The Unofficial Newsletter of Delphi Users

Issue #3 - May 1st, 1995

Robert Vivrette - CIS: 76416,1373

Welcome back! First, I want to apologize over the delay in getting this third issue of **UNDU** to you. A 2-week publication schedule is a little more than I can handle, so it looks like it will be more like 3-4 weeks at this point.

I want to thank those of you who have contributed material to me for this issue. For space & size reasons, I have not included everything I have received for this latest issue, but rest assured more will be coming!

I would like to toss up a few article ideas to see if I can catch anyone's interest. If you would like to write on any of the following topics, please contact me:

1. Review/Discussion of ReportSmith
2. Discussion/Examples of Database operations
3. Delphi Book/Magazine Reviews

Please do think about contributing. Even small little code snippets would make a good addition to the newsletter.

Publications Available

There has been a lot of activity in the bookstores and on the newstands lately with regards to Delphi. Each month, I will bring you a list of the currently available books and periodicals as well as an idea of their content and presumed experience level. I will also highlight articles about Delphi that appear in various other programming periodicals.

Return To Front Page

The Unofficial Newsletter of Delphi Users - Issue #3 - May 1st, 1995

Books Currently Available

Delphi Programming For Dummies

Delphi By Example

Instant Delphi Programming

Teach Yourself Delphi

Books On The Way

Delphi Developers Guide

Delphi How-To

Delphi Nuts & Bolts

Delphi Programmer Explorer

Developing Client/Server Applications With Delphi

Developing Windows Applications Using Delphi

Master Delphi

Mastering Delphi

Software Engineering With Delphi

Teach Yourself Delphi in 21 Days

Using Delphi, Special Edition

I define "Currently Available" as meaning "in the stores". If you notice any incorrect or incomplete information listed here, please feel free to contact me. I will be updating this list each month. I am currently looking for individuals who would like to write reviews of any Delphi-related books. Please contact me if you would like to help!

Publications Available
Return To Front Page

Title:	***Delphi Nuts & Bolts***
Author:	Gary Cornell & Troy Strain
Level:	Intermediate/Experienced
Publisher:	Osborne-McGraw Hill
ISBN #:	0-07-882136-3
Available:	May, 1995
List Price:	$24.95
Summary:	A 330 page book designed for anyone with some programming experience. Covers the features of Delphi, but assumes you don't need your hand held. If you are an experienced programmer and want a fast introduction to Delphi, this book is for you.

Title:	***Teach Yourself Delphi in 21 Days***
Author:	Andrew J. Wozniewicz
Level:	Beginner/Intermediate
Publisher:	Sams Publishing
ISBN #:	0-672-30470-8
Available:	In Production
List Price:	$29.99
Summary:	Introduces Delphi to the beginning programmer and includes question-and-answer section at end of each less to test readers progress as they learn. Emphasis on Object Pascal.

Title:	***Instant Delphi Programming***
Author:	Dave Jewell
Level:	Intermediate
Publisher:	Wrox Press
ISBN #:	1-874416-57-5
List Price:	$19.95
Summary:	Instant Delphi is the fast-paced tutorial guide for the programmer who wants to get up to speed on the Delphi product as quickly as possible.

Title:	***Using Delphi, Special Edition***
Author:	Jon Matcho & David Faulkner
Level:	???
Publisher:	Que
ISBN #:	1-56529-823-3
List Price:	$29.99
Summary:	This 3-part tutorial on the most important Delphi features covers how to install the product and develop applications using Delphi's visual tools, explores the Windows application development process, and deals with some advanced programming topics.

Title:	***Delphi Programming For Dummies***
Author:	Neil Rubenking
Level:	Beginner
Publisher:	IDG
ISBN #:	1-56884-200-7
List Price:	$19.99
Summary:	It may be Greek to you now, but not for long! Delphi Programming For Dummies is your ticket to writing Windows applications with Borland's new product, Delphi. This fun, results-oriented book jumps right into creating working programs by collecting and connecting Delphi's powerful components. You'll find out what you need to know about how your programs work; previous programming experience is not necessary!

Periodicals Currently Available

Delphi Informant

Periodicals On The Way

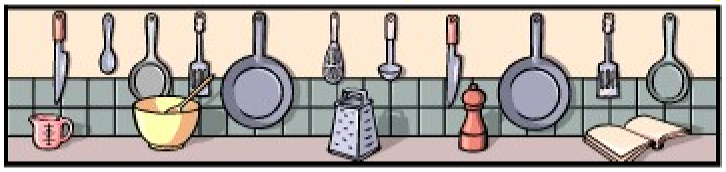
Delphi Developer

I am currently looking for individuals who would like to write reviews of any Delphi-related periodicals. Please contact me if you would like to help!

Publications Available
Return To Front Page

Periodical:	***Delphi Informant***
Publisher:	Informant Communications Group
Issue Cost:	$4.95 US, $5.95 Canada, £7.50 UK
Subscription:	1 Year/12 Issue - $49.95 US, $54.95 Canada, £79.95 UK
Latest Issue:	Volume 1, Number 1
Contact:	Circulation Department, Delphi Informant, 10519 E. Stockton Blvd. Suite 142, Elk Grove, CA 95624-9704. (916) 686-6610
Summary:	Delphi Informant is the Complete Monthly Guide to Delphi Development. It features in-depth discussion of pressing technical topics facing the Delphi programmer. Each issue is packed with the latest development techniques including database development, object-oriented programming, client/server programming, component building, Windows development, news from the Delphi community, third-party product releases, product reviews, book reviews, Delphi user group information, and development with third-party tools.
	To receive one year of the Delphi Informant for US$49.95, call Informant Communications Group at (800) 88INFORMANT (800-884-6367) or outside the US call (916) 686-6610. Delphi Informant is also available in stores such as Barnes and Nobel, B Dalton Bookseller, Tower Books, CompUSA, Computer City, and Waldenbooks. Code listings may be downloaded from the Informant Communications Group BBS at (916) 686-4740.

Premiere Issue



Cooking Up Components

Hello. My name is Robert. And I am a componentaholic. *(Hi Robert!!!)*

That's right folks. I'm hooked on components, and you'll never get me to go back. Over the last month I have been having a ball creating every sort of component you can imagine. When I am not designing a component at work for some of the Delphi applications we are writing here, I am designing some for myself at home. Component design is one of the most rewarding aspects of Delphi. The ability to create a windows control and then to have it become an integral part of the same language that created it is phenomenal.

The component for this issue is called **TImageTransform**. It provides the ability to transform one bitmap into another using various visual effects such as wipes and fades. Currently, I have only implemented a few of the simpler effects, but I will be presenting others later on that you can add to this component. Also, if you like this component and come up with your own transformation effect, send it to me and I will include it in a future issue.

The component illustrates a few interesting aspects of Windows/Delphi programming. First, it overrides some basic behaviors of the stock graphic components to provide flicker-free motion on the screen. Second, it illustrates the use of custom event handlers.

Using the component is very simple. There are 2 "Picture" fields in the object inspector. First assign a bitmap to each of these properties. Ideally, they should both be the same size and color depth, to minimize any potential visual side-effects. Second, choose a Transform type from those provided. And lastly, when you want the images to switch, simply call the objects "Transform" method. The speed of the transformation and the number of steps the transformation is broken down into, is adjustable by means of the "Interval" and "Steps" properties.

TImageTransform Component Source Code
Modifications to the **CurrEdit** component

Return to Front Page

Articles

I have been doing quite a bit of hunting for Delphi-related articles. I have managed to catch a few, but no doubt I have missed many more. If you have seen an article that discusses Delphi issues and I have not mentioned it here, please let me know and I will pick it up!

Game Developer, April/May 1995

PC Techniques, Feb/Mar 1995

PC Techniques, April/May 1995

Software Development, April 1995

Windows Tech Journal, March 1995

Delphi Informant, Volume 1, Number 1

Publications Available
Return To Front Page

Game Developer Magazine

In the April/May 1995 issue, Larry O'Brien devoted his editor's column to a discussion of Delphi entitled *"Delphi is the Answer"*. He made some very favorable comments and discussed Delphi's impact for Windows game developers. Some excerpts:

"Normally, I make it a rule not to recommend any programming language"... "I am going to break that rule to recommend Borland's Delphi".

"For Windows developers and for Windows game developers especially, Delphi is so far ahead of the competition, it's embarrassing."

"I'll make a bold prediction - the most successful Windows game of 1996 will be written entirely in Delphi."

Title:	***Delphi Programmer Explorer***
Author:	by J. Duntemann/J. Mischel/D. Taylor
Level:	???
Publisher:	Coriolis Group
ISBN #:	1-883577-25-X
List Price:	$39.99
Summary:	A new type of tutorial: Theory and practice alternate in short chapters, with the emphasis on creating useful software starting on the very first page.

Title:	***Mastering Delphi***
Author:	Marco Cantu
Level:	???
Publisher:	Sybex
ISBN #:	0-7821-1739-2
List Price:	$29.99
Summary:	Introduces programmers to Delphi's features and techniques, including secrets of the environment, the programming language, the custom components and Windows programming in general.

Title:	***Delphi Developer's Guide***
Author:	by Xavier Pacheco/Steve Teixeira
Level:	Intermediate/Advanced
Publisher:	Sams Publishing
ISBN #:	0-672-30704-9
List Price:	$45.00
Summary:	Intermediate to advanced guide to developing applications using Delphi.

Title:	***Delphi How-To***
Author:	by Gary Frerking
Level:	???
Publisher:	Waite Group Press
ISBN #:	1-57169-019-0
List Price:	$34.95
Summary:	Presents large collection of programming problems and their solutions in standard, easy-to-use reference format, including unique solutions that use VBX controls and easy ways to build multimedia projects with Delphi.

Title:	***Developing Windows Applications Using Delphi***
Author:	by Paul Penrod
Level:	???
Publisher:	John Wiley
ISBN #:	0-471-11017-5
List Price:	$29.95
Summary:	This introduction for traditional C programmers who want to make the transition to rapid application development also provides detailed instructions for building sophisticated Windows applications and for creating graphical interfaces.

Title:	***Master Delphi***
Author:	by Charlie Calvert
Level:	???
Publisher:	Sams Publishing
ISBN #:	0-672-30499-6
List Price:	$45.00
Summary:	Comprehensive tutorial/reference for intermediate programming with Delphi.

Title:	***Teach Yourself Delphi***
Author:	Devra Hall
Level:	Beginner
Publisher:	MIS Press
ISBN #:	1-55828-390-0
List Price:	$27.95
Summary:	Here is a complete, self-guided tour to the new development environment from Borland, encompassing all the features of the language and all the tools, tricks, and advantages of Delphi. No programming experience necessary.

Title:	***Delphi by Example***
Author:	by Blake Watson
Level:	Beginner
Publisher:	Que
ISBN #:	1-56529-757-1
List Price:	$29.99
Summary:	Aimed at the beginning programmer who has no prior experience with other languages or development products, this book presents basic concepts of programming along with a clear explanation of the key development tools that are part of Delphi.

Title:	***Developing Client/Server Applications with Delphi***
Author:	Vince Killen and Bill Todd
Level:	**???**
Publisher:	Sams Publishing
ISBN #:	
List Price:	
Summary:	Walks the reader through the development process of creating real-world C/S applications, explaining in detail what the thought processes must be even before any code is written.

Title:	***Software Engineering with Delphi***
Author:	Edward C. Webber, J. Neal Ford, and Christopher R. Webber
Level:	
Publisher:	Prentice Hall Professional, Trade & Reference
ISBN #:	
List Price:	
Summary:	A guide to developing client/server applications with an emphasis on Delphi's object-oriented tools.

Copying Table Records

Contributed by Alec Bergamini - 75664,1224

If you like useful code snipits here's one that's real sweet. It copies a single record from one table to another table (both tables have the same structure). It only works if the TFields for each table are exactly the same. The user should be aware of the restricitions on the assign method prior to using this. I wrote it after searching in vain for a method to copy a single record. BatchMove is cool but does nothing on the record level. This procedure is nice because cause it is generic and doesn't requre knowledge of the tables record structures. It copies the current record in the source table to the destination table. It even handles blobs.

```
Procedure CopyRecord(const SourceTable, DestTable : TTable);
var
  I : Word;
begin
  DestTable.Append;
  For I := 0 to SourceTable.FieldCount - 1 do
     DestTable.Fields[I].Assign(SourceTable.Fields[I]);
  DestTable.Post;
end;
```

Tips & Tricks

Return to Front Page

PC Techniques Magazine

The Apr/May 1995 issue of *PC Techniques* presents another article by Ray Konopka titled *"Reusability - Delphi Style"*. It gives a good overview of developing an application in Delphi as well as the creation of a custom component that will display icons in a list box. The application developed uses the TIconListBox component to create an icon viewer utility. If you have a lot of icons lying around on your hard disk (like me), this can be a pretty handy little utility. Gives some good insights into Delphi programming as well.

The same issue also includes an article on communications using Delphi and presents a **TComm** component for doing serial communications.

Software Development Magazine

The April 1995 issue of *Software Development* magazine carried an article on code reusability entitled *"Delphi's Code Reuse Infrastructure"*. It gives a very good discussion of the benefits and implementation of code reuse principles. With regard to Delphi, it covers how Components, Form Templates, and Project Templates bring different aspects of reusability to a corporate developer.

Windows Tech Magazine

The March 1995 issue of *Windows Tech Journal* had an article titled *"Self-Constructing Objects"* by Blake Watson. It also discusses component design but illustrates its use by means of a component that bridges the gap between the older Pascal Records and typed files into Delphi's more object oriented structures. Blake is also the author of *Delphi By Example*.

PC Techniques Magazine

The Feb/Mar 1995 issue of *PC Techniques* presents an article titled *"Creating Delphi Components"* by Ray Konopka. It discusses the basics of component design by using the example of a Traffic light graphic control. A very simple control, but the article is clear and well illustrated and serves as a good starting point for developers learning about graphic custom components.

Delphi Informant Magazine

For those of you who have not already obtained a copy of the premier of Delphi Informant, I will give a brief overview of its contents.

"Visual Programming" provides a brief visual tour of the Delphi IDE discussing the various features such as the Component Palette, Object Inspector, Code editor, and more.

"The Way of Delphi" illustrates the fundamentals of component design and begins with a sample component. A very good tutorial for users who have not yet stepped into component creation.

"DataSource1" demonstrates a number of Delphi's capabilities including modifying a Query component at run-time, sharing an event handler among multiple components, and creating a dynamic tabbed interface to filter database information.

"Sights & Sounds" discusses the creation of a simple Audio CD player application. Very good introduction to the Media Player component as well as connections to the Windows API.

"Delphi C/S" presents a discussion of Delphi Client/Server capabilities covering issues such as ODBC, the BDE, and the Delphi Database Form Expert.

"DBNavigator" discusses Delphi's database features covering topics including InterBase, Paradox for Windows, and dBase for Windows, as well as the BDE.

"From The Palette" introduces the basics of Component design, including property and method basics and the proper use of class directives.

"At Your Fingertips" is a "tips & tricks" column devoted to help programmers quickly get up to speed on Delphi and Object Pascal. This issue's column includes: Implementing a status bar, locating values in a table, and filtering records in a DBGrid.

"API Calls" provides a very good discussion of dynamically calling DLL functions from within Delphi.

TImageTranform Component Source Code

```
unit Transfrm;

interface

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

type
  TTransType = (ttWipeLeft,ttWipeRight,ttWipeUp,ttWipeDown,
                ttTurnLeft,ttTurnRight,ttTurnUp,ttTurnDown,
                ttWipeDownRight,ttWipeDownLeft,ttWipeUpRight,ttWipeUpLeft);
  TImageTransform = class(TGraphicControl)
  private
    FPicture1   : TPicture;
    FPicture2   : TPicture;
    FAutoSize   : Boolean;
    FTimer      : TTimer;
    FInterval   : Integer;
    FImageShown : Byte;
    FSteps      : Integer;
    FType       : TTransType;
    StepNum     : Integer;
    FOnFinished  : TNotifyEvent;
    procedure PictureChanged(Sender: TObject);
    procedure SetAutoSize(Value: Boolean);
    procedure SetPicture1(Value: TPicture);
    procedure SetPicture2(Value: TPicture);
    procedure SetImageShown(Value: Byte);
    procedure SetInterval(Value: Integer);
    procedure SetSteps(Value: Integer);
    procedure SetType(Value: TTransType);
  protected
    function GetPalette: HPALETTE; override;
    procedure Paint; override;
    procedure TimerTick(Sender: TObject);
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure Transform;
  published
    property AutoSize: Boolean read FAutoSize write SetAutoSize default False;
    property DragCursor;
    property DragMode;
    property Enabled;
    property ImageShown: Byte read FImageShown write SetImageShown default 1;
    property Interval : Integer read FInterval write SetInterval default 1;
    property ParentShowHint;
    property Picture1: TPicture read FPicture1 write SetPicture1;
    property Picture2: TPicture read FPicture2 write SetPicture2;
    property PopupMenu;
    property ShowHint;
    property Steps: Integer read FSteps write FSteps default 10;
    property TransformType: TTransType read FType write SetType default ttWipeLeft;
    property Visible;
    property OnClick;
    property OnDblClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property OnFinished: TNotifyEvent read FOnFinished write FOnFinished;
  end;

procedure Register;

implementation

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

constructor TImageTransform.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FImageShown := 1;
  FPicture1 := TPicture.Create;
  FPicture1.OnChange := PictureChanged;
  FPicture2 := TPicture.Create;
  FPicture2.OnChange := PictureChanged;
  FTimer := TTimer.Create(Self);
  FTimer.OnTimer := TimerTick;
  FTimer.Enabled := False;
  FInterval := 1;
  FType := ttWipeLeft;
  FSteps := 10;
  Height := 105;
  Width := 105;
end;

destructor TImageTransform.Destroy;
begin
  FPicture1.Free;
  FPicture2.Free;
  FTimer.Free;
  inherited Destroy;
end;

function TImageTransform.GetPalette: HPALETTE;
begin
  Result := 0;
  if FPicture1.Graphic is TBitmap then
    Result := TBitmap(FPicture1.Graphic).Palette;
end;

procedure TImageTransform.SetAutoSize(Value: Boolean);
begin
  FAutoSize := Value;
  PictureChanged(Self);
end;

procedure TImageTransform.SetPicture1(Value: TPicture);
begin
  FPicture1.Assign(Value);
end;

procedure TImageTransform.SetPicture2(Value: TPicture);
begin
  FPicture2.Assign(Value);
```

```
end;

procedure TImageTransform.SetImageShown(Value: Byte);
begin
  If Value in [1,2] then
    begin
      FImageShown := Value;
      Invalidate;
    end;
end;

procedure TImageTransform.SetInterval(Value: Integer);
begin
  FInterval := Value;
  if Value > 1000 then FInterval := 1000;
  if Value < 1 then FInterval := 1;
  {Reset the timer interval}
  if FTimer <> Nil then FTimer.Interval := FInterval;
end;

procedure TImageTransform.SetSteps(Value: Integer);
begin
  FSteps := Value;
end;

procedure TImageTransform.SetType(Value: TTransType);
begin
  FType := Value;
end;

procedure TImageTransform.PictureChanged(Sender: TObject);
begin
  if AutoSize and (Picture1.Width > 0) and (Picture1.Height > 0) then
    SetBounds(Left, Top, Picture1.Width, Picture1.Height);
  if (Picture1.Graphic is TBitmap) and (Picture1.Width = Width) and (Picture1.Height =
Height) then
    ControlStyle := ControlStyle + [csOpaque]
  else
    ControlStyle := ControlStyle - [csOpaque];
  Invalidate;
end;

procedure TImageTransform.Transform;
begin
  StepNum := 0;
  {Turn on the timer}
  if FTimer <> Nil then
    begin
      FTimer.Interval := 1;
      FTimer.Enabled := True;
    end;
end;

procedure TImageTransform.TimerTick;
begin
  if FTimer <> Nil then FTimer.Interval := FInterval;
  Inc(StepNum);
  Paint;
  {Turn off the timer if we have reached the end.}
  if FTimer <> Nil then
    if FTimer.Enabled then
      if StepNum >= Steps then
        begin
```

```
          FTimer.Enabled := False;
          If ImageShown = 1 then ImageShown := 2 Else ImageShown := 1;
          if Assigned(FOnFinished) then FOnFinished(Self);
        end;
end;

procedure TImageTransform.Paint;
var
  PctDone  : Real;
  PctLeft  : Real;
  DestRect : TRect;
  SrcRect  : TRect;
  ShowCurrentImage : Boolean;
  TempCanvas : TCanvas;
begin
  with inherited Canvas do
    begin
      {Draw border if in design mode}
      if csDesigning in ComponentState then
        begin
          Pen.Style := psDash;
          Brush.Style := bsClear;
          Rectangle(0, 0, Width, Height);
        end;
      {Are we not animated, or on first paint of transformation?}
      ShowCurrentImage := False;
      if FTimer <> Nil then
        if not FTimer.Enabled then ShowCurrentImage := True;
      if StepNum < 1 then ShowCurrentImage := True;
      if ShowCurrentImage then
        begin
          if ImageShown = 1 then
            Draw(0,0, Picture1.Graphic)
          else
            Draw(0,0, Picture2.Graphic);
          exit;
        end;
      {Calculate percentage done and percentage left}
      if FSteps > 0 then PctDone := (StepNum/FSteps) else PctDone := 0.0;
      PctLeft := 1-PctDone;
      if PctDone > 0.0 then
        case TransformType of
          ttWipeLeft  : if ImageShown = 1 then
                          Draw(Round(Picture1.Width*PctLeft),0,Picture2.Graphic)
                        else
                          Draw(Round(Picture1.Width*PctLeft),0,Picture1.Graphic);
          ttWipeRight : if ImageShown = 1 then
                          Draw(-Round(Picture1.Width*PctLeft),0,Picture2.Graphic)
                        else
                          Draw(-Round(Picture1.Width*PctLeft),0,Picture1.Graphic);
          ttWipeUp    : if ImageShown = 1 then
                          Draw(0,Round(Picture1.Height*PctLeft),Picture2.Graphic)
                        else
                          Draw(0,Round(Picture1.Height*PctLeft),Picture1.Graphic);
          ttWipeDown  : if ImageShown = 1 then
                          Draw(0,-Round(Picture1.Height*PctLeft),Picture2.Graphic)
                        else
                          Draw(0,-Round(Picture1.Height*PctLeft),Picture1.Graphic);
          ttTurnLeft  : begin
                          with Picture1 do
                            DestRect := Rect(Round(Width*PctLeft),0,
                                           Round(Width*PctLeft)+
                                           Round(Width*PctDone),Height);
                          if ImageShown = 1 then
                            StretchDraw(DestRect,Picture2.Graphic)
                          else
                            StretchDraw(DestRect,Picture1.Graphic);
                        end;
          ttTurnRight : begin
                          with Picture1 do
                            DestRect := Rect(0,0,Round(Width*PctDone),Height);
                          if ImageShown = 1 then
                            StretchDraw(DestRect,Picture2.Graphic)
                          else
                            StretchDraw(DestRect,Picture1.Graphic);
                        end;
          ttTurnUp    : begin
                          with Picture1 do
                            DestRect := Rect(0,Round(Height*PctLeft),
                                             Width,Round(Height*PctLeft)+
                                             Round(Height*PctDone));
                          if ImageShown = 1 then
                            StretchDraw(DestRect,Picture2.Graphic)
                          else
                            StretchDraw(DestRect,Picture1.Graphic);
                        end;
          ttTurnDown  : begin
                          with Picture1 do
                            DestRect := Rect(0,0,Width,Round(Height*PctDone));
                          if ImageShown = 1 then
                            StretchDraw(DestRect,Picture2.Graphic)
                          else
                            StretchDraw(DestRect,Picture1.Graphic);
                        end;
          ttWipeDownRight : if ImageShown = 1 then
                              Draw(-Round(Picture1.Width*PctLeft),-
Round(Picture1.Height*PctLeft),Picture2.Graphic)
                            else
                              Draw(-Round(Picture1.Width*PctLeft),-
Round(Picture1.Height*PctLeft),Picture1.Graphic);
          ttWipeDownLeft  : if ImageShown = 1 then
                              Draw(Round(Picture1.Width*PctLeft),-
Round(Picture1.Height*PctLeft),Picture2.Graphic)
                            else
                              Draw(Round(Picture1.Width*PctLeft),-
Round(Picture1.Height*PctLeft),Picture1.Graphic);
          ttWipeUpRight   : if ImageShown = 1 then
                              Draw(-
Round(Picture1.Width*PctLeft),Round(Picture1.Height*PctLeft),Picture2.Graphic)
                            else
                              Draw(-
Round(Picture1.Width*PctLeft),Round(Picture1.Height*PctLeft),Picture1.Graphic);
          ttWipeUpLeft    : if ImageShown = 1 then

Draw(Round(Picture1.Width*PctLeft),Round(Picture1.Height*PctLeft),Picture2.Graphic)
                            else

Draw(Round(Picture1.Width*PctLeft),Round(Picture1.Height*PctLeft),Picture1.Graphic);
        end;
    end;
end;

end.
```

Modifications to CurrEdit

In Issue #1 of this newsletter, I presented a CurrencyEdit field that allowed formatted display and editing of numbers. Since then I have received a few suggestions for ways of improving CurrEdit, so I thought I would pass them on. The original CurrEdit was presented in Issue #1 of this newsletter.

Modifications by Massimo Ottavini
Modifications by Thorsten Suhr
Modifications by Bob Osborn

Return to Front Page

CurrEdit Modifications

Contributed by Massimo Ottavini (South Africa)   - CIS 100100,1620

I made the following enhancements to the CurrEdit component to correct an exception condition, and to allow backspacing.

```
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
    If TmpText='' Then TmpText := '0.00';     (*** Note 1 ***)
    FieldValue := StrToFloat(TmpText);
    if IsNeg then FieldValue := -FieldValue;
  except
    MessageBeep(mb_IconAsterisk);
  end;
end;

procedure TCurrencyEdit.KeyPress(var Key: Char);
begin
  if Not (Key in ['0'..'9','.','-',#8]) Then Key := #0;  (*** Note 2 ***)
  inherited KeyPress(Key);
end;
```

```
(*** Note 1 ***)
 If the field is blank an exception is raised by StrToFloat

(*** Note 2 ***)
 I could not backspace numbers entered, so just added the #8
```

Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #3 - May 1st, 1995

Connecting to an MS Access database

Contributed by David Kebler - dgkeb@aol.com

Here is a description of what you need to do get a connection to an MS ACCESS database table. This description is my first attempt so bear with me!

1. ACCESS maintains a security system for its tables which you need to look at first. Open ACCESS and load you MDB file. Go to the security menu. There are two default users (Admin and Guest) choose one or the other and then check all the boxes at the bottom that have to do with security access. You may set up a new user/password if you want but make sure it has all the access privileges. Save your ACCESS DATABASE.

2. Go to the ODBC administrator in the control panel. Click the Drivers button and make sure that there is a driver called MS ACCESS 2.0 for Microsoft Office. If not, you need to install MS Office on your machine, or get a copy of the driver from someone.

3. If you have never used the ODBC Admin before you might want to delete all the existing data sources as they are just samples anyway. Next, choose Add. In this dialog box choose the MS ACCESS 2.0 driver. In the next dialog box create a data source name (keep it simple). Under user name put ADMIN or any other user you created with full privileges in step 1. Now click the Select Database button and choose the same MS ACCESS database you modified in step 1. You have now created a source. Close the ODBC administrator Box.

4. You now need to fire up the Borland BDE Configuration Utility that came with Delphi. From the Drivers sheet click on New ODBC Driver. Go to the drop down list box for Default ODBC Driver and choose the MS ACCESS 2.0 for MS office driver. Now go to the default data source and choose the source you created in the ODBC Administrator. Now for the SQL link Driver, enter in a name (I suggest the same name as the default database source name). Click OK and you will see your new driver name as ODBC_"your source name". Go to the Alias page and click on New Alias. From the Alias Type drop down box choose the ODBC_"your source name" you just created. Under New Alias name enter a name. Once again, I suggest you use the same source name as created in step 3. Choose OK. You are now ready to access your MS ACCESS source.

5. Go to the Database Desktop. Choose ALIAS under the file menu and hit the New button. For DRIVER TYPE choose the driver you created in BDE (ODBC_"your source name"). Now enter a name in the database alias box. Enter in your user name you gave full privileges to in the MS ACCESS file (probably "Admin"). Hit the connect button and you should connect to the database. If you connected, hit the disconnect button. Click on the OK button and answer yes to the question about Saving the Alias. Now choose OPEN/TABLE. Go to the drop down box for Drive/Alias and choose the alias you created in the BDE Config Utility will appear in the list. Enter the appropriate user and password you have already defined. You should now see a list of your ACCESS tables. Double click on one and DBD should open it.

6. In Delphi drag a TDatabase component onto your form. Set its AliasName to your source (which will be in the drop down box). Enter a database name in the DatabaseName property. Do not choose the same name here (try an abbreviation). Under DriverName choose your ODBC_"your source name". Now drag and drop a TTable control to the form. Set its database name to your new abbreviated name made in the TDatabase control. Now drag on a TDBsource component and Edit Fields etc to view your data tables. Of course you don't necessarily need the TDatabase component as you can access this directly from the TTable. Try checking out the DBrowser that came with Delphi. I loaded the BDETable unit that comes with Delphi into the library and reused the DBOpen unit of the DBrowser program to create a way for users to open any alias. When you are in design mode though you will need to set the TDatabase alias and the connected property to True to have access to the table structure. Before running it turn Connected to False, let the DBOpen form set the alias name, and have your code set the TDatabase.Connected property to True. This works great and is a great template for starting any project. It makes more sense to users too who are familiar with having to "open" a file when starting a program.

Return to Front Page

Unit Ordering Bug

Contributed by Hallvard Vassbotn - INTERNET:hallvard@falcon.no

The compiler seems to be a bit picky about unit order in the uses-statement. There as been many people in comp.lang.pascal reporting that they get compile errors like:

```
  WinCrt, MyObj;
        ^Error 85: ";" expected.
```

With some experimentation I've found that moving WinCrt so that it becomes the last unit solves the problem. The reason seems to be that WinCrt checks the presence of VCL by checking some signature bytes and using a windows handle if it finds it. The signature and data are stored at fixed addresses in the beginning of DSeg. I think that Borland hardcoded the compiler to force the WinCrt unit be loaded after any other units to make sure the signature is written by VCL units (Controls, actually) prior to being checked by WinCrt.

For example, if you use the application expert to create a CRT application and then add a new unit, Delphi will automatically generate the following code:

```
unit Unit1;
interface
implementation
end.

program CrtApp;
uses
  WinCrt,
  Unit1 in 'UNIT1.PAS';
begin
  Writeln('Delphi');
end.
```

However, the program will not compile, saying that it expects a semicolon (;). The solution is to move the WinCrt unit as the last unit in the uses clause:

```
program CrtApp;
uses
  Unit1 in 'UNIT1.PAS',
  WinCrt;
begin
  Writeln('Delphi');
end.
```

Then it compiles fine.

When Things Go Wrong

Return to Front Page

"Real" Type Property Bug

Contributed by Hallvard Vassbotn - INTERNET:hallvard@falcon.no

Although the documentation (the Delphi Object Language Guide downloaded from Borlands CIS and FTP sites) states that published properties cannot be of the real type (Pascal specific floating point type), the compiler doesn't complain and compiles it fine. However, if you try to use the Object Inspector on such a component, it will generally give you a GPF.

This can be tested by installing and adding this component to a form. Delphi will GPF with an invalid op-code message. Generally it will lock up the computer really good, so be sure to save all work before trying this.

```
unit Realcmp;

interface

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

type
  TRealComp = class(TComponent)
  private
    FRealProp : real;
  protected
  public
  published
    property RealProp: real read FReadProp write FReadProp;
  end;

procedure Register;

implementation

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

end.
```

So, beware that the compiler will allow this behavior even though the documentation says not to do it.

When Things Go Wrong
Return To Front Page

When Things Go Wrong

If you have any bugs, documentation errors, or unexplained behaviors in Delphi, please forward them to me and I will include them in the next issue of the newsletter.

Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #3 - May 1st, 1995

Tips & Tricks

Do you have a programming technique that you are particularly proud of? Have you discovered a neat trick with Delphi? Here's the place for it!

Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #3 - May 1st, 1995

Hidden In Delphi's About Box

Did you know that you have strange things lurking in your Delphi About Box? Try these!

```
<alt> A-N-D                Picture of Anders Hejlsberg /w a wink
                           (The BIG Cheese of Delphi..so to speak)
<alt> T-E-A-M              A list of those responsible for this fine product
<alt> D-E-V-E-L-O-P-E-R-S  A list of the R&D folks specifically
```

There does seem to be some occasions where the first key-combination (A-N-D) does not seem to work, but I have not figured out why yet. If you know of other hidden surprises in the About Box, please let me know!

Tips & Tricks

Return to Front Page

CurrEdit Modifications

Contributed by Thorsten Suhr - CIS: 100275,2736

After adding the CurrEdit component into the VCL and testing it with a sample form, I ran into a problem. I entered a number and received a floating point exception error. I examined the source-code and quickly discovered, that the decimal-separator had been hard-coded. Since I am in Germany, the decimal-separator is different. Delphi has several constants, which allow easy handling of international language support. It gets those pieces of information from the WIN.INI File and the Unit SYSUTILS contains these constants. So I altered the source to exchange the "." with the constant 'DecimalSeparator" and it works fine.

```
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',DecimalSeparator] then
      TmpText := TmpText + Text[Tmp];
  try
    FieldValue := StrToFloat(TmpText);
    if IsNeg then FieldValue := -FieldValue;
  except
    MessageBeep(mb_IconAsterisk);
  end;
end;

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

Return to Front Page

CurrEdit Modifications

Contributed by Bob Osborn - 100033,326

If you haven't already changed the unit, may I humbly offer you my modifications to the KeyDown method as suggestions. This isn't foolproof (a determined user can still paste invalid characters) and not very well tested way beyond bedtime, but for what it's worth ..

```
      procedure TCurrencyEdit.KeyPress(var Key: Char);
        {I've added a field <FDecimalPlaces: Word> for my use.}
      var
        S: String;
      begin
        inherited KeyPress(Key);
        if (not (Key in [Char(vk_Back),'0'..'9','.','-']))  {Allow backspace
to edit}
                 or ((Key = '.') and (FDecimalPlaces = 0))  {Integers}
                 or ((Key = '.') and (Pos('.', Text) <> 0)) {too many decimal
points}
        then
          begin
            Key := #0;
            Exit;
          end;

        if Key <> Char(vk_Back) then
          begin
           {S is a model of Text if we accept the keystroke.  Use SelStart and
           SelLength to find the cursor (insert) position.}
          S := Copy(Text, 1, SelStart) + Key  +
               Copy(Text, SelStart + SelLength + 1, Length(Text));
          if ((Pos('.', S) > 0)
               and (Length(S) - Pos('.', S) > FDecimalPlaces))  {too many
decimal places}
               or ((Key = '-') and (Pos('-', Text) <> 0))       {only one
minus...}
               or (Pos('-', S) > 1)                             {... and only
at beginning}
          then begin
            Key := #0;
          end;
        end;
      end;
```

Return to Front Page

INI File Example

Contributed by Norman McIntosh - 72164,203

Here is a way to allow user's to specify colors in an INI file. The INI file should have a section that looks like the following :

```
[COLORS]
Act=Green
CTS=Yellow
DNS=Red
Sld=Grey
Exp=Blue
Wth=Purple
Current=Aqua
```

The first thing to do is define some constants that will make the code easier to maintain. Also, define the variables that will hold the color results.

```
const
  INI_FILE          = 'AFB.INI';
  COLOR_SECTION     = 'COLORS';
  ACTIVE_COLOR      = 'ACT';
  SOLD_COLOR        = 'SLD';
var
  ActiveStatus, SoldStatus : TColor;
```

Now define the "procedure" that will read the INI file and set the variables based on the values in the INI file.   Notice that the WhatColor function gets passed a default. This is used if the INI file is not present, the Section name is not present, or the Value is not present in the INI file. The WhatColor function will call the "ReadString" function to get the value from the Ini file. Make sure it is UpCase'd and then determine the color specified. If we can't determine the color, default to Black.

```
procedure GetColors;
begin
  ActiveStatus := WhatColor(ACTIVE_COLOR,'Green');
  SoldStatus   := WhatColor(SOLD_COLOR,'Gray');
end;

function WhatColor(name, Default: String) : TColor;
var
  val : String;
  myIni: TIniFile;
begin
  myIni := TIniFile.Create(INI_FILE);
  val := UpperCase(myIni.ReadString(COLOR_SECTION,name,default));
  myIni.Free;
  if val = 'MAROON'  then WhatColor := clMaroon  else
  if val = 'GREEN'   then WhatColor := clGreen   else
  if val = 'OLIVE'   then WhatColor := clOlive   else
  if val = 'NAVY'    then WhatColor := clNavy    else
  if val = 'PURPLE'  then WhatColor := clPurple  else
  if val = 'TEAL'    then WhatColor := clTeal    else
  if val = 'GRAY'    then WhatColor := clGray    else
  if val = 'SILVER'  then WhatColor := clSilver  else
  if val = 'RED'     then WhatColor := clRed     else
  if val = 'LIME'    then WhatColor := clLime    else
  if val = 'YELLOW'  then WhatColor := clYellow  else
  if val = 'BLUE'    then WhatColor := clBlue    else
```

```
    if val = 'FUCHSIA' then WhatColor := clFuchsia else
    if val = 'AQUA'    then WhatColor := clAqua    else
    if val = 'WHITE'   then WhatColor := clWhite   else
                            WhatColor := clBlack;
  end;
```

The last thing you must do is make sure that your "uses" section has the IniFiles declared. I am sure you can think of other applications to use this sort of routine with. Have fun.

Tips & Tricks
Return to Front Page

Progress Bar Bug

Contributed by Ernst Genaehr - 100315,3574

In the "Samples" Section you find a "Gauge" component which works fine except for *very* high values. Here's how to reproduce the error:

1. Create a New Project (Just a plain form) and drag a Gauge Component onto it.
2. Set the **MaxValue** value to 99,000,000 in the Object Inspector.
3. Set the **Progress** value to 40,000,000.
4. Watch the Progress bar: it shows -2%. Neat huh?

The values are just examples. You can try other high values to get the same effect. Here's how to correct the problem (the source file is GAUGES.PAS in the directory DELPHI\SOURCE\SAMPLES):

Just change line 90 (function SolveForY) from:

```
else SolveForY := Trunc(  (X*100) / Z );
```

to:

```
else SolveForY := Trunc( X / Z * 100);
```

This should improve things... Of course you will need to recompile the Component Library.

Editors Note: *Granted this is a pretty obscure bug, but Ernst is correct in that the division should have been done first to prevent the variable overflow.*

When Things Go Wrong
Return to Front Page

Floating Palette

You can create a floating palette with proper behavior by doing

```
    Style := Style or ws_Overlapped;
    WndParent := Form1.Handle;
```

in the component's CreateParams method.

Tips & Tricks
Return to Front Page