The Unofficial Newsletter of Delphi Users

March 13th, 1995

by Robert Vivrette

CIS: 76416,1373

I would like to welcome all of you to the very first issue of *The Unofficial Newsletter of Delphi Users*. It has been only a few short weeks since I received my final version of Delphi and I must say, I dont think I have ever been as excited about a development platform as I am today.

I am writing this newsletter initially to achieve the following results:

1. Disseminate my discoveries and ideas about Delphi
2. Share some programming tips and techniques.
3. Act as a forum for Delphi interest.

First, I want to make it clear that I do not consider myself a Delphi expert. However, I am very fluent at writing Windows programs using BP7 and am making rapid progress at getting proficient with Delphi. As a result I think that you will find the general thrust of this newsletter will be from the angle of a *BP7 Windows Programmer Moving To Delphi*.

I also do not claim that everything I say here is gospel. If I make a wrong assumption about something, please let me know, and I will let everyone know how stupid I am in the next issue.

Contents

What You Can Do To Help

Ideally, this newsletter is for everyone to use. If you have something that you think others would benefit from, be it a neat idea, programming tip or trick, piece of elegant code, bug report (No! Never!), or whatever strikes your fancy, then I encourage you to forward it to me at 76416,1373. Because of its electronic nature, this newsletter can be as large as necessary to accommodate any print-worthy information its readers may submit. Actually, the more assistance I get from all of you, the easier it will be for me!
Please also keep in mind that this is a completely volunteer effort on my part. The newsletter will be free of charge to anyone who wants it. There will be no money changing hands in any form, so no advertising, no compensation for articles submitted, etc.

I am going to try a publishing schedule of every two weeks, time permitting. The style will be quite casual, so please excuse any lack of eloquence on my part.

Contents

Component Design

Component Design is one of the most fascinating aspects of Delphi. I hadnt even thought of developing my own custom controls in BP7, VB, or C++, yet with Delphi I have already created 4.

Below is my first custom control, so please be merciful. I needed a simple currency edit field, and the MaskEdit field simply does not cut it. I feel the CurrencyEdit field is a small step forward in usability. It isnt perfect and can definitely be improved on.

Essentially, the CurrencyEdit field is a modified memo field. I have put in keyboard restrictions, so the user cannot enter invalid characters. When the user leaves the field, the number is reformatted to display appropriately. You can left-, center-, or right-justify the field, and you can also specify its display format - see the FormatFloat command. Since an modifiable display format is used, it really doesnt even need to hold a currency value... It could just as easily be a percentage, or a high-precision real value with fixed digits to the right of the decimal. The field value is stored in a property called Value so you should read and write to that in your program. This field is of type Extended.

If you like this control you can feel free to use it, however, if you modify it, I would like you to send me whatever you did to it. If you send me your CIS ID, I will send you copies of my custom controls that I develop in the future. Please feel free to send me anything you are working on as well. Perhaps we can spark ideas!

CurrencyEdit Source Code
Contents

Source Code for TCurrencyEdit

```
Unit CurrEdit;

Interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Menus, Forms, Dialogs, StdCtrls;

type
  TCurrencyEdit = class(TCustomMemo)
  private
    DispFormat: string;
    FieldValue: Extended;
    procedure SetFormat(A: string);
    procedure SetFieldValue(A: Extended);
    procedure CMEnter(var Message: TCMEnter);              message CM_ENTER;
    procedure CMExit(var Message: TCMExit);                message CM_EXIT;
    procedure FormatText;
    procedure UnFormatText;
  protected
    procedure KeyPress(var Key: Char); override;
    procedure CreateParams(var Params: TCreateParams); override;
  public
    constructor Create(AOwner: TComponent); override;
  published
    property Alignment default taRightJustify;
    property AutoSize default True;
    property BorderStyle;
    property Color;
    property Ctl3D;
    property DisplayFormat: string read DispFormat write SetFormat;
    property DragCursor;
    property DragMode;
    property Enabled;
    property Font;
    property HideSelection;
    property MaxLength;
    property ParentColor;
    property ParentCtl3D;
    property ParentFont;
    property ParentShowHint;
    property PopupMenu;
    property ReadOnly;
    property ShowHint;
    property TabOrder;
    property Value: Extended read FieldValue write SetFieldValue;
    property Visible;
    property OnChange;
    property OnClick;
    property OnDblClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnEnter;
    property OnExit;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
  end;

procedure Register;

implementation
```

```
procedure Register;
begin
  RegisterComponents('Additional', [TCurrencyEdit]);
end;

constructor TCurrencyEdit.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  AutoSize := True;
  Alignment := taRightJustify;
  Width := 121;
  Height := 25;
  DispFormat := '$,0.00;($,0.00)';
  FieldValue := 0.0;
  AutoSelect := False;
  WantReturns := False;
  WordWrap := False;
  FormatText;
end;

procedure TCurrencyEdit.SetFormat(A: String);
begin
  if DispFormat <> A then
    begin
      DispFormat:= A;
      FormatText;
    end;
end;

procedure TCurrencyEdit.SetFieldValue(A: Extended);
begin
  if FieldValue <> A then
    begin
      FieldValue := A;
      FormatText;
    end;
end;

procedure TCurrencyEdit.UnFormatText;
var
  TmpText : String;
  Tmp     : Byte;
  IsNeg   : Boolean;
begin
  IsNeg := (Pos('-',Text) > 0) or (Pos('(',Text) > 0);
  TmpText := '';
  For Tmp := 1 to Length(Text) do
    if Text[Tmp] in ['0'..'9','.'] then
      TmpText := TmpText + Text[Tmp];
  try
    FieldValue := StrToFloat(TmpText);
    if IsNeg then FieldValue := -FieldValue;
  except
    MessageBeep(mb_IconAsterisk);
  end;
end;

procedure TCurrencyEdit.FormatText;
begin
  Text := FormatFloat(DispFormat,FieldValue);
end;

procedure TCurrencyEdit.CMEnter(var Message: TCMEnter);
begin
  SelectAll;
  inherited;
end;
```

```
procedure TCurrencyEdit.CMExit(var Message: TCMExit);
begin
  UnformatText;
  FormatText;
  Inherited;
end;

procedure TCurrencyEdit.KeyPress(var Key: Char);
begin
  if Not (Key in ['0'..'9','.','-']) Then Key := #0;
  inherited KeyPress(Key);
end;

procedure TCurrencyEdit.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  case Alignment of
    taLeftJustify   : Params.Style := Params.Style or ES_LEFT and Not ES_MULTILINE;
    taRightJustify : Params.Style := Params.Style or ES_RIGHT and Not ES_MULTILINE;
    taCenter          : Params.Style := Params.Style or ES_CENTER and Not ES_MULTILINE;
  end;
end;

End.
```

The Bug Hunter Report

Even the best programming language has them. But with a little bit of inter-communication we can get around any limitations they impose. For the first release of Delphi, I find that the environment and the compiler are remarkably solid. Yet, I have found a few things that I thought I would pass on…

Speedbar and Component Palette

In case you are fiddling around with the ability to show and hide the Speedbar and the Component Palette at the top of the Delphi IDE, try not to hide both of them at once. I did and it took me over an hour to get them back. The reason is that there is an internal value that holds the location of the dividing bar between the two. By some strange sequence of events this bar located itself off of the screen (thereby making it unreachable by mouse). Turning on the Speedbar or the Component Palette independently worked, but as soon as I tried to get both of them up at once, the IDE placed the Speedbar and the dividing bar off the screen to the left. I have had some difficulty reproducing this consistently, but it does happen. ***The solution:*** In the DELPHI.INI file residing in your main Windows directory, there is a item labeled Split in the [MainWindow] section. This is the location of the divider bar. Simply alter this number to put it on the screen somewhere and restart Delphi.

Resource Name Case Sensitivity

There is an erroneous statement in the section Adding Palette Bitmaps in the CWG.HLP help file (Component Writers Guide). Inside the Example hot link, it states: *The resource names are not case-sensitive, but by convention, they are usually in uppercase letters.*

It turns out that they are case sensitive, and must be all in upper case. When defining a new Bitmap for a custom component I had developed, I used mixed case on the name **TCurrencyEdit** and for the life of me could not figure out why it was not loading the bitmap correctly. It wasnt until I changed it to all upper case (**TCURRENCYEDIT**) that it worked correctly. When you discover documentation errors like this, it is a good habit to use the **Annotate** feature of the help system (under the Edit menu) to leave a note to yourself about the problem and the workaround.

Lockups While Linking

This is an operating without a safety net issue. While working on a Delphi project, an associate innocently named a form as Header. From that point on every time he tried to compile the application, Delphi would get to the Linking... stage and then lockup Windows, forcing a very messy restart. Finally, and after much frustration, he tracked the problem back to this change and after rewording the form name, the fast and efficient compile process was restored to health.

It turns out that when you enter a Form name, (and probably in other areas I have not yet discovered) Delphi creates a Type declaration of the name with a T added to the front. Unfortunately, adding a T to the Header name creates THeader causing a naming conflict with the standard Delphi component **THeader**.

To see this behavior (as if we really need another excuse to see an operating system blow up on us) simply start a new project. Name the form Header and then drop a THeader component on it. Then look at the code and see why this is clearly a problem to be avoided! In this example, the application apparently compiles cleanly, but watch the sparks fly when you try to run it! (and make sure you have saved any other work before you do so, please).

I guess the solution to this is to pay close attention to your form, variable, and type names, to make sure you dont generate a naming conflict like this. It would be nice if Borland can resolve this with a compile-time error (some kind of *Type Mismatch* error would probably do). However, I suspect it probably would not happen simply for the reason that the amount of name and type checking involved might be far too costly to implement. The best solution for now is simply to name with caution!

Saving Files in the Image Editor

Be very careful about saving files created with the Image Editor application provided with Delphi (found under the Tool menu). I have encountered a number of bugs relating to the saving of files. On one occasion I had a file with three bitmaps in it, I would delete one of the bitmaps, pick Save, and then quit the Image Editor. Later, when I came back to the same file, the one I deleted was still there! The only way I could get it to correctly remove the image was to use Save As… and create an entirely new resource output file. As you can imagine, it gets pretty annoying having to do this all the time.

Another seemingly related bug, occurred just recently as well. I started a new .RES resource file, added 3 bitmaps and let them keep their default naming (BITMAP_#). After creating each, I scribbled on the bitmap area so I could distinguish them. Then I saved the file, quit the Image Editor, came back and reloaded the file. So far, everything is fine. Then I click on BITMAP_2 and click the remove button, save the file, quit the Image Editor, start it up again (I am quitting and restarting the editor for this test to make sure I am not introducing any other issues). When I retrieve the file again, now BITMAP_2 is gone, but I have two identically named BITMAP_3 bitmaps!

I have also seen occasions where I make a change to a resource file (such as deleting an icon for example) and double-click on the Image Editors close box, assuming it will give me the Would you like to save this file? dialog box. Nope. Apparently there are some conditions where its internal I have been modified flag is not getting set properly causing some file changes to be lost.

I am not too concerned about these problems, as Borland will likely release a new copy of the editor and post it on the various on-line services. Until then, I think I will stick with Resource Workshop, or continue to use the more convenient Image Editor, but just walk a bit more gingerly…

About The Editor

For those of you who actually care, I thought I would tell a little bit about myself:

I have been a Pascal/VB/C++ programmer (in that skill order) for the past 8 years or so. For most of that time I have been self-employed and have been designing and moderating Play-By-Mail games. The language for these games has been Pascal from the very early stages (OK, I used dBase for a while before that, but Im not too proud of that fact.) If memory serves, I started with Turbo Pascal 3.0 and migrated through all of the ensuing versions until getting to BP7.

And then, just when I figured there wasnt much more to learn… Delphi! I must admit that programming in Pascal was looking like a dead-end street these past 2 years. It wasnt too impressive on résumés and I got sour looks from people whenever I mentioned Pascal. It is my opinion that Delphi will reverse quite a lot of that. I truly hope it has a long a fruitful life.

About a year ago, (after selling the games I had developed), I started doing some contract programming work for a major utility company that needed a Pascal programmer (dont ask me...). This has given me an excellent opportunity to expand my experience with Pascal, and, because of their newly found desire to work with Delphi, to have the time to work with Borlands latest toy.

Contents

The Walnut Creek Delphi Users Group

Since I have never formed a Users Group before, I figured the best thing to do would be to just put the word out! I will be forming the Walnut Creek Delphi Users Group effective immediately. The current roster indicates a membership of, hmmmm...let me see... (rustling paper)... **ONE!** (Thats me in case you didnt get it...)

Walnut Creek is located in the San Francisco Bay Area, about 20 minutes East of Oakland. Currently there is no official meeting schedule until I determine how many of you there are in the vicinity. I have several places available for meeting locations, so that should not be a problem.

If you are interested in getting together with other Delphi programmers in the Walnut Creek area, please contact me via CompuServe. My CIS ID is 76416,1373. Please let me know where you live, and the general level of experience you have with both Pascal and Delphi. Initially, I will be contacting you via CompuServe, but later we might setup a phone list for face-to-face (or rather ear-to-ear) communications.

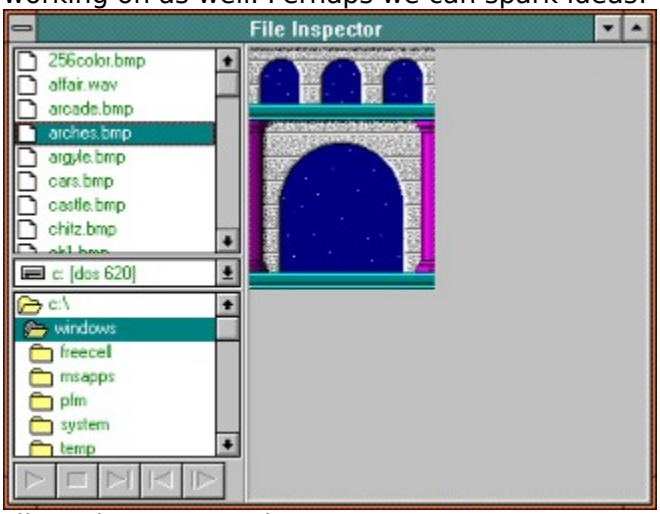
Contents

Sample Applications

This is a simple little file viewer I whipped together in an hour or two after getting my Delphi. It is able to display ICO, BMP, WMF files as well as playing WAV and AVI files via the MediaPlayer control.

You will notice that there arent too many lines of code (about 25 it looks like). This attests to the amazing power of Delphi applications. I would hate to have tackled this before Delphi came out. I have written a number of pretty complex Windows apps using BP7 and the amount of drudgery that Delphi alleviates is immense.

If you like this program you can feel free to use it, however, if you modify it, I would like you to send me whatever you did to it. If you send me your CIS ID, I will send you copies of my custom controls that I develop in the future. Please feel free to send me anything you are working on as well. Perhaps we can spark ideas!



FilePeek Source Code
FilePeek Form Layout
Contents

FilePeek Source Code

```
Unit PeekUnit;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, StdCtrls, FileCtrl, MPlayer;

type
  TForm1 = class(TForm)
    LBHolder: TPanel;
    FL: TFileListBox;
    DL: TDirectoryListBox;
    DCB: TDriveComboBox;
    ImageHolder: TPanel;
    Image: TImage;
    MP: TMediaPlayer;
    AVIPanel: TPanel;
    procedure FLChange(Sender: TObject);
    procedure FLDblClick(Sender: TObject);
    procedure AVIPanelResize(Sender: TObject);
    procedure FormResize(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FLChange(Sender: TObject);
begin
  if FL.ItemIndex >= 0 then
    begin
      {Close the Media Player}
      MP.Close;
      {Only stretch metafiles}
      Image.Stretch := (Pos('.wmf',FL.FileName) > 0);
      if (Pos('.bmp',FL.FileName) > 0) or
          (Pos('.wmf',FL.FileName) > 0) or
          (Pos('.ico',FL.FileName) > 0) then
          begin
            Cursor := crHourglass;
            Image.Picture.LoadFromFile(FL.FileName);
            AVIPanel.Hide;
            Image.Show;
            Cursor := crDefault;
          end;
    end;
end;

procedure TForm1.FLDblClick(Sender: TObject);
begin
  {If the user double-clicks on any WAV or AVI file, it will load and
   activate the MediaPlayer control. AVI files are displayed on their own
   area (AVIPanel). I selectively hide and show the AVIPanel and the Image
   control based off of what is being displayed. This was because the
   MediaPlayer needs a Panel to display on, and the BMP's, ICO's, and WMF's
   needed an Image control to display on}
  if (Pos('.wav',FL.FileName) > 0) or (Pos('.avi',FL.FileName) > 0) then
    begin
```

```
        Screen.Cursor := crHourglass;
        MP.FileName := FL.FileName;
        MP.Open;
        if (Pos('.avi',FL.FileName) > 0) then
          begin
            Image.Hide;
            AVIPanel.Show;
            MP.Display := AVIPanel;
            {I don't know why I have to shift the AVI panels display rect, but
            it was off center if I didn't. Oh well, someone out there will be
            able to explain it I am sure.}
            With AVIPanel.BoundsRect do
              MP.DisplayRect := Rect(Left-3,Top-3,Right-3,Bottom-3);
          end;
        Screen.Cursor := crDefault;
        MP.Play;
      end;
end;

procedure TForm1.AVIPanelResize(Sender: TObject);
begin
  With AVIPanel.BoundsRect do
    MP.DisplayRect := Rect(Left-3,Top-3,Right-3,Bottom-3);
end;

procedure TForm1.FormResize(Sender: TObject);
begin
  {This procedure resizes the list boxes based off of the size of the main
   form. This allows the listboxes to always be as large as they can}
  MP.Top := Form1.ClientHeight-3-MP.Height;
  DCB.Top := (Form1.ClientHeight-DCB.Height) div 2;
  FL.Height := DCB.Top - 6;
  DL.Top := DCB.Top + DCB.Height + 3;
  DL.Height := Form1.ClientHeight - DL.Top - MP.Height - 6;
end;

end.
```

FilePeek Form Layout

Simply copy all of this text using the Edit/Copy command, and paste it into a new blank document in the Delphi IDE. Then save the file choosing the File Type as a **DFM** file.

```
object Form1: TForm1
  Left = 201
  Top = 100
  Width = 435
  Height = 333
  ActiveControl = DL
  Caption = 'File Inspector'
  Font.Color = clGreen
  Font.Height = -12
  Font.Name = 'MS Sans Serif'
  Font.Pitch = fpVariable
  Font.Style = []
  PixelsPerInch = 96
  Position = poScreenCenter
  OnResize = FormResize
  TextHeight = 13
  object LBHolder: TPanel
    Left = 0
    Top = 0
    Width = 157
    Height = 306
    Align = alLeft
    TabOrder = 0
    object FL: TFileListBox
      Left = 3
      Top = 3
      Width = 151
      Height = 118
      ItemHeight = 16
      Mask = '*.wmf;*.bmp;*.ico;*.wav;*.avi'
      ShowGlyphs = True
      TabOrder = 0
      OnChange = FLChange
      OnDblClick = FLDblClick
    end
    object DL: TDirectoryListBox
      Left = 3
      Top = 149
      Width = 151
      Height = 126
      FileList = FL
      ItemHeight = 16
      TabOrder = 1
    end
    object DCB: TDriveComboBox
      Left = 3
      Top = 124
      Width = 151
      Height = 19
      DirList = DL
      TabOrder = 2
    end
    object MP: TMediaPlayer
      Left = 3
      Top = 278
      Width = 151
      Height = 25
      VisibleButtons = [btPlay, btStop, btNext, btPrev, btStep]
      ParentShowHint = False
      ShowHint = True
      TabOrder = 3
    end
  end
```

```
  object ImageHolder: TPanel
    Left = 157
    Top = 0
    Width = 270
    Height = 306
    Align = alClient
    BevelInner = bvLowered
    BorderWidth = 1
    TabOrder = 1
    object Image: TImage
      Left = 3
      Top = 3
      Width = 264
      Height = 300
      Align = alClient
      Stretch = True
    end
    object AVIPanel: TPanel
      Left = 3
      Top = 3
      Width = 264
      Height = 300
      Align = alClient
      BevelOuter = bvNone
      TabOrder = 0
      Visible = False
      OnResize = AVIPanelResize
    end
  end
end
```