

# The Unofficial Newsletter of Delphi Users

by Robert Vivrette - CIS: 76416,1373

The **Delphi** community continues to grow by leaps and bounds. I consider myself fortunate that I am able to spend a lot of time working with Delphi and to learn about all of its features. Every day I get the opportunity to talk with other developers about the benefits of Delphi over other programming languages. Sometimes I even drag them over to my computer and put on my Delphi "dog and pony" show. It's amazing the reactions you get from people when you whip together a program in 4 or 5 minutes that it would have taken hours to do in VB or C.

At work, I recently finished up a set of components that allow access to the various facilities of **Banyan Vines**. One is a **SetDrive** component that adds a drive letter to your computer that is mapped to a network service. The best part is that, for the user of the component, it requires **zero** lines of code. You just drop the component on your program, set a few properties and you're done. Now, when the program runs, it will connect to the network service, and when it terminates, it disconnects. Very Cool... I will likely discuss some of the technical aspects of this next issue.

But, enough about me... Let's get on with it!

[What's New](#)

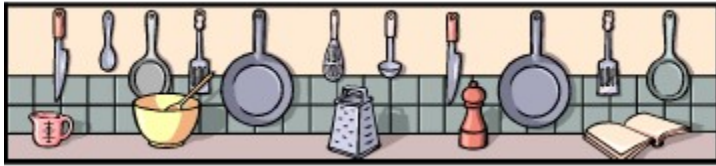
[Delphi On The Internet](#)

[FontViewer Sample Application](#)

[Cooking Up Components](#)

[Tips & Tricks](#)

[When Things Go Wrong](#)

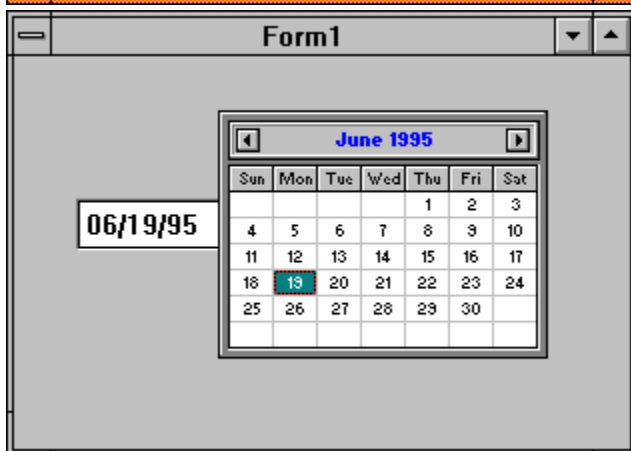
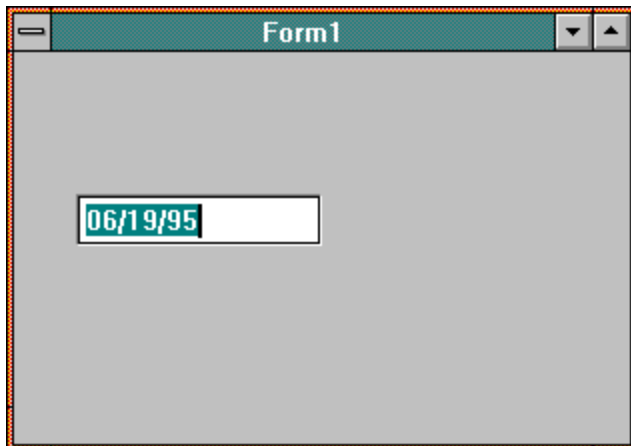


## Cooking Up Components

By Robert Vivrette

I know I promised last issue that I would update the **TStatusBar** component, but I figured I would take a break on that one and get back to it next issue. Hope this does not ruin too many people's day.

The component for this issue is a **TDateEdit** component. I actually created this component a while ago, but decided to dust it off a bit and share it with everyone. Essentially, it is a normal TEdit field for inputting dates. The fancy part is that when you double click on the field, you get a pop-up calendar that you can navigate through to pick a date. The calendar has arrow buttons allowing you to move a month ahead or back, and it also responds to arrow keys from the keyboard. If you double click on the banner of the calendar, it will return you to today's date. Double clicking on a date will select it and dismiss the calendar. Hitting escape will put away the calendar and leave the contents of the edit field intact.



There are properties that control date validation as well as for controlling the placement of the pop-up calendar. It is smart enough to pop-up completely on the screen even if the date edit control is way off to one side of the screen.

The calendar grid portion of the popup form uses the TCalendar sample component that came with Delphi, so make sure that you have not removed it from your system.

[TDateEdit Source Code](#)

[Calendar Popup Source Code](#)

[Calendar Popup Form](#)

[Return to Front Page](#)

## The Unofficial Newsletter of Delphi Users - Issue #5 - June 26th, 1995

This is the source code for the TDateEdit component. Simply choose the Copy command from the Edit Menu above and select all the area of source code and copy it to the clipboard. Then save the clipboard into a file called DATEEDIT.PAS. You will need to perform a similar feature for the source code and form definition of the DATE2 unit/form as well.

```
unit DateEdit;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Menus, StdCtrls, Grids, Date2;

type
  TPlacement = (cpAbove, cpBelow, cpLeft, cpRight, cpOnTop, cpOffsets, cpAuto);

  TDateEdit = class(TCustomEdit)
  private
    FOffsetX : Integer;
    FOffsetY : Integer;
    FPlacement : TPlacement;
    FDateFormat : String;
    FValidateInput : Boolean;
    FRequired: Boolean;
    TestDate: TDateTime;
    procedure CMEExit(var Message: TCMEExit); message CM_EXIT;
  protected
    procedure Db1Click; override;
    procedure KeyPress(var Key: Char); override;
  public
    constructor Create(AOwner: TComponent); override;
    function ValidDate: Boolean;
    function FieldValidated: Boolean;
  published
    property AutoSelect;
    property AutoSize;
    property BorderStyle;
    property Color;
    property Ctl3D;
    property DateFormat: string read FDateFormat write FDateFormat;
    property Enabled;
    property Font;
    property HideSelection;
    property ParentColor;
    property ParentCtl3D;
    property ParentFont;
    property ParentShowHint;
    property OffsetX: Integer read FOffsetX write FOffsetX default 0;
    property OffsetY: Integer read FOffsetY write FOffsetY default 0;
    property Placement: TPlacement read FPlacement write FPlacement default cpOnTop;
    property PopupMenu;
    property ReadOnly;
    property RequiredField: Boolean read FRequired write FRequired default False;
    property ShowHint;
    property TabOrder;
    property Text;
    property ValidateInput: Boolean read FValidateInput write FValidateInput default
True;
    property Visible;
    property OnChange;
    property OnClick;
```

```

    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', [TDateEdit]);
end;

(*****
TDateEdit Object
*****)

constructor TDateEdit.Create(AOwner: TComponent);
begin
    inherited Create(AOwner);
    FOffsetX := 0;
    FOffsetY := 0;
    FPlacement := cpOnTop;
    FDateFormat := 'mm/dd/yy';
    FValidateInput := True;
    FRequired := False;
end;

procedure TDateEdit.DblClick;
var
    Pt : TPoint;
    Shift : Byte;
begin
    inherited DblClick;
    try
        InitialDate := StrToDate(Text);
    except
        InitialDate := 0;
    end;
    Application.CreateForm(TCalForm, CalForm);
    case FPlacement of
        {Move according to the offsets}
        cpOffsets : Pt := ClientToScreen(Point(OffsetX, OffsetY));
        {Move up the height of the calendar}
        cpAbove : Pt := ClientToScreen(Point(0, -CalForm.Height));
        {Move down the height of the DateEdit control}
        cpBelow : Pt := ClientToScreen(Point(0, Height));
        {Move left the width of the calendar}
        cpLeft : Pt := ClientToScreen(Point(-CalForm.Width, 0));
        {Move Right the width of the DateEdit control}
        cpRight : Pt := ClientToScreen(Point(Width, 0));
        {Center on top of the DateEdit control}
        cpOnTop : Pt := ClientToScreen(Point(-(CalForm.Width-Width) div 2,

```

```

        -(CalForm.Height-Height) div 2));
    cpAuto    : Pt := ClientToScreen(Point(0,Height));
end;
{If Ctl3D is on, shift it by an additional pixel up and to the left}
if Ctl3D and (FPlacement <> cpOffsets) then Shift := 1 else Shift := 0;
{Place the popup}
CalForm.Left := Pt.X-Shift;
CalForm.Top := Pt.Y-Shift;
{Make sure the popup appears on the screen fully}
if CalForm.Left < 0 then CalForm.Left := 0;
if CalForm.Top < 0 then CalForm.Top := 0;
if CalForm.Left + CalForm.Width > Screen.Width then
    CalForm.Left := Screen.Width - CalForm.Width;
if CalForm.Top + CalForm.Height > Screen.Height then
    CalForm.Top := Screen.Height - CalForm.Height;
{Show the popup - if exit is normal, assign date value to field}
if CalForm.ShowModal = mrOK then
    Text := FormatDateTime(FDateFormat,DateSelection);
{Destroy the popup and release its memory}
CalForm.Release;
{Select all the text on exit}
SelectAll;
end;

function TDateEdit.ValidDate: Boolean;
begin
    try
        TestDate := StrToDate(Text);
        ValidDate := True;
    except
        TestDate := 0.0;
        ValidDate := False;
    end;
end;

function TDateEdit.FieldValidated: Boolean;
begin
    if (Length(Text) < 1) and Not RequiredField then
        begin
            FieldValidated := True;
            exit;
        end;
    if ValidateInput and not ValidDate then
        begin
            MessageDlg('Please enter a valid date.',mtError,[mbOK],0);
            SetFocus;
            FieldValidated := False;
        end
    else
        begin
            Text := FormatDateTime(FDateFormat,TestDate);
            FieldValidated := True;
        end;
    end;
end;

procedure TDateEdit.CMExit(var Message: TCMExit);
begin
    if FieldValidated then inherited;
end;

procedure TDateEdit.KeyPress(var Key: Char);
var
    Shift    : TDateTime;

```

```

begin
  {Validate and select text if enter is pressed}
  if Key = #13 then
    if not FieldValidated then
      Key := #0
    else
      if AutoSelect then SelectAll;
    Shift := 0.0;
    if Key = '+' then Shift := 1.0;
    if Key = '-' then Shift := -1.0;
    {Increment/Decrement day and select text if plus/minus is pressed}
    if Shift <> 0 then
      begin
        try
          TestDate := StrToDate(Text);
        except
          Key := #0;
        end;
        if Key <> #0 then
          begin
            Text := FormatDateTime(FDateFormat,TestDate + Shift);
            if AutoSelect then SelectAll;
          end;
        end;
      end;
    if Not (Key in ['0'..'9','/','#8]) then Key := #0;
    inherited KeyPress(Key);
  end;

end.

```

[Return to Cooking Up Components](#)

[Return to Front Page](#)









## **When Things Go Wrong**

[StrToTime Function](#)

[Image Editor Bugs](#)

[Object Alignment Bug](#)

[Return to Front Page](#)

## The Unofficial Newsletter of Delphi Users - Issue #5 - June 26th, 1995

This is the source code for the TDateEdit component. Simply choose the Copy command from the Edit Menu above and select all the area of source code and copy it to the clipboard. Then save the clipboard into a file called DATE2.PAS. You will need to perform a similar operation for the form definition for DATE2 as well.

```
unit Date2;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Grids, ExtCtrls, Buttons, Calendar;

type
  TCalForm = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    PrevMonthBtn: TSpeedButton;
    NextMonthBtn: TSpeedButton;
    MonthYear: TPanel;
    BorderEdge: TShape;
    CalGrid: TCalendar;
    procedure ShowMonth;
    procedure NextMonthBtnClick(Sender: TObject);
    procedure PrevMonthBtnClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure CalGridDblClick(Sender: TObject);
    procedure MonthYearDblClick(Sender: TObject);
    procedure FormKeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  CalForm: TCalForm;
  DateSelection : TDateTime;
  InitialDate   : TDateTime;

implementation

{$R *.DFM}

(*****
(* Calendar Wrapper *)
*****)

procedure TCalForm.ShowMonth;
begin
  MonthYear.Caption := FormatDateTime('mmm yyyy', CalGrid.CalendarDate);
end;

procedure TCalForm.NextMonthBtnClick(Sender: TObject);
begin
  CalGrid.NextMonth;
  ShowMonth;
end;
```

```

procedure TCalForm.PrevMonthBtnClick(Sender: TObject);
begin
    CalGrid.PrevMonth;
    ShowMonth;
end;

procedure TCalForm.FormCreate(Sender: TObject);
begin
    if InitialDate > 0 then
        CalGrid.CalendarDate := InitialDate
    else
        CalGrid.CalendarDate := Now;
        ShowMonth;
end;

procedure TCalForm.CalGridDblClick(Sender: TObject);
begin
    DateSelection := CalGrid.CalendarDate;
    ModalResult := mrOK;
end;

procedure TCalForm.MonthYearDblClick(Sender: TObject);
begin
    CalGrid.CalendarDate := Now;
    ShowMonth;
end;

procedure TCalForm.FormKeyUp(Sender: TObject; var Key: Word;
    Shift: TShiftState);
begin
    case Key of
        VK_RETURN    : ModalResult := mrOK;
        VK_ESCAPE    : ModalResult := mrCancel;
        VK_TAB       : begin
            if Shift = [ssShift] then
                CalGrid.PrevMonth
            else
                CalGrid.NextMonth;
                ShowMonth;
            end;
        VK_ADD       : begin
            CalGrid.CalendarDate :=
                CalGrid.CalendarDate + 1.0;
            ShowMonth;
            end;
        VK_SUBTRACT  : begin
            CalGrid.CalendarDate :=
                CalGrid.CalendarDate - 1.0;
            ShowMonth;
            end;
    end;
end;

begin
    InitialDate := 0;
    DateSelection := 0;
end.

```

[Return to Cooking Up Components](#)

[Return to Front Page](#)



## Delphi-Related Internet Sites

Delphi sites have been popping up all over the place on the internet and I thought that it was time to start reporting some of these locations here in the newsletter. In addition, the **Unofficial Newsletter of Delphi Users** now has an Internet home as well. Mark Cooke has converted the back issues of the newsletter into pages on his Web site, and will be doing so with all future issues as well. Mark also has the Help files themselves available on an ftp server as mentioned below. Here is the pertinent information:

<b>Mark Cooke's Home Page</b>	<a href="http://www.doit.com/mcooke/">http://www.doit.com/mcooke/</a>
<b>The Delphi Source</b>	<a href="http://www.doit.com/mcooke/delphi/home.html">http://www.doit.com/mcooke/delphi/home.html</a>
<b>The Delphi Newsletter</b>	<a href="http://www.doit.com/mcooke/delphi/dn0315.htm">http://www.doit.com/mcooke/delphi/dn0315.htm</a>
<b>FTP Site</b>	<a href="http://www.doit.com/mcooke/delphi/ftp.htm">http://www.doit.com/mcooke/delphi/ftp.htm</a>
<b>Help Files</b>	<a href="ftp://ftp.doit.com/mcooke/delphi/">ftp://ftp.doit.com/mcooke/delphi/</a>

In addition, the following are addresses of other internet sites relating to Delphi:

<b>Grumpfish Home Page</b>	<a href="http://www.teleport.com/~grump">http://www.teleport.com/~grump</a>
<b>Grumpfish Borland Delphi Support</b>	<a href="http://www.teleport.com/~grump/delphi.htm">http://www.teleport.com/~grump/delphi.htm</a>
<b>The Aquarium Information Page</b>	<a href="http://www.teleport.com/~grump/aquarium.htm">http://www.teleport.com/~grump/aquarium.htm</a>
<b>The Delphi Aquarium Announcement</b>	<a href="http://www.teleport.com/~grump/delphiaq.htm">http://www.teleport.com/~grump/delphiaq.htm</a>
<b>The Delphi WWW Forum</b>	<a href="http://www.pennant.com/delphi/hn/dconn.html">http://www.pennant.com/delphi/hn/dconn.html</a>
<b>Coriolis Group Delphi EXplorer</b>	<a href="http://www.coriolis.com/coriolis/whatsnew/delphi.htm">http://www.coriolis.com/coriolis/whatsnew/delphi.htm</a>
<b>Anders Ollsen Delphi Corner</b>	<a href="http://www.it.kth.se/~ao/">http://www.it.kth.se/~ao/</a>
<b>Wild Thing Cool Delphi Stuff</b>	<a href="http://www.teleport.com/~cwhite/wilddelphi.html">http://www.teleport.com/~cwhite/wilddelphi.html</a>
<b>Delphi WINSOCK Project</b>	<a href="http://www.teleport.com/~sig/delphi.html">http://www.teleport.com/~sig/delphi.html</a>
<b>Bibliography of Delphi Books &amp; Articles</b>	<a href="http://www.iscinc.com/dugbib.html">http://www.iscinc.com/dugbib.html</a>
<b>Delphi Tech Support Home Page</b>	<a href="http://loki.borland.com:8080/">http://loki.borland.com:8080/</a>
<b>CMP Publications</b>	<a href="http://techweb.cmp.com:80/techweb/docs/list-o-pubs.html">http://techweb.cmp.com:80/techweb/docs/list-o-pubs.html</a>
<b>Delphi Components</b>	<a href="http://super.sonic.net/ann/delphi/max/">http://super.sonic.net/ann/delphi/max/</a>
<b>C.I.U.P.K.C. Software Headquarters</b>	<a href="http://www.webcom.com/~kilgalen/welcome.html">http://www.webcom.com/~kilgalen/welcome.html</a>
<b>RMC Delphi Page</b>	<a href="http://sunsite.icm.edu.pl/~robert/delphi//">http://sunsite.icm.edu.pl/~robert/delphi//</a>
<b>Borland Delphi Components Page</b>	<a href="http://www.neosoft.com/~startech/delphi/delphi.htm">http://www.neosoft.com/~startech/delphi/delphi.htm</a>
<b>Delphi Frequently Asked Questions</b>	<a href="http://www.mhn.org/delphi.faq">http://www.mhn.org/delphi.faq</a>
<b>Michael's Delphi Home-Page</b>	

<http://linux.rz.fh-hannover.de/~holthoefel/delphi.html>

If you have additional Delphi-related internet information, please let me know!

[Return to Front Page](#)

## **Delphi Unleashed**

**A brief review by Jay J. Garnett - 76317,461**

**Title:** *Delphi Unleashed*, 930 pgs.  
**Author:** Charles Calvert  
**Publisher:** Sams Publishing  
**ISBN #:** 0-672-30499-6

We all have experienced the incredible lack of documentation Borland provides with the Delphi Development Platform. One of the first things I did was search all the online sources to see what everyone was using. I was surprised to find that there were very few books actually available. Yes, there were a lot in the book mill, but only a handful had actually made it out the door.

I was fortunate to have come across a reference to Charles Calvert's book, *Delphi Unleashed*. Its original title was to be "Master Delphi" and I knew that's what I wanted to be. When I called to order the book, I was told about the name change. No problem. It wouldn't hurt for me to become "unleashed".

The book arrived about a week and a half later at a time when I **really** needed it. I was working on a fairly complicated section of code and couldn't find answers to some questions I had. Delphi Unleashed to the rescue!

Delphi Unleashed is valid for the whole range of Delphi programmers: beginner to expert. The chapters are logically laid out starting with basic concepts and building on it chapter by chapter.

You could probably use this book as your single reference on Delphi. It covers everything from the basics of setting the Delphi IDE to creating your own Components. There are chapters covering stuff like:

- Delphi IDE
- Eccentricities of Programming for the Windows environment
- Basics of Delphi Programming
- Variable and Types
- Repetition Structures
- Working with Arrays
- Strings and Text Files
- Pointers, Linked Lists and Memory issues
- Programming with the Visual Database Tools
- SQL Language
- OOP Programming Techniques and Topics:
  - Encapsulation, Inheritance, Polymorphism
- Using the Delphi Multimedia Tools
- Creating a DLL
- Delphi OLE and DDE
- OWL Integration with Delphi

The book also includes a handy CD-Rom filled with all the books example code. The CD is sub-divided in chapters corresponding to the book's chapters. I have used it the CD quite a bit.

I have tried to think of something negative I can say about Delphi Unleashed so as to present a balance review, but I just couldn't think of anything. This book is an unqualified BUY.

[Return to What's New](#)

[Return to Front Page](#)

## **Delphi Power Toolkit**

**Title:** Delphi Power Toolkit, 800 pages, 400 illustrations  
**Author:** Harold Davis  
**Publisher:** Ventana Press  
**ISBN#:** 1-56604-292-5  
**Price:** \$49.95 U.S.  
**Available:** October

*"Delphi is Visual Basic done right."*

-- PC Computing

### ***Cutting-Edge Tools & Techniques for Programmers***

Delphi is the hot, new, next-generation programming Windows tool that everyone is talking about as the way to rapidly build applications. This new Power Toolkit helps users master the Delphi Integrated Development Environment (IDE) and build blazingly fast, stand-alone, compiled programs. Topics covered include Object Pascal, converting Visual Basic applications, Object-Oriented Programming (OOP), using the Visual Component Library (VCL), creating Delphi components (DCUs) and run-time libraries (DLLs), using VBXs with Delphi, making VBXs in Delphi, database development, and multimedia programming in Delphi.

CD-Rom: All source code from the sample programs in the book plus exciting sample Delphi components (DCUs) and utilities from third-party vendors.

Online Companion: Free utilities and links to Borland and other developer resources on the Internet.

[Return to What's New](#)

[Return to Front Page](#)



## What's New?

Each month, this column will bring additional information about new things available for Delphi. This includes books, magazines, and other publications, as well as programming tools and libraries. I am going to modify my policy a bit about "advertising" as follows: If you would like to **briefly** mention your publication/product here in the newsletter, send a short textual description to me, and I will be happy to include it in the next issue. I will not charge for this service primarily because I am a nice guy and I want others to hear about all the cool Delphi products that are out there.

As always, I welcome reviews of Delphi publications and tools. However, if you are the author/creator of such a product and you would like it reviewed, in some detail, please contact me directly.

I am currently looking for reviews of each of the principal Delphi periodicals, namely: *Delphi Informant*, *The Delphi Magazine*, *Delphi Developer*, and *Delphi Developer's Journal*. Please contact me if you would like to review one.

**Electronic Newsletter Announcement:** [Delphi Aquarium](#)

**Book Announcement:** [The Delphi Power Toolkit](#)

**Book Review:** [Delphi Unleashed](#)

**Recent Issues:** [Delphi Informant](#)

**Recent Issues:** [Delphi Developer](#)

[Return to Front Page](#)





## **[Tips & Tricks](#)**

[Faster String Loading](#)

[Connecting to a Database](#)

[A Second Helping: Loading a Custom Cursor](#)

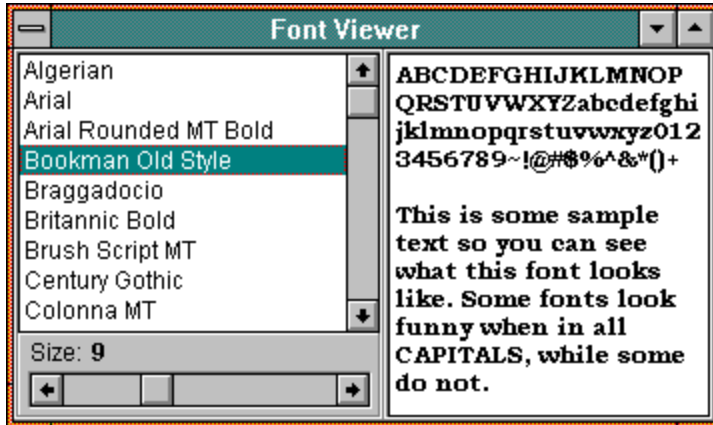
[Loading a Bitmap from a Resource File](#)

[Return to Front Page](#)

## Font Viewer Application

*by Robert Vivrette*

A while back, I needed a quick and dirty application that would display all fonts I had available in my system. I put together a simple application that did just that and thought I would share it.



[Font Viewer Source](#)

[Font Viewer Form Definition](#)

[Return to Front Page](#)

## Font Viewer Source

```
unit Fontunit;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, Printers, SlideBar, ExtCtrls;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    ListBox1: TListBox;
    Panel2: TPanel;
    Memo1: TMemo;
    Panel3: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    ScrollBar1: TScrollBar;
    procedure FormCreate(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure ScrollBar1Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Clear;
  ListBox1.Sorted := True;
  ListBox1.Items := Screen.Fonts;
  ListBox1.ItemIndex := 0;
  Memo1.Clear;
  Memo1.Font.Name := ListBox1.Items[ListBox1.ItemIndex];
  Memo1.Lines.Add('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789~!@#$
%^&*()+');
  Memo1.Lines.Add('');
  Memo1.Lines.Add('This is some sample text so you can see what this font looks like.
');
  Memo1.Lines.Add('Some fonts look funny when in all CAPITALS, while some do not.');
```

```
  ScrollBar1.Position := Memo1.Font.Size;
end;

procedure TForm1.ListBox1Click(Sender: TObject);
begin
  Memo1.Font.Name := ListBox1.Items[ListBox1.ItemIndex];
end;

procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
  Memo1.Font.Size := ScrollBar1.Position;
end;
```

```
Label2.Caption := IntToStr(ScrollBar1.Position);  
end;  
  
end.
```

[Return to Font Viewer Article](#)

[Return to Front Page](#)

## Font Viewer Form Definition

```
object Form1: TForm1
  Left = 206
  Top = 100
  Width = 431
  Height = 303
  Caption = 'Font Viewer'
  Font.Color = clWindowText
  Font.Height = -12
  Font.Name = 'Arial'
  Font.Style = []
  PixelsPerInch = 96
  OnCreate = FormCreate
  TextHeight = 15
object Panel1: TPanel
  Left = 0
  Top = 0
  Width = 185
  Height = 276
  Align = alLeft
  BevelInner = bvLowered
  TabOrder = 0
  object ListBox1: TListBox
    Left = 2
    Top = 2
    Width = 181
    Height = 230
    Align = alClient
    ItemHeight = 15
    TabOrder = 0
    OnClick = ListBox1Click
  end
  object Panel3: TPanel
    Left = 2
    Top = 232
    Width = 181
    Height = 42
    Align = alBottom
    TabOrder = 1
    object Label1: TLabel
      Left = 6
      Top = 2
      Width = 26
      Height = 15
      Caption = 'Size:'
    end
    object Label2: TLabel
      Left = 37
      Top = 2
      Width = 34
      Height = 15
      AutoSize = False
      Font.Color = clWindowText
      Font.Height = -12
      Font.Name = 'Arial'
      Font.Style = [fsBold]
      ParentFont = False
    end
  end
  object ScrollBar1: TScrollBar
    Left = 6
```

```
        Top = 20
        Width = 170
        Height = 17
        LargeChange = 3
        Max = 24
        Min = 2
        Position = 12
        TabOrder = 0
        OnChange = ScrollBar1Change
    end
end
end
object Panel2: TPanel
    Left = 185
    Top = 0
    Width = 238
    Height = 276
    Align = alClient
    BevelInner = bvLowered
    TabOrder = 1
    object Mem1: TMemo
        Left = 2
        Top = 2
        Width = 234
        Height = 272
        Align = alClient
        TabOrder = 0
    end
end
end
end
```

[Return to Font Viewer Article](#)

[Return to Front Page](#)

## **Aquarium Publications**

The Aquarium Publications are electronic technical journals catering to the specific needs to today's Windows/DOS database developers. Aquarium editions include CA-Clipper, CA-Visual Objects, and Borland Delphi.

The trademark of The Aquarium Publications is understandable and usable examples that will enhance your productivity immediately. We also feature instructive discussions of other relevant topics, and bring you news of developers conferences, product updates, bug reports. Darren Forcier's patented in-depth product reviews will keep you up-to-the-minute with detailed information about the latest add-on products for your favorite development platform.

### Who Writes For The Aquarium?

#### Aquarium Features

### **Delphi Aquarium**

Grumpfish, Inc. is proud to announce our new Delphi edition of the Aquarium Publications! The first issue is August 1995, and will be distributed free of charge at the Sixth Annual Borland Developers Conference.

Compare Delphi Aquarium to other print-based Delphi publications, which either give you just a couple dozen pages or are chock full of advertising. Each issue of Delphi Aquarium includes between 40 and 80 pages of solid information with no filler and no ads!

Here are some of the topics we will discuss in our upcoming issues.

- \* The Delphi IDE
- \* Creating Context Sensitive Help
- \* TNotebook/TTabbed Notebook
- \* The Borland Database Engine
- \* Implementing Drag and Drop
- \* Creating Components
- \* Making Standard Components Data Aware
- \* Implementing Undo
- \* TStream and TWriter
- \* Delphi for VB/PowerBuilder/CA-Clipper programmers
- \* Writing Delphi DLLs
- \* Debugging
- \* Screen Savers
- \* Object Pascal

The regular price for a one-year subscription to Delphi Aquarium will be \$159 US. Grumpfish is offering a special price for Delphi Aquarium charter subscribers -- sign on before June 30, 1995, and pay only \$99! Be on the cutting edge of this exciting new technology with Delphi Aquarium!

To subscribe to any of The Aquarium Publications, contact Mary Gries at Grumpfish, Inc., via voice (800-367-7613), CompuServe (70673,355), or E-mail ([grump@teleport.com](mailto:grump@teleport.com)).

Since The Aquarium's inception in June 1990, we have gotten many favorable comments from our subscribers:

#### Comments From Our Subscribers

[Return to What's New](#)

[Return to Front Page](#)

Our authors include the following authorities: Joe Booth and Greg Lief (co-authors of the books "Visual Objects: A Developer's Guide", "Clipper 5.2: A Developer's Guide", and "Network Programming in CA-Clipper 5.2"), Darren Forcier, Ted Means, Clayton Neff, Mark Lukianchuk, and Ted Blue. All of these authors have spoken at numerous U.S. and international developers conferences and user groups, and have a writing style that will entertain and educate you.



Unlike print-based publications, The Aquarium's table of contents is completely additive. The Aquarium also includes searching facilities (by date, author, and keyword). Combined with the additive table of contents, you can quickly locate all relevant Aquarium articles without having to dig through a pile of back issues.

Windows editions of The Aquarium are delivered in the form of Windows Help Files. The Aquarium takes full advantage of that medium, including numerous hypertext links to articles in the same and previous issues; pop-up glossary definitions; graphics; screen captures; easy Clipboard access; and much more.

DOS editions of The Aquarium (such as CA-Clipper) lets you view an article and its accompanying source code at the same time. If you want to test out an example, you can press the Tab key to move to the source code window and output it to a file. By pressing another key, you can shell out from Aquarium to DOS, where you can compile, link, and run the sample.

*"I subscribe to all the major Clipper-related journals, and The Aquarium educates and entertains me the most" -- C. G.*

*"Mere words cannot describe the value I derive from my Aquarium subscription" -- D. W.*

*"The Aquarium is the best money I have ever spent. It has really kept me up on the latest techniques" -- S. N.*

*"The Aquarium is indispensable -- I can't imagine working without it" -- B.S.*

*"Being the only programmer in my shop can be intimidating with no one to turn to. Now I turn to The Aquarium. If my company dropped its subscription, I would continue my subscription out of my own pocket!" -- S.H.*

*"The Aquarium is simply superb... it is worth a thousand times what we pay for it" -- H. H.*

*"The Aquarium is a reference system that no-one should be deprived of. It is a great buy for the beginning to advanced developer" -- Clipper User Group of Orange County*

*"The Aquarium's articles present invaluable timely information. Your editorial style is easy to read and presented for both novice and advanced users" -- B. K.*

*"The Aquarium is one of the most useful and informative tools I have ever used" -- P. F.*

*"It gives me great pleasure to read the inspiring and down to earth articles I find in each month's Aquarium." -- F. L.*

*"I used to subscribe to other Clipper newsletters which frequently disappointed me. For me, The Aquarium is now the standard by which all technical journals are judged." -- F. D.*

*"The Aquarium is fun to read and very helpful to my Clipper learning" -- T.T.*

*"Since I've been reading The Aquarium my programming skills have improved at least tenfold" -- T. H.*

*"The Aquarium has been an endless resource for me to draw upon, and both have helped me acquire knowledge not available from any manuals. Your real life programs and functions are great" -- G. S.*

*"I subscribe to four other xBase-related publications, and The Aquarium is the most informative. The expertise of your authors is second to none" -- R. D.*

*"I used to subscribe to several Clipper publications, and have made the decision to keep only one: The Aquarium." -- K. W.*

*"If not for The Aquarium, I would never have taken my first timid steps into the world of CA-Clipper 5.0" -- B. K.*

*"The Aquarium is the best subscription I've ever gotten" -- K. B.*

*"The Aquarium articles and source code have been lifesavers again and again" -- M. T.*

The 'StrToTime' function on-line help says that it raises EConvertError when an invalid time is converted. This is apparently not correct. No error is raised when converting times like '12:88' or '57:20'. The EConvertError IS raised when converting invalid dates, however.

This becomes useful for validating dates: rather than checking each individual part of a date for validity, use StrToDate() and handle the conversion error as the main result of the conversion, throwing away the actual result:

```
try
  StrToDate(Value);
except
  on EConvertError do MessageDlg('Invalid date', mtError, mbOK, 0);
end;
```

***Reported by Thomas Hill - hillt@sill-emh.army.mil***

## Image Editor Bugs

I think Delphi is one of the most stable programming environments available. That's why it disturbs me so much that Borland included the **Image Editor** in with it. When it was first released, Image Editor was riddled with bugs.

Now... I realized that all new software is entitled to a few bugs, so I waited for the patches to come out. When the first patch was made available through CompuServe and on various web sites, I eagerly snatched up a copy. I noticed during the patching procedure that IMAGEDIT.EXE was one of the files patched. *"At last!"*, I thought, *"We can now use the Image Editor!"*

Sorry... Wishfull thinking... In my opinion, there are now more bugs in the Image Editor than there were to begin with. Now that this is "post-patch", I feel justified in publicly **whining** about these bugs. Below is just my **short** list:

1. The CTRL-C short cut key does not work while editing an image. It closes the form instead.
2. Double-clicking on the selection tool selects the entire image (as it should). However, the cut, copy, and paste options are still dimmed in the edit menu and are unavailable.
3. If you select an area of a bitmap and then use the arrow keys to try and move the selected area, you will only get to move one pixel. After the move, the arrow key shifts the focus away from the editor so all further arrow keys no longer move the selection.
4. There is some very unusual resizing behavior, particularly when the editing window is maximized, and the main window is reduced. Scroll bars will get resized over when they should be still visible, for example.
5. When working in 256 color mode, it is very difficult to reliably make 16 color bitmaps. The color palette will often remap colors (particularly light and dark grays).
6. If you open an existing DCR file, rename or add a resource in it, and then do a "Save As" to give the file a different name, the resulting file will be corrupted. When Delphi tries to compile the DCR file into the component library it will report a "Disk Full" error. You also will not be able to reload the DCR file into the Image Editor.
7. The File|Open dialog box does not retain its filter settings. The main application window should also save its size and location in a INI file, so it launches with the same size and location as the last time you used it. This may seem like a small thing, but it saves countless resizes and moves of the Image Editor window every time it is launched. It would take the Borland developer of the program less than 10 minutes to implement this feature. PS: Borland - While you're at it, do the same thing to Resource Editor (which I suspect the Image Editor was born from, since it is inheriting most of RE's bugs as well!).
8. About 50% of the time, if you load RES file with cursors in them, the cursors are not present when you click the CURSORS tab.
9. The flood fill often does not perform correctly. For example: Draw a black line down the right most column of the image, and then draw a green line horizontally across the middle up against (but not overwriting) the black line (so the two lines form a "T" on its side). Now select the blue color, and the flood tool, and click somewhere on the green line. The blue flood will creep part way into the black line.
10. When you paste (CTRL-V works) the copied image, the colors of the copy sometimes do not match the original.

[Return to When Things Go Wrong](#)

[Return to Front Page](#)

Use a temporary StringList to speed up some operations... Instead of this:

```
ListBox1.Clear;  
Ini := tIniFile.Create( 'C:\R\TEST.INI' );  
ReadSectionValues( 'System', ListBox1.Items );  
Ini.Free;
```

Try this...

```
StrList := tStringList.Create;  
ListBox1.Clear;  
Ini := tIniFile.Create( 'C:\R\TEST.INI' );  
ReadSectionValues( 'System', StrList );  
Ini.Free;  
ListBox1.Items.Assign( StrList );  
StrList.Free;
```

In a test case with 630 items, the former approach took 31 seconds while the latter took just under 5 seconds!! Things get even worse if you use a Memo instead of a ListBox: 54 seconds for direct loading versus 6.5 seconds using a temporary string list!!

***Contributed by Fernando Madruga***

## Connecting to a Database in 12 Easy Steps


1. Create a new blank form
2. Add a **Table** component to the form. This is on the Data Access page of the component palette.
3. Set the **DatabaseName** field to the directory of where the database resides. Example: c:\delphi
4. Set the **TableName** property to the actual name of the database file. Example: names.dbf
5. Set the **Exclusive** property to TRUE if you don't want any other application to access the database while you are using it.
6. Leave **TableType** set to **ttDefault** unless you are using a non-standard extension for the database file.
7. Place a **DataSource** component on the form. This is also on the Data Access page of the component palette.
8. Set its **DataSet** property to the name of the TTable component you added earlier. If you double click on the **DataSet** property, it will select it for you.
9. The DataSource component serves as a middle-man between your database (in this example the TTable) and any data-aware controls that you place on the form.
10. Place a **DBGrid** component on the form. This is on the Data Controls page of the component palette.
11. Set the **DataSource** property of the DBGrid to the name of the DataSource component you placed earlier. If you double-click in this field, it will pick it for you.
12. Now go back to the Table component and set its Active property to TRUE. Delphi will immediately activate the connection, open the database and fill the columns in the DBGrid with the fields of the database.

[Return to Tips & Tricks](#)

[Return to Front Page](#)



## **Second Helping**

A response to "Food For Thought" on May 24 

**Original Code by Ian Martin. Adapted for clarity By Tim Benham**

It is not necessary to go through the bit twiddling shown by Siamak Shams. Sorry Siamak... The key to loading them is to give them names that are numbers and store them in a \*different\* resource file as noted by our editor. The following code will load any number of cursors provided they are named as ascending numbers starting from crFirst.

[Source Code for Loading a Cursor](#)

[Return to Tips & Tricks](#)

[Return to Front Page](#)

## The Unofficial Newsletter of Delphi Users - Issue #5 - June 26th, 1995

```
{===== Project =====}
program cursors;

uses
  Forms, Winprocs, Wintypes, Main in 'MAIN.PAS' {MainForm};

{$R *.RES}
{$R Cursor2.res}          (* Contains Cursors named '1', '2' '3' *)

type
  XStr = String[14];  (* string with nul terminator *)

const
  crFirst = 1;

Var
  Counter      : Integer;
  CursorName   : Xstr;
  ACursor      : HCursor;

function StrEx(x: Integer): XStr;
(* Creates String from integer plus a null terminator *)
var
  t: XStr;
begin
  Str(x, t);
  StrEx := t + #0;
end;

begin
  (* Create Cursors From resource file *)
  Counter := crFirst;
  Repeat
    CursorName := StrEx(Counter);
    ACursor := LoadCursor(HInstance, @CursorName[1]);
    If ACursor <> 0 Then Screen.Cursors[Counter] := ACursor;
    Inc(Counter);
  Until ACursor = 0;  (* keep going until we get a miss *)
  (* Do something .... *)
  Application.CreateForm(TMainForm, MainForm);
  Application.Run;
end.

{===== MainForm =====}

unit Main;

interface

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

type
  TMainForm = class(TForm)
    procedure FormCreate(Sender: TObject);
  end;

var
  MainForm: TMainForm;
```



**implementation**

*{ \$R \*.DFM }*

**procedure** TMainForm.FormCreate(Sender: TObject);

**begin**

    Cursor := 1; *(\* select first cursor read from resource \*)*

**end;**

**end.**

[Return to Second Helping](#)

[Return to Front Page](#)

Bitmaps can be easily loaded from a .RES file by using the Windows API call **LoadBitmap** as follows:

```
MyBitmap.Handle := LoadBitmap(HInstance, 'BitmapID');
```

Of course you need to include the RES file in with your program by including the following compiler directive:

```
{ $R MYSTUFF.RES }
```

Place a **Panel** on a form and set its **Align** property to **alTop**. Then run the application. When the form comes up, shrink the form vertically so that you pass over the bottom edge of the panel you placed. Now, with the panel partially obscured, drag the right side of the form to the right. The panel should resize, but it doesn't, apparently because it is partially obscured.

**Delphi Developer** from Pinnacle Publishing has recently made its debut. Issue #1 is 24 pages and is packed with useful tips and techniques for the professional Delphi developer.

**Issue #1:**

1. Building Data-Aware Components
2. Creating an Animation Component
3. Reviews of 4 Delphi Books
4. Tips on Using Images in Database Apps
5. Use of Templates to Speed Development

Subscriptions are normally \$149 for 12 monthly issues, but they are offering a special subscription rate of \$99 for a limited time. Code samples are available on their BBS or via CompuServe (PINNACLE). Contact them at 1800 72nd Avenue South, Suite 217, Kent, WA 98032.

***Delphi Informant*** from Informant Communications just recently shipped its 3rd issue. This monthly magazine is very professional & polished and provides a wealth of information for Delphi users.

**Issue #2:**

1. OOP for the Uninitiated
2. The Triumph of Objects
3. Migration from VB to Delphi
4. Delphi Exception Handling
5. Table and Query Components
6. 3-D Custom Label Component

**Issue #3:**

1. Tracking Database Events
2. Moving to Local Interbase
3. OnCalcFields in databases
4. Delphi Executable Sizes
5. Cursor management in the Memo Control
6. The .DFM file

Subscriptions to Delphi Informant are \$49.95 for 12 monthly issues. Code samples are available on their BBS or their new CompuServe forum (ICGFORUM). Contact them at (916) 686-6610.



