each one of my QuickReports forms and I was also not real thrilled with showing the user the form with bands and other items that had nothing to do with the final report. When the user clicks on a report he or she wants one of two things: either a preview on screen or a printed report. So I came up with a form unit to wrap around my reports which I could drop into the Object Repository and then tweak for each new project as needed.

This *Report Preface* unit, in its initial incarnation, was designed to allow the user to specify whether a report was to include all records or just one, to specify a date other than today to appear on the report, to select another font, to fiddle with the printer setup (change printers, whatever), to preview or to print the report. To support using this report wrapper, I set up each QuickReports form to be called by a procedure rather than invoking the form directly. This approach also allows me to customize each report for that particular call. For example, most of my report calling procedure declarations look like this:

```
Procedure DoReport(DoPrint : boolean);
```

but I've one where I let the user select which columns to include in the report, so the calling procedure is declared as

```
Procedure DoChoiceReport(DoPrint : boolean;
  WhichCols : tstringlist);
```

Implementing each of these report control procedures involves something along these lines:

```
begin
  QR_ReportForm :=
    tQR_ReportForm.create(application);
  with QR_ReportForm do begin
    { do whatever intializations are required for
      this report}
    with QuickReport1 do begin
      { Any special tweaks to the QR component
        itself, such as }
      showProgress := true;
      if DoPrint
        then Print
        else Preview;
      end;
    end;
  QR_ReportForm.release;
end;
```

Calling on one of these reports usually looks like this:

```
procedure TF_Main.SomeReport1Click(
  Sender: TObject);
begin
  f_reportPreface :=
    tf_reportPreface.create(application);
  f_reportPreface.reportName := 'Doctor''s Visit';
  case f_reportPreface.showModal of
    mrOK : DocVisRpt(true);
    mrYes : DocVisRpt(false);
  end;
  f_reportPreface.hide;
  f_reportPreface.free;
end;
```

Contributed by Brandon Smith, synature@aol.com

Delphi 3 Hints Bug

Try to compile the simple project comprising the three files in Listing 1. You will get two hint messages saying *Hint: D:\SRC\t1u.pas(-2): Private symbol 'Unused1' declared but never used* and *Hint: D:\SRC\t1u.pas(-1): Private symbol 'Unused2' declared but never used*. Notice that the line numbers for the messages are all wrong (negative). It seems that the hint messages get confused because of the include file.

If you now try to recompile, Delphi 3 will give an access violation. You can now no longer compile anything, you will have to restart Delphi. Usually you will be able to do `File|Save all`. If you try `File|Close all` you will get the same access violation and it will be impossible to close down Delphi politely (Task Manager's `End Task` command will still work). Trying to

```
T1.DPR:
program t1;
uses t1u;
end.

T1U.PAS:
{$A+,B-,C+,D+,E-,F-,G+,H+,I+,J+,K-,L+,M-,N+,O+,P+,Q-,R-,S-,
 T-,U-, ,V+,W-,X+,Y-,Z1}
unit t1u;
interface
{$I SIMPLE.INC}
type
  TSimpleClass = class
  private
    Unused1: integer;
    Unused2: integer;
  end;
implementation
end.

SIMPLE.INC:
{1
2
3
4
5
6
7
8
9}
```

➤ *Listing 1*

work in the IDE gives lots of access violations again. When you close down Delphi you will get a further set of errors before it finally dies and you can restart it.

There seem to be four workarounds. Firstly, don't use include files. Secondly, don't ever declare private fields or methods that are actually not used. Thirdly, turn off hints, or lastly, maybe compile using DCC32.EXE instead of the IDE.

Contributed by Hallvard Vassbotn, hallvard@falcon.no

Delphi 2 & 3 Hints And Messages Bug

There are bugs in Delphi 2 and 3 that relate to the "localness" of the `$HINTS` and `$WARNINGS` directives. These directives do not have the same scope as other local directives like `$ALIGN`. The three files in Listing 2 demonstrate the problem.

Whenever the compiler recompiles the `LocBugB` unit, `HINTS` and `WARNINGS` are turned on and kept on from the point the unit is used in other units. The same does not happen with the `ALIGN` directive. So the "(sometimes)" in the comment in the listing refers to whenever the `LocBugB` unit is recompiled. This means that to make sure `HINTS` and `WARNINGS` are turned off in a given unit you have to include `{$HINTS OFF}` and `{$WARNINGS OFF}` directives after the `uses` clause in the `interface` section *and* after the `uses` clause in the `implementation` section. This bug might seem innocent, but in combination with the other one (described above) is very annoying.

There is also another related shortcoming. Often a unit needs to turn off hints and/or warnings for a small section of code. In the ideal case, this should be done without affecting the initial state of the `HINTS` directive. You would think that the `$IFOPT` directive could be used to achieve this, but regretfully that is not so. The `$IFOPT` directive only supports testing of single-letter directives and there are no single letter variants of the `HINTS` and `WARNINGS` directives. This means that you cannot write:

```
LOCBUG.DPR:
program LocBug;
uses LocBugA;
begin
end.

LOCBUGA.PAS:
unit LocBugA;
interface
{$HINTS OFF} {$WARNINGS OFF} {$ALIGN OFF}
type
  TObjectA = class
  private
    UnusedA: integer;  // Should not get hint, we don't
  public
    procedure Destroy; // Should not get warning, we don't
  end;
  TRecA = record // SizeOf(TRecA) should be 5, and it is
    A: char;
    L: longint;
  end;
implementation
uses LocBugB;
{.$HINTS OFF}       // Remove dots to work around buggy behavior
{.$WARNINGS OFF}
type
  TObjectB = class
  private
    { Shouldn't get hint, but we do in D3 (sometimes) }
    UnusedB: integer;
  public
    { Should not get warning, but we do (sometimes) }
    procedure Destroy;
  end;
  { If you remove the dot, you will get an invalid
    typecast below }
  {.$ALIGN ON}
  TRecB = record // SizeOf(TRecA) should be 5, and it is
    A: char;
    L: longint;
  end;
procedure TObjectA.Destroy; begin end;
procedure TObjectB.Destroy;
var A: TRecA;
begin
  // This is Ok because SizeOf(TRecA) = SizeOf(TRecB)
  TRecB(A).L := 123;
end;
end.

LOCBUGB.PAS:
unit LocBugB;
interface
implementation
{$HINTS ON} {$WARNINGS ON} {$ALIGN ON}
end.
```

➤ *Listing 2*

```
{$IFDEF _HINTS_ON_} Error: Define name clash!
{$ENDIF}
{$IFOPT HINTS ON} {$HINTS OFF}
{$DEFINE _HINTS_ON_} {$ENDIF}
...
{Hint-disabled code here}
{$IFDEF _HINTS_ON_} {$HINTS ON}
{$UNDEF _HINTS_ON_} @CODE 8ON12PT = {$ENDIF}
```

This is admittedly ugly looking code, but is the way we have handled local directives back to Turbo Pascal days. It would be handy if Borland could add `{$PUSHOPT}` and `{$POPOPT}` directives to help in these situations.

Contributed by Hallvard Vassbotn, hallvard@falcon.no

Memo Line And Column

To get the current line and column from a memo or `RichEdit` control use:

```
Memo1.Line := Perform(EM_LINEFROMCHAR,SelStart, 0);
Memo1.Column := SelStart-Perform(EM_LINEINDEX,Line,0);
```

Contributed by Bruno Sonnino, bsonnino@geocities.com