Basically Visual Magazine

ISSUE #2 FEBRUARY 1993

What's in the future?

Object Vision   DOS
???
ACCESS   Windows

THIS MONTH....


OPEN SUB()

Well, I made it. The first issue got out on-time. Only by hours, but it DID get out on time. The response has been very encouraging. People from all over have been downloading the magazine from the various boards that have elected to carry it. I even have had calls from as far away as Sweden on my own board to pick up the magazine. No one has sent in any subscriptions yet, but I haven't given up hope yet. In addition, the comments about the magazine are with out exception, very favorable. The only complaint that I have received (as of January 20) is that the archive is too big. You can read many of the comments in the E-MAIL section. All in all it looks like the magazine will be well accepted.

I did, however, make a couple of goofs. Most of the big ones can be found in the corrections section. There is one big one that I would like to address here. What I forgot in my frenzy to get the magazine out, was to explain to all that the Windows 3.1 Help "edition" is more than just a way to present the text. The Help utility that comes with Win 3.1 is fully a hyper-text viewer. Some readers already know that, but many don't. So let me spend this time giving you an overview of using this special electronic edition. Many of these things were not in the first issue, but have been added into this one.

When you see the green solid underlined text,
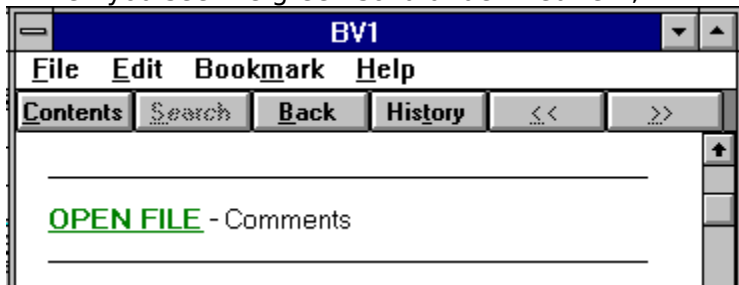
it designates that by clicking here, you will be presented with that topic or section.
When you see green broken underlined text, like this, it lets you know that there is a PopUp box waiting for you to click on it. Go ahead. Try it.
The browse buttons at the top right of the help toolbar

 will move you from one article/topic to the next either forward or backward.
The search button...
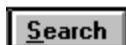
allows you to use a keyword search to find the general article you are looking for. When this button is pushed, you will be presented with a screen similar to this:

Search

Type a **word**, or select one from the list.
Then choose Show Topics.

[Cancel]

[Show Topics]

```
ADS
ADVANCED
BBS
BEGINNERS
COMMENTS
CONTENTS
```

Select a **topic**, then choose Go To.

[Go To]

As you can see, instructions are included.

If you find something that you want to include in your program, you don't have to print it out and re-type it.   Click on the "Edit" menu item, and select the Copy option.   This puts the article/topic into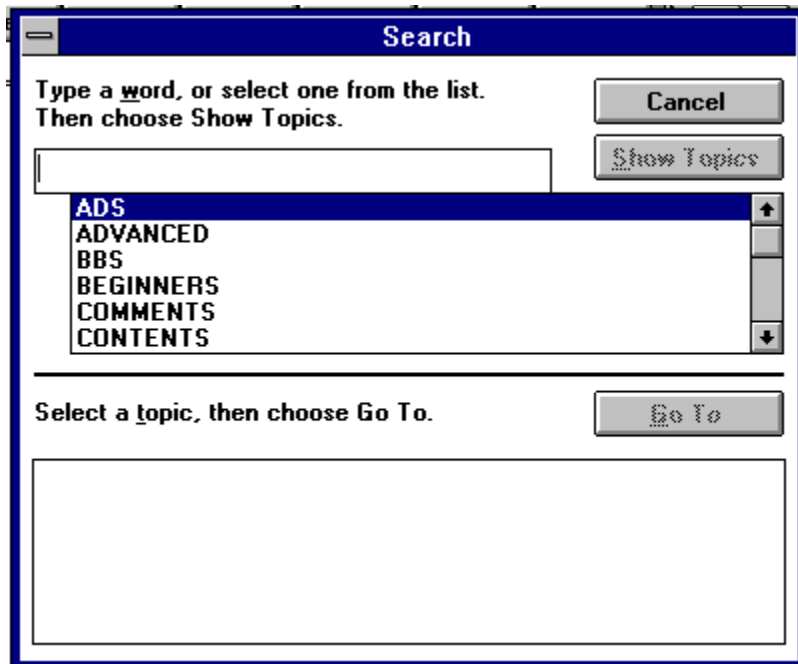 a box that you can mark the section you want and then paste to the clipboard.   From there, paste it into your application and you are home free.   *For those using VB-DOS, you can paste to notepad and then save it as a text file that can be imported later.*

I have also included a couple of special "hot-spot" graphics that you will see from time to time.   These are included where a tip or some special information is called for, but there isn't any specific short text to use as a link.   Here is one you will see this issue.   Go ahead and try it.

Finally, if you want to print out a specific article/topic, you can do so by selecting [F]ile/[P]rint Topic from the menu.

Now, you should have a pretty good idea of how to use this version of the magazine.   We are looking for a good hyper-text reader for the DOS world, since there are a number of you out there who can't or won't run Windows.   So for all you DOSer's, hang in there.   We REALLY haven't forgotten you.

END SUB()

Well, we come to the end of another issue.   I must admit that the last two months have been a REAL challange.   Trying to get a full issue out in 2 weeks was hard, and I let too many things slip on the BBS.   Hopefully, now that I have a full month to get things done, I can spend more time with the board and my family.   I can also spend some time learning the help system better so we can look even better for you.

We are still looking for other authors to help out.   We need a Visual Basic for DOS Advanced author, an Object Vision Advanced author and both levels for Microsoft Access. If you feel you can contribute in these areas, please let us know.

We also would love to get articles from you.   Code snippets, some of your tricks, what ever. You don't need to be a professional author.   Please help us out.

Coming up next month...
Microsoft ACCESS Review

Visual Basic for DOS Review

New Object Vision Beginner's Column

More from Mark, Tony and Greg

and much much more!

Thanks for reading and see you in March...

Greg

SUBSCRIPTION INFORMATION

Basically Visual Magazine is published monthly and distributed via BBS Systems.
Basically Visual Magazine is a shareware concept magazine.   If you find the magazine to be useful, the price is $20.00 US. Funds per year.

To subscribe, please send check or money order to:
BASICALLY VISUAL MAGAZINE
PO. BOX 1214
YORK, PA USA 17405-1214

The success of the magazine depends on you contributing.

E-MAIL TO THE MAGAZINE

We got a great deal of mail this last two weeks about the magazine.   Here is just some of the responces...

===

Greg,
[The magazine] Looks good!   Just got back this afternoon and took a look at it.   I applaud your use of the Windows help compiler - it makes a very flexible format.
Since you can cut and past from the help file into VB, anything you publish that contains source code can easily be brought into the programming environment without having to resort to some intermediate file format.   GREAT!
One problem that I see - SIZE.   Most networks that I know of will balk at sending around a half-meg file, especially since the information inside is mostly redundant.   How do you feel about distributing only the .HLP file?   Since everyone can print help topics (and since the HP and Epson print files are so huge) how about leaving the .PRN files out completely?
I realize that the page headers and table formatting won't look as nice, but the guy sitting out there with a C. Itoh dot matrix or HP Deskjet printer (or any of the other hundreds of Windows supported printers) won't be too happy to find that he just downloaded a massive file of which only 7% is useful.
My suggestion is to break out the .HLP file and only distribute it and adjust the readme.
Jim Harre (sent via Netmail)
{Jim,
*   While it is true that most of the people that can use the magazine are already on Windows, there are many people out there that can't or WON'T run Windows.   Many of the people who use VB-DOS don't have Windows.   That is why I decided to include the print image files.   I had to make a determination as to the type of printers we would support.   The two most supported printers in the world are the HP laser format and the Epson Dot matrix format.   Until I can find a DOS viewer good enought for our tastes, I feel that I MUST include the print image files.*
*   Any board that distributes the magazine may, at their choice, include all three versions or any of the three.   This issue is being distributed as three files so the boards can choose which ones they want to support.*
*   Greg}*

===

Hi Greg,
Just took a look at BV1 and wanted to let you know how impressed I was with the quality and professionalism evident in this premiere issue. Well done!...
I was wondering how I'd go about putting an add in the next issue for my BBS?
Also had 2 suggestions. First, have the general release only include the .hlp file. The size of BV1 prevents most users here from downloading it, I made a second archive with just
the .hlp file (hope you dont mind) and that version is being downloaded regularly. If the user needs to print a section, this can be accomplished under the Win help facility. In all likelihood, users would only want to print a particular section and not the entire magazine.
Second suggestion, have a compressed version of your file list available for freq rather than the raw listing.
Well, that's about all, but keep up the good work! :-)
Mead Himelstein (from NETMAIL)
{ Mead,
*   Thanks.   In answer to your second question, the cost of a BBS ad is $10 per issue or one article per ad.   (Maybe this will get some people to send in articles :-))*

As for your first suggestion, please see my first reply above.   As to the second, it is done.
*   Greg}*
===

What is BV Mag?   How can I get a copy?
Gregory Chang   (on the Visual Basic Message Echo - FIDONET)
{Gregory,
*   Many boards are picking up the magazine.   In fact, thanks to Mark Wisecarver, you can also find the latest issue on CIS.   If you are having problems finding the magazine, contact one of the boards shown in this issue.}*
===
Hi Greg
Do you know of any UK based BBS that carries your mag as I would be very intrested in reading a copy, I am sure many others over here would be too.   I do not feel ocnfident enough in BBSing to try to get it 'direct' via netmail.
Best wishes with your new project.
Janet.
Janet Barkaway (on the Visual Basic Message Echo - FIDONET)
{Janet,
*   I don't know of any BBS's in the UK that currently carry the magazine.   If there are any, let us know and we'll post it here.*
* }*
===
More next month!

CORRECTIONS

Last issue, I started our rating system by using an eye bitmap.   What I forgot to do was tell you what the rating system was...

The minimum rating is 1 eye and the maximum is 5 eyes.   There are many things that are taken into consideration when awarding points to a product.   In no particular order, they are...

1) ease of use

2) documentation

3) size of the program vs what it does

4) ease and options available in installation

5) price

6) gut feelings.

The rating system has been set down by the magazine, but the rating given is that of the reviewer and the magazine WILL NOT override that rating.

We try to assign reviews to people who actually use the product or similar products.   This way, you will get a more honest review that you can really use, and what you can expect ... both good and bad.

Last month, I said that we would have a review of Visual Basic for DOS and a disk labeling program in this issue.   I under-estimated the amount of time required for these articles and I just flat ran out of time.   I just couldn't finish them before we "went to press".   They WILL be included in the March issue.

LIGHTHOUSE FILE SYSTEM
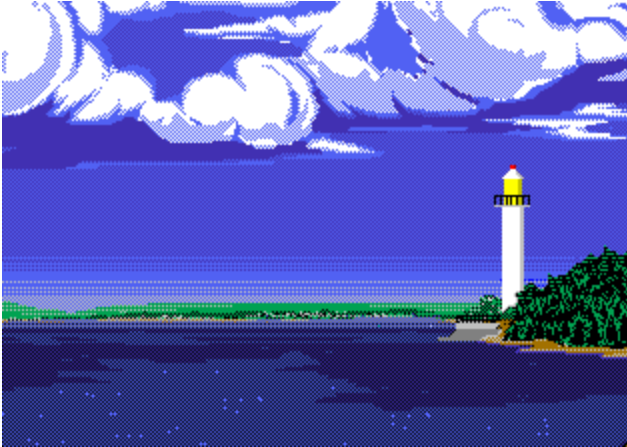
The Light House is located in Flatrock, MI and is a Programmer's support system. While it is not a BBS in the standard sense of the word, it is a very good place to get any kind of Visual Basic support files.   The Light House is a FREE system that accepts NO MONETARY CONTRIBUTIONS.   If you have FREQ abilities, give us a call.

The Light House is a PKWARE Liscensed distributor.

For more information on how to gain access to this system contact Mark Wisecarver at:

FIDONET 1:2380/410 USR HST 14.4 COURIER

CIS: 72400,505 (All Microsoft forums)

VB DOS KICKSTART

Visual Basic for DOS - Beginner's Level

BY Anthony V. SEO

Learning to crawl

The intent of this column is to help new users of Visual Basic for DOS (hereafter referred to as VB/DOS) get familiar with the IN's and out's of the package. If you are a long time Basic programmer, just making the move to the forms driven environment of VB/DOS or even a novice, starting out for the first time, this column hopefully will be of some interest to you. I am going to make the assumption that the readers at least have some familiarity with some of the Basic keywords.

In the last installment, we started reviewing some of the differences between the sequential programming methods used with the Basic language over the years and the Forms driven programming style of Visual Basic. We created a form called DATA1.FRM, that we used a simple example to illustrate some of the basic techniques of VB/DOS. This article will expand on that and introduce a couple of new items as well.

First fire up VB/DOS and load the form called DATA1. In the box titled Project on the right hand side of the screen, you should see DATA1.FRM followed by the word data1 in parenthesis. Click your mouse one time one the Data1.FRM line to highlight it, then click the Form button to switch back to the Forms Designer. Using your mouse go to to the Tools list and again select the tool called Text Box. This time the box that appears will have the word Text2 in it. First use your mouse to drag the Text2 box to where you want it on the screen. Then using the Properties menu, find the line that says Text and delete the word Text2. Pretty much the same as last time. Now notice that it is hard to tell what box is what. That calls for a new object called a Label. The label is a text box that displays what is entered into the Caption property. So Click on the Label tool, and then using the Property menu, change the caption from Label1 to Month. Then drag it along side of the first Text box. Next add a second label, change the name to Year, then move it along side of the second text box, which we just added.

Now we have the beginnings of a real data entry form. If you have been following the script, there should be two command buttons, two text boxes, and two labels. Now press F12 to save the form and get to the Event Procedures menu. First thing we want to do is to setup the same kind of data validation that we used in the Text box, where we checked for a number between 1 and 12, but change it so that we can test for a range of years to be entered. So at the Event Procedures menu, click on Text2, then select LostFocus. Then enter the following.

```
SUB Text2_LostFocus ()
    IF Text2.Text = "" THEN
        oops% = MSGBOX("Year not entered", 5, "Warning")
        SELECT CASE oops%
            CASE 4
                Text2.SETFOCUS
            CASE 2
                Command1.SETFOCUS
        END SELECT
    END IF
    If VAL(Text2.Text) < 1992 Or Val(Text2.Text) > 2002 Then
        oops% = MsgBox("Invalid Year entered", 5, "Warning")
        SELECT CASE oops%
            CASE 4
```

```
                    Text2.Text= ""
                    Text2.SETFOCUS
              CASE 2
                    Command2.SETFOCUS
          END SELECT
      END IF
      Command2.SETFOCUS
  END SUB
```

SHORTCUT: Press F2 to display a list of all of the subroutines.   Go to to Text1_LostFocus, then using the mouse or the Shift and the arrow keys, highlight everything under the subroutine name, and to just above where it says END SUB.   While it is still highlighted, press <CTRL><INSERT>.   This will copy the text.   Then jump back to the Text2_LostFocus subroutine and put the cursor under the first line, then press <SHIFT><INSERT>.   This will paste the text into this area.   Then just make the necessary changes.

Now you also have to make a change to Text1_LostFocus.   At the end of the routine, where it says Command2.SETFOCUS, between the END IF and END SUB, change that to say Text2.SETFOCUS.   This means that after we enter a month, that pressing <TAB> will take us to the Text2 box.

So now we have a form that should allow us to enter a month (1 - 12) and a year (1992 - 2002).   Now for some fancy stuff.   Sometimes it can be hard to follow the cursor around on a form, especially a crowded one.   So, we are going to use the GOTFOCUS event procedure and a new command word called BACKCOLOR.   All of the properties available to the Tools (or objects), such as color, event size and position, can be manipulated within the program itself.   BACKCOLOR simply refers to the item's background color.   To get started, press F12, then select the Text1 object, then click on the GotFocus event.   A GotFocus event occurs any time an object is selected, or is now the current cursor position.   Then type the following:

```
  SUB Text1_GotFocus ()
      Text1.BACKCOLOR = 14
      Label1.BACKCOLOR = 14
  END SUB
```

Now what will happen is that when you click on the Add button, the background color of the Text1 box and the Month label, become bright Yellow.   Do the same for Text2_GotoFocus.   Then at the top of the Text1_Lostfocus procedure (where we did the month validation), insert the following lines:

```
  Text1.BACKCOLOR = 7
  Label1.BACKCOLOR = 7
```

Then do the same for Text2_LostFocus, changing the Text1 and Label1 to Text2 and Label2.

What happens here is that when the Text1 box is selected, it is yellow, when you press tab to move to Text2, Text1 turns back to gray, and Text2 turns yellow.   You can use the same principles with the Command buttons as well.

This ability to change the property of an object is a very important tool in VB/DOS programming.   For those of you with the Professional Edition of VB/DOS, the SETUP.MAK, which is a nice software installation tool, (Microsoft style, of course).   Take a close look at how the properties of the various boxes, color size, shape, etc., are manipulated.   One

working example of this would be to have the name of the month that we selected in Text1, to appear in another box, which does not become visible until we enter a valid month.   For starters, go back to the DATA1 form and add another Text box (should be Text3).   Delete the Text3 line from the Text property.   NOTE: You can easily do this at the start of the program, by setting Text1.Text="", etc.   However for the sake of speed, and program size, I prefer to set as many of the properties as I can in the Form Designer.   Then from the Property menu select the Visible property, and change it to False (for Text3).

     Press F12, then go back to the Text1_LostFocus event.   What we will do here is build a separate subroutine to assign the name to the month.   Then we call that subroutine and make Text3 visible with the correct month.   First build the subroutine.   Type SUB month_name (m%, mname$) and hit enter.   A new subroutine will be created (still part of the Data1.FRM), then enter the following:

```
SUB month_name (m%, mname$)
    SELECT CASE m%
        CASE 1
            mname$ = "January"
        CASE 2
            mname$ = "February"
        CASE 3
            mname$ = "March"
        CASE 4
            mname$ = "April"
        CASE 5
            mname$ = "May"
        CASE 6
            mname$ = "June"
        CASE 7
            mname$ = "July"
        CASE 8
            mname$ = "August"
        CASE 9
            mname$ = "September"
        CASE 10
            mname$ = "October"
        CASE 11
            mname$ = "November"
        CASE 12
            mname$ = "December"
    END SELECT
END SUB
```

     Press F2 and go back to the Text1_LostFocus subroutine.   Right before the line that says Text2.SETFOCUS (which should be at the end of your routine).   Type in the following:

```
m% = val(Text1.Text)
Call month_name (m%, mname$)

Text3.Text = mname$
Text3.VISIBLE = -1
```

     Now run the program.   When you type in month number into Text1 and <TAB> or click on Text2, the Text3 box with the month name should appear.

     That's all for this issue.   Next time we will start working with the concept of a Project,

and the basics of working with multiple forms.   Remember that if you have any questions or suggestions (except for an early expiration), please send them care of this magazine.   Till the next time.

THE PROGRAMMER'S PIT BBS

Where can you ALWAYS find the latest issue of **Basically Visual Magazine**?   Where can you find ALL the back issues of **Basically Visual Magazine**?

The Programmer's Pit BBS - Home of **Basically Visual Magazine**, that's where.

==
==

(717) 845-2725   (717) 845-2725   (717) 845-2725   (717) 845-2725

==
==

We have 700+ MEG ONLINE and over 2 GIG off-line of applications, program source, games and information files.   We are a member of FIDO and SPEAKEASY NETWORKS.   We carry program source for:

> **Visual Basic**
> **Pascal**
> **C / C++**
> **Modula-2**
> **Assembly**
> **AND MORE!**

23 hours a day!   7 days a week!

If you have FREQ abilities, the magic filenames are:

FILES	ALL ON-LINE FILES LIST
NEWFILES	ON-LINE FILES LESS THAN 15 DAYS OLD
MAGAZINE	THE LATEST ISSUE OF BASICALLY VISUAL MAGAZINE
BV-HLP	Windows 3.1 Help version
BV-HP	HP Laser Print image version
BV-EPS	EPSON Dot Matrix Print image version
CD-ROM	FILE LISTS OF CD-ROMS

ALL ON-LINE FILES CAN BE FREQ'ed.
SEND FILE REQUESTS FOR OFF-LINE FILES.
FIDONET ADDRESS - 1:270/612
SPEAKEASY ADDRESS - 18:18/10

VISUAL BASIC DEPOT
Advanced Visual Basic for Windows programming
BY Mark Wisecarver

Development made easy (well, easier)

Welcome back gang. In Issue One of Visually Basic Magazine I mentioned that this column would be going into some of the new features in Visual Basic 2.0, (a revolutionary product), in this the second press release of the Visual Basic Depot. So throw another log on the fire and kick up your slippers while I go into some of those new features for you right now.

One of the best enhancements is a hidden change in version 2.0, and that being the smaller and faster applications you create with the new VB. Code is loaded on demand rather than all at once which was the case with the earlier version, allowing your applications to load and run much faster. There is also a new form property, ClipControls, which eliminates the creation of control masks, speeding up the form painting event in your application. I'd go into more on the properties of the Form.ClipControls here but Microsoft has included a very good help base with version 2.0 which explains the ClipControls in much detail. (which allows us to cover more items in this issue)
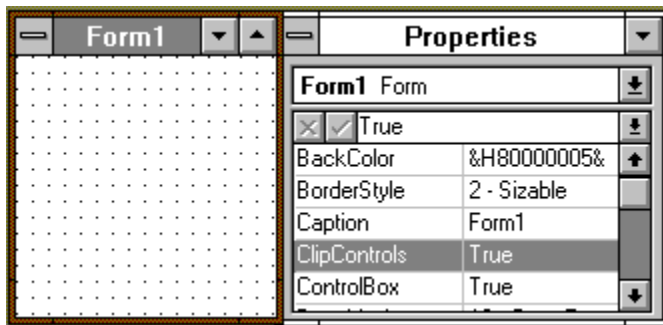
Arrays are much improved also. You remember the horrors of the limitations for Huge arrays, well now they are gone, limited only by available memory. Arrays can now be declared in Type declarations for data structures also. Plus there is the addition of the Preserve option in the Redim statement which let's you change the size of a dynamic array without losing its contents. You can now therefore read from a file into a string array, re-dimensioning the string array as you go, without having to know ahead of time how large to make the array. Application capacity has been greatly increased. Huge arrays blow the limits off application size, and string space is limited only by available memory. Also, strings can be compared to a pattern using another new function called Like. If the given string matches the pattern string, then this function returns the Boolean value True, otherwise it returns False.

Probably the first thing you will notice about VB 2.0 is the new Properties Window. It is scrollable, sizeable, and you can hide it out of site if you don't need it. All at your leisure. Debugging is enhanced with Watch options to keep tabs on the contents of variables and properties, and the value of expressions. You can set multiple breakpoints, and single step through your program.

Microsoft also added another new feature, AUTOLOAD.MAK, which can be customized to load only the files that you want to start each new project with. Another new keyword, Private,

makes subprograms and functions visible only to routines in the same source code module.

Multiple instances of an application are now allowed. The App.PrevInstance property is used to determine if a previous instance of an application is already running. Another new keyword, Me, refers to the current form when a form name is passed to a function or subprogram. If you are creating MDI applications, (Multiple Document Interface), there is a new method available to you called ZOrder, which sends controls or forms to the front or back in the overlapping order that you assign to them.

Visual Basic 2.0 also shares a new concept in variables. called variants, with Microsoft's new Windows database Access. In previous versions of BASIC, variables defaulted to single precision floating point values if not explicitly declared otherwise. Now, variables default to the new variant type. A variant adjusts automatically to whatever is stored in it.

It is now easy to select and move a group of controls on a form by clicking and dragging a rubber band type rectangle over the face of the form.
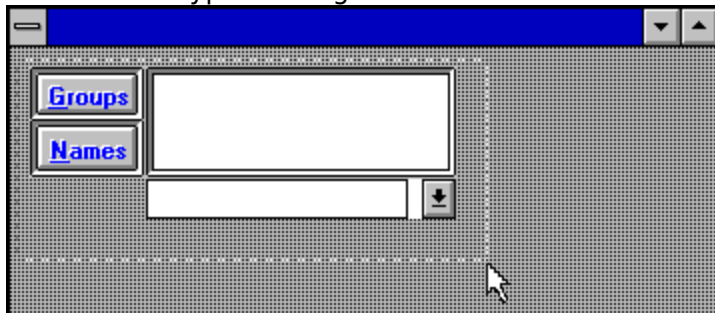
Among all of the new controls that come with both the Standard and Professional versions my favorite among them is the CMDialog control which creates five types of commonly used dialog boxes in a style consistent with other Windows applications. With the Action property you set the type of dialog to be created, and other properties allow you to customize any of the dialog boxes as you desire.
The following table lists the Action settings:
0   No action
1   File Open Dialog Box
2   File Save Dialog Box
3   Color Dialog Box
4   ChooseFont Dialog Box
5   Printer Dialog Box
6   Invoke WINHELP.EXE

The Professional edition also comes with a very good Help compiler that uses the Rich Text Format (RTF). All control and form objects have a new property called HelpContextID. The setting of this property matches with a section in your help file, letting the Windows Help engine pop up context sensitive help for whichever control has focus in your application.

All of this started in 1991 when Microsoft introduced Visual Basic. The Visual Basic programming system packaged up the complexity of Windows in a truly amazing way. Combining the proven capabilities of Basic language with the new visual design tools. The initial release of Visual Basic was a runaway success, selling tens of thousands of copies and winning awards from most of the major computer magazines. Now with the release of Visual Basic version 2 we can add these important new strengths and features to the list:
- Improved performance
- New properties, events, methods, and keywords

- New debugging tools
- Greater flexibility for declarations
- Multiple selection of objects in forms
- Color-coded program elements
- Support for 256-color displays and improved graphics support
- Support for advanced Windows features such as OLE (object linking and embedding)
   and   MDI (multiple document interface)
- The addition of ODBC (Microsoft open database connectivity programming interface.

There are a lot of new features in both the Standard and Professional editions of Visual Basic version 2.0, and I could keep writing on and on about them but it's time to get into some programming for this months readers. One thing is obvious, Microsoft listened to it's customers and came out with one of the best upgrades we have ever seen.

And now let's get into this months questions from our readers.

BG: Mark, I think there must be a way to drag controls on a form during run-time but for some
 reason I can't figure it out. Can you help? *signed desperate*
MW: Dear "desperate", truthfully I'm glad you asked this question. Many   VB programmers may
have heard or have felt there was a way to do exactly what you have asked, and it's easier than you
think. Load up Visual Basic cause you are going to love this example.
First let's have some fun. Start a new form and place a Command button on it. Now the only change I want you to make is from the Properties Window. Change the DragDrop property of Command1 to Automatic and then start the form. Now use your mouse to drag the Command button. Something very interesting is happening right now. The Command button can be moved during runtime but jumps back to its original location. This is because you have not moved the Command button, you have only dragged its outline around on the form. To actually move it, add a Move method to the form's DragDrop procedure.
Sub Form_DragDrop (Source As Control, X As Single, Y As Single)
Source.Move X, Y
End Sub
Now the DragDrop event occurs whenever a control is dragged over a form (or other control) and dropped. The *Source* argument contains the name of the control (the Command button) that was dragged over and dropped on the form. The **X** and **Y** arguments contain the location of the mouse pointer when the control was dropped. The *Move* method is used here to actually move the control to the new coordinates. Now run the application and notice again what happens to the control that is being dragged. The control being dragged does not respond like an object that was dragged and dropped. This is because the upper-left corner of the control is moved to the **X**, **Y** location supplied by the DragDrop procedure. Therefore the **X**, **Y** location is not the upper-left corner of the outline being dragged around on the form but the location of the mouse pointer on that outline.
Interesting isn't it? I'm sure by now you are aware that we need to chance the coordinates that the DragDrop procedure is using if we are going to do this correctly. We are going to have to capture the **X**, **Y** coordinates of the *MouseDown* event on the control and then subtract that location from the **X**, **Y** coordinates in the *DragDrop* procedure.
'Place this code in the form's General procedure and define the variables we need for the offsets.
Dim XOffset As Single   'This is for later use
Dim YOffset As Single   'Again, for later use
Const StartDrag = 1
Const Drop = 2
'Now place this code in the controls MouseDown procedure

Sub MyControl_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
'Now we need to store the location of the MouseDown event and initiate the Drag method.
Sub Form_DragDrop (Source As Control, X As Single, Y As Single)
Source.Drag Drop 'Remember we defined a constant for this earlier.
Source.Move X - XOffset, Y - YOffset   'Here we are calling the variables we defined.
End Sub
This procedure ends the manual dragging and actually moves the control for you. The procedure then offsets the move by the location of the initial MouseDown event of the control. Remember to change the DragMode of your control back to **0 - Manual**. (The default setting)
Now run your application again and DragDrop with your control. Allot better isn't it? This example should be of use to many of the programmers reading this magazine, and this type of user control over a well developed application gives it a very professional appearance. Remember, most people buy a car on looks and feel, not driveability. The method I have shown you will work with any control that has a MouseDown event, which includes list boxes, labels, file list boxes and picture boxes. Now remember, you can insert a custom icon in your controls DragIcon property. The custom icon is always centered on the mouse pointer so you can manually insert the size of the offset.
Well, that's it for my article this month. Next month I promise to do allot less verbiage and allot more coding. Get those entries in for the Visual Basic for Windows contest, the deadline for all entries is March 15th and the winner will be announced in the April issue of this magazine. Until next month, Happy programming.

Mark Wisecarver

Publishers of Engineering & Scientific Software is looking for software titles to add to our catalog.   We are looking for quality software that meets specific needs in the professional community.   If you written a package or are thinking about writing a package, feel free to contact us.   Even if it might have market potential of only 200 or less copies, let us know.   We offer help with software design and documentation, and have a solid royalty program.   Write to us at

Master\sofT
P.O. BOX 579
Camp Hill, PA 17001

Or call us at (717)-763-5772 Voice, (717)-763-4419 BBS or Fidonet 1:270/711

BASIC BEGINNINGS

Visual Basic for Windows - Beginner's Level

by Greg Walters

STEPPING BACK

Last month we did a very simple program, but we didn't talk much about the processes behind it.   So, let's step back a bit and talk about programming in general.   When we finish this month, we will create a traffic light simulation.   I know that this sounds silly, but the principles behind the program are sound.

The first thing you should do when writing a program, is to try and understand as much about the task that you want the program to do as possible.   If you can do it on paper, you should be able to write the code for the program.   With this in mind, let's look a our simulation.
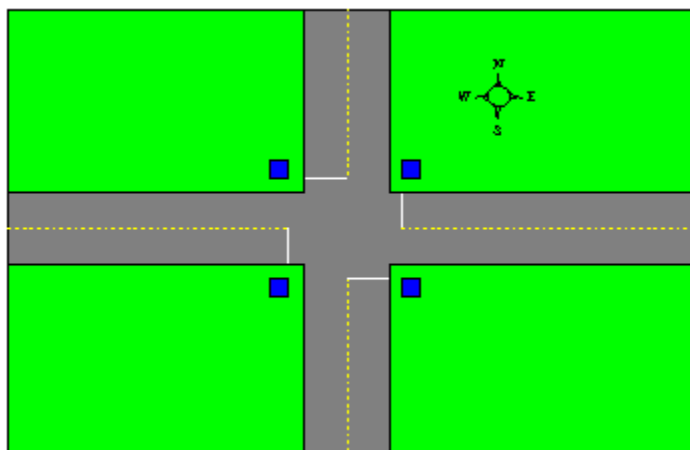
Here is a simple street layout.   Traffic flows North/South and East/West.   The blue squares are traffic signals.   There are no turn signals for this simulation.   **[Editors note: For those in England and other countries that drive on the left, look at this in a mirror. :-)]**
Now let's look at the timing of the lights...
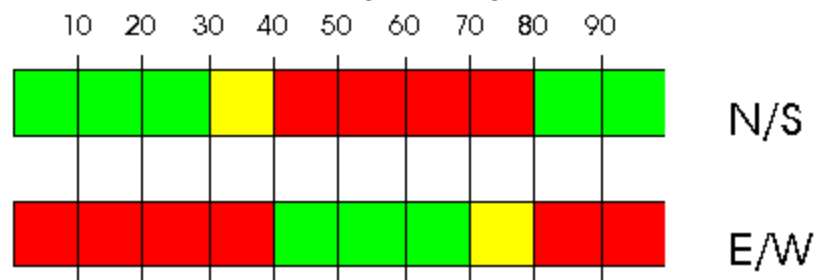
As you can see, the E/W side light is red thru both the green and yellow lights for the N/S side.   Only when the yellow cycle is over, do the lights change from red to green on the other side.   Then the whole thing starts over again.   We can use this fact in our program. We will need two timing cycles.   One for green and one for yellow.   Since the red light time is a sum of the two of these, we don't really need a red cycle.   (Remember...this is a VERY simple simulation.   There are many traffic systems where the light cycles are different.)

 We also need a flag of some sort that marks when the N/S side is active (Green&Yellow) and one that marks when the E/W side is active.   Thankfully, this is easy.   We can us a <u>BINARY</u> switch.   If the switch is TRUE, the N/S side is active.   If the switch is FALSE, the E/W side is

active.

Now this sounds great, but what does it mean for the program?   Well, it means that we need a Binary variable for our N/S flag and we need two time intervals, one for green and one for yellow.

Since Visual Basic doesn't support a Boolean (Binary) variable directly, we use an INTEGER type of variable.   An Integer is a whole number between -32,768 and 32,767.   Integers can NEVER have fractional parts.   An integer type variable is useful for timer intervals, counters and Binary variables.

So, now we have three things our program will need...two timer intervals and a Binary value.   Now that that is decided, we need to decide what our program will actually do.

We want our program to show two traffic lights...one for the N/S side and one for the E/W side, showing the proper lights, switching from green to yellow to red and back to green. We should have a way to start and stop the simulation.   We also should have a way for the user to find out a little bit about the program, an about box is standard in Windows applications.   There also should be an exit button, so the user can quit the simulation.

Here is a VERY simple flow chart of how the main program loop should run...
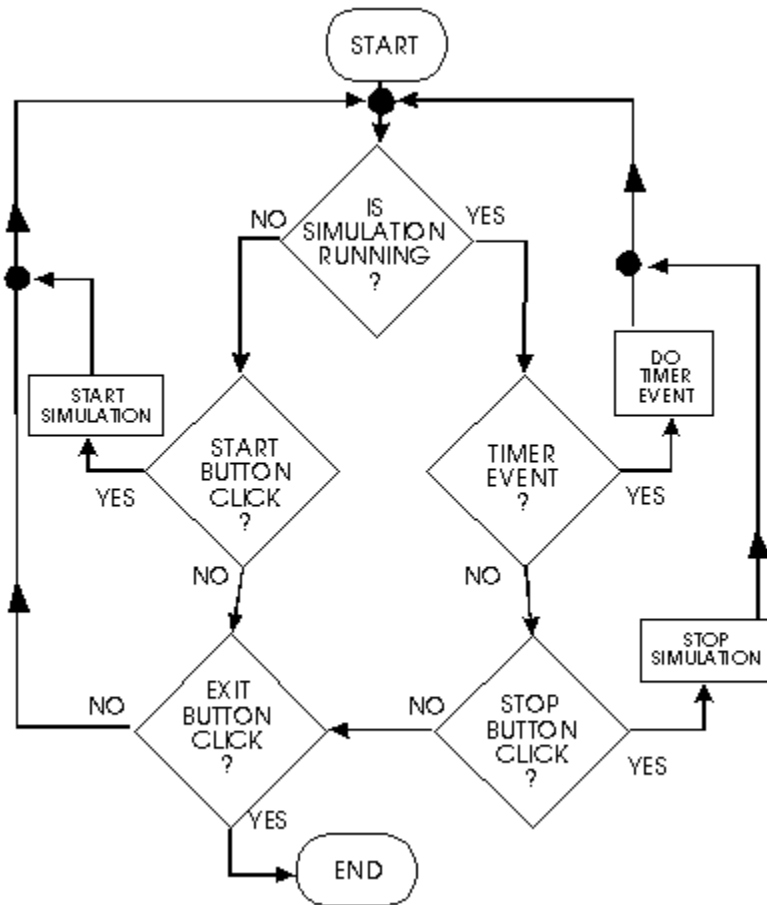
Now for the real work of our program.   How should the timer event work?   In fact, what is a timer event?   We touched on that last month, but let's go into it a bit further.
When we include a timer into our program, we can set the amount of time that goes by until the system flys a red flag that say's "THE TIMER HAS GONE OFF!!!   DO SOMETHING!"   It's like an alarm.   We set the amount of time between these "flags" by setting the interval property in the timer control.   This time is expressed in milliseconds. (1000 milliseconds = 1 second)   When the timer event occurs, the system looks in the timer subroutine to see what to do.   Here is the flow of the timer event...
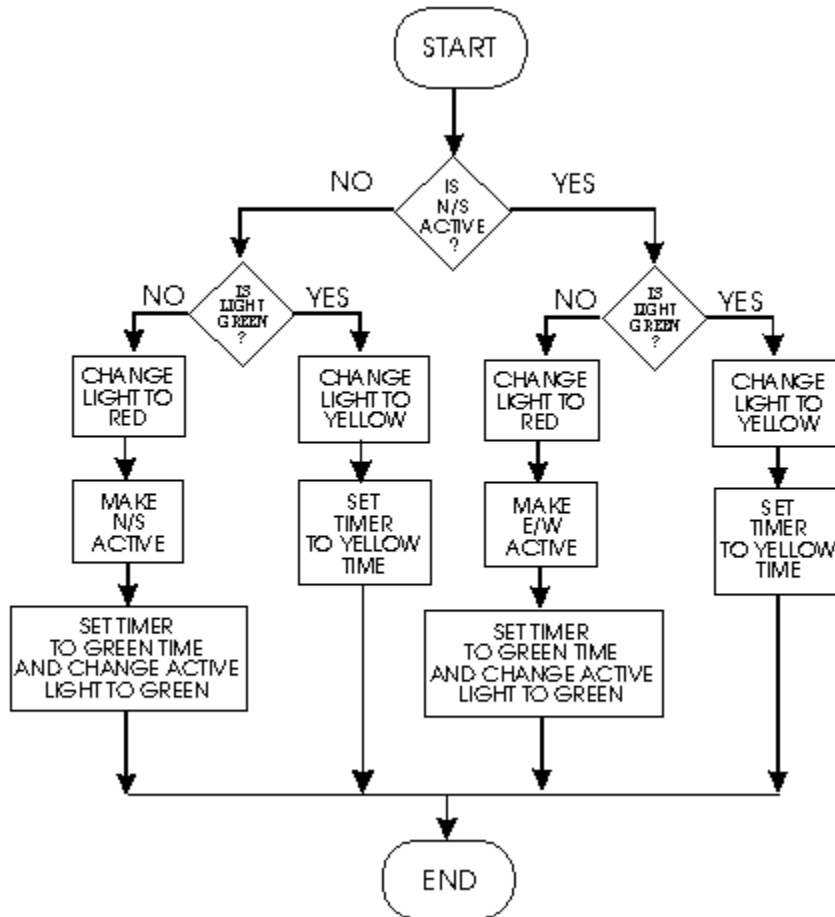
Notice that the two sides of the flow chart are exactly the same with the exception of the line that makes one side or the other the active side.

Where do we go from here?   Well, we have to design our forms and then write the actual code that glues the controls and forms together into a single application.   For this program, we will use two forms.   One for the main simulation and one for the about box.   (No, I didn't forget the about box click on the first flowchart...Just ran out of room for the graphic.   It can be put before the "Exit Button Click?" box.)   Let's do our main form.
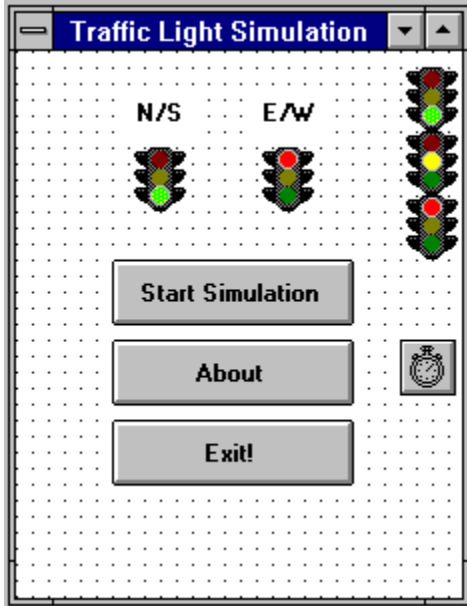
You will need to place two labels (shown as "N/S" and "E/W") and make the CAPTION property N/S and E/W. Next place 5 image controls. Place them as shown. Change the NAME and PICTURE properties as follows:

NAME	PICTURE	WHICH IMAGE
=======	=========	=================
N_S	Trffc10a.ico	Far Left Image
E_W	Trffc10c.ico	Next Image to the right
Green	Trffc10a.ico	Top right Image
Yellow	Trffc10b.ico	Center right Image
Red	Trffc10c.ico	Bottom right Image

(Now, I know that there is a great scream from those who are super-duper programmers out there, that there is a "better" way to do this...but remember, we are teaching beginners. We'll get into that later)

Next place 3 control buttons on the form and change the CAPTION and NAME properties as follows (top to bottom):

NAME	CAPTION
======	=============
Toggle	Start Simulation
About	About
EXIT	Exit!

Place a timer control on the form. Change the VISIBLE property of the three image controls on the right to FALSE. Finally, change the CAPTION property of the form to TRAFFIC LIGHT SIMULATION and change the NAME property to TRAFFIC.

Now let's create our About box. This is a simple form with two labels and one control button. Set it up like the picture below and change the NAME property to ABOUTBOX. Make the form background Blue and the label foreground white. The label BackStyle should be made Transparent. Then type the text shown into the Label controls.

About Traffic Light Simulation

Traffic Light Simulation program was written by G.D. Walters for Basically Visual Magazine Issue 2 as part of the Beginning Basics column.

This program is released into the PUBLIC DOMAIN by G.D. Walters and Basically Visual Magazine.

Return
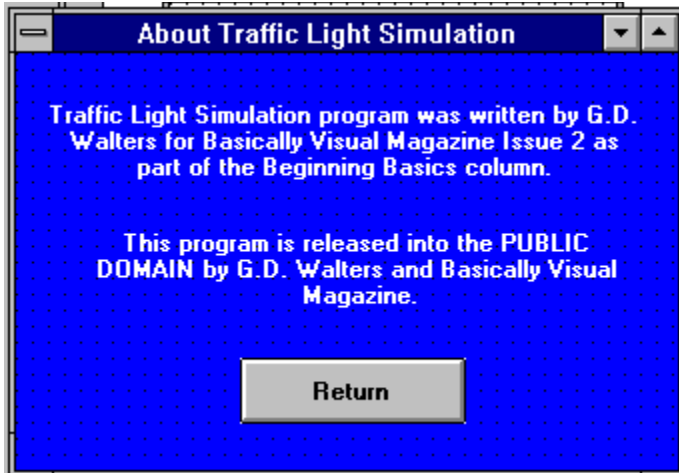
Now we are almost ready to write the code to glue all of this together.   Notice that, in the first flow chart, there is a reference to a Start Button Click and a Stop Button Click.   In our form, however, we don't have enough buttons...right?   WRONG!   We will use the same button for both.   That's why we named the first button "Toggle".   Let's look at the logic behind this.   We will need another BINARY flag.   One called ToggleFlag.   When we start the program, we will set this flag to FALSE.   When the user clicks on button #1, we will change this to TRUE.   When the user clicks on the button again, we will reset it to FALSE.   We will do some other things in the same code, but we'll look at that in a bit.   Now we start coding. On the PROJECT window, highlight the TRAFFIC form and click on the "View Code" button. This brings up the declarations section of our code window.   Here is where we will declare all the variables used in the program.   We need to make all the variables available to our main form so we have to declare them as SHARED.   We use the DIM keyword to reserve memory space for them.   Type the following...

```
DIM SHARED ToggleFlag As Integer
DIM SHARED NS_Go As Integer
DIM SHARED LightState As Integer
CONST GreenTime = 5000
CONST YellowTime = 2000
```

Remember eariler I said that we need a binary flag that will show which direction is active? That is what NS_Go is.   We have already talked about ToggleFlag, but what about this LightState variable?   This is used to give us a quick way to tell if the light is green or not. A "1" here says that the light is green.   A "2" says it is yellow.

The CONST keyword defines a CONSTANT value to a name.   Once declared, you can't change it from inside the program.   I've set the length of a green light to 5 seconds and the length of a yellow light to 2 seconds.   Not real world, but you would have to wait too long to see anything happen.   Feel free to change these numbers as you wish. REMEMBER 1 second is 1000 milliseconds.

Now let's set up the FORM.LOAD procedure.   When the program starts, this routine will be run.   We will use this to set the defaults for our program.   Click on the "Object" pulldown and select Form.   Make sure you are in the FORM_LOAD procedure and type...

```
Timer1.Enabled = FALSE
Toggle.Caption = "Start Simulation"
ToggleFlag = FALSE
```

The first line turns off the timer.   The second line makes sure that the caption of our first button is set to "Start Simulation".   The last line sets ToggleFlag to FALSE.   Now go to Toggle object and in the Toggle_Click procedure type...

```
IF ToggleFlag = FALSE Then
    ToggleFlag = TRUE
```

```
        Toggle.Caption = "Stop Simulation"
        Timer1.Interval = GreenTime
        NS_Go = TRUE
        LightState = 1
        N_S.Picture = Green.Picture
        E_W.Picture = Red.Picture
        Timer1.Enabled = TRUE
    ELSE
        ToggleFlag = FALSE
        Timer1.Enabled = FALSE
        Toggle.Caption = "Start Simulation"
    END IF
```

From top to bottom...

We check to see if the ToggleFlag variable is set to FALSE.   If so, we set it to TRUE.   We then change the CAPTION to "Stop Simulation".   This gives us the two buttons in one.   We set the timer interval to the GreenTime constant we defined earlier, set the NS_Go flag to true, set the LightState to 1 (green).   We then make the image controls show the pictures we want.   The N_S image to Green and the E_W to Red.   We do this by assigning the PICTURE property to that of the PICTURE property of the hidden control we want to show. Of course we could load the picture off of disk, but this takes time and we would have to include the pictures when we distribute the files.   This way is much quicker and easier on us.   Next we turn on the timer to start the simulation (Timer1.Enabled = TRUE).   If the ToggleFlag was set to TRUE, then we ignore all of that and set it to FALSE, turn off the timer and reset the caption to "Start Simulation".

Now let's do the code for the other two buttons.   For the About_Click procedure...

```
        AboutBox.Show
```

causes the AboutBox form to be loaded and shown.   For the Exit_Click procedure ...

```
        Unload ME
        END
```

will unload the main form and end the program.

Now for the real meat of the program.   The Timer1.Timer function.   Type...

```
    IF NS_Go = TRUE THEN
        IF LightState = 1 THEN
            LightState = 2
            N_S.Picture = Yellow.Picture
            Timer1.Interval = YellowTime
        ELSE
            LightState = 2
            N_S.Picture = Red.Picture
            E_W.Picture = GreenPicture
            Timer1.Interval = GreenTime
            NS_Go = FALSE
        END IF
    ELSE
        IF LightState = 1 THEN
            LightState = 2
            E_W.Picture = Yellow.Picture
            Timer1.Interval = YellowTime
        ELSE
            LightState = 2
            E_W.Picture = Red.Picture
            N_S.Picture = GreenPicture
            Timer1.Interval = GreenTime
```

```
            NS_Go = TRUE
        END IF
    END IF
```

Between the flow chart and the previous code, you should be able to follow this code by now.   Finally, change to the ABOUTBOX form and under the Command1_Click procedure, type...

```
    Unload ME
```

Well, now you are on your own.   If you want to do more with this project, you might try two things...

        1) Add a Walk/Don't Walk picture to each side and change them at the proper times.

        2) Allow the user to change the times for Green and Yellow.   (You will have to remove them from CONST status, and assign them as variables.)

Below is quick access to the full source code.   You can copy the code to the notepad, save it as a text file and pull it into VB that way.   The code is written to be under a subdirectory named "TRAFFIC".   You will need to change this for the pictures if you don't use that directory name.

Until next month...Have fun...

Click here for the Main form Code

Click here for the AboutBox form code

Click here for the Makefile code

ABOUT THE AUTHORS

G.D. Walters

Greg Walters has been working with computers and programming since 1972.   He has worked as a freelance programmer and a MIS Director.   He has written articles for Modules & Definitions Magazine and has worked as a freelance technical editor for Osborne/McGraw-Hill on a number of books.   He is currently a programmer for St. Onge Company in York, PA. He is also the editor of Basically Visual Magazine and runs The Programmer's Pit BBS.

Mark Wisecarver

Mark is a senior programmer for a major corporation and lives in Flat Rock, Michigan .   He has been a programmer since 1979.   He has background in Basic, GWBasic , Quick Basic, Microsoft C, Assembly Language, JCL, Turbo Pascal and Visual Basic programming.   He runs an online service free to all specializing in Visual Basic code.

Anthony V. Seo

Tony has been working in the personal computer area for 14+ years.   He has a background in Basic and   Xenix/Unix programming.   Tony runs Marketing & Automation Resource Company, a systems consulting company and software publishing company in Camp Hill, PA.   He also runs the M.A.R.C. Information System BBS System.

L. John Ribar CCP

John has been a programmer since 1978, and is the author of several books, including ***C DiskTutor*** and ***FORTRAN Programming for Windows*** from Osborne/McGraw-Hill.   He is the president of Picasso Software Group Ltd., a software development firm in York, PA, specializing in the creation of Windows-based applications and tools. John has written articles for Computer Language, the C Users Journal, Modules & Definitions, and Micro Cornucopia magazines.

TRADEMARKS

Microsoft, MS, MS-DOS, Visual Basic, Access are registered trademarks, and Windows is a trademark of Microsoft Corporation.

Borland and Object Vision are trademarks or registered trademarks of Borland International, Inc.

Basically Visual Magazine is a trademark of Gregory D. Walters.

Other brand and product names are trademarks or registered trademarks of their respective holders.

TIP: Use your mouse to highlight the ClipControls setting in the properties window and then press F1 to call up Visual Basic's on-line help for this property.

Here is an example of a PopUp.   Extra information will be place here in the form of text and sometimes graphics.

TIP: When you see this button, press on it for more information about the subject. 🙂

Starting Out with ObjectVision

SOFTWARE REVIEW

by L. John Ribar

Basically Visual Magazine rating---

It's been a long time since the first version of ObjectVision hit the market.   It was billed as the next great frontier in programming, using "objects" and "pointing" to develop programs. Programmers not required!

Well, the first few revisions didn't help OV's reputation as not needing programmers.   In fact, most programmers I know didn't even want to play with it.   With the newest release of Object Vision Pro (version 2.1), there may be more reasons to take a look.   And now, finally, the market has caught up with the vision and promise that ObjectVision offered.

When you are getting started with OV, Borland has assembled a vast array of examples, including a checkbook manager, address book, time tracking and billing, and more.   Just look at the menu of demos, shown here:

Demo Application Menu (Complete)

Real Estate Buyer Qualification	Contact Management	Expense Report	Hospital Admissions Form	Motel Reservation System	Time Sys
Nutrition Expert System	Ernst & Young Survey	Checkbook Management System	Vehicle Maintenance System	Vacation Planner	Re Pla
Address Book	Game Show	Hangman Game	Home Inventory	Cost of Living Compare Two Cities	J Estir Sys
Order +db History	Wine Selection	Inventory Management for Parts Truck	Computer Diagnosis System	Auto Buyer Financing Guide	E

The "Pro" version of OV includes Crystal Reports, a graphical report writing program, and Turbo C++ for Windows.   Using these tools, a complete package can be written with minimal coding.   OV is used to draw and "animate" the user screens, menus, etc.   Crystal Reports allows you to quickly draw your reports.   Turbo C++ allows you to write your own DLLs (dynamic link libraries), for any functions that OV forgot to include.   If you want to work with OV, I would suggest the "Pro" version of the package.

The screens that can be created are quite exciting in the Borland demos.   Here is an address book application, with pages that look like an address book!

Address Book (Goal)

| Name |
| Philippe Kahn |

| Title/Position |
| President/CEO |

| Company |
| Borland International, Inc. |

| Address |
| 1800 Green Hills Road |

| Address2 |
| P.O. Box 660001 |

| City | State | Zip Code |
| Scotts Valley | CA | 95067-0001 |

| Country | Relationship |
| | ☒ Business ☐ Personal |

| Work Telephone | Ext | Home Telephone |
| (408) 438-5300 | | |

| Fax Telephone | Birthday | Anniversary |

| Notes |
| Borland makes the hottest Windows development products, including ObjectVision and Turbo Pascal for Windows. Stay in touch to get all info on product updates! |

Buttons: Top, Previous, Next, Bottom, Lookup..., Find..., Birthdays, Store, Clear, Delete, Help, Exit

And the following is an application for managing your recipes.   While these types of demos can also be created in other environments, I was impressed that Borland included useful, tasteful, and well designed applications to show off the capabilities available in OV Pro.

Recipe Planner (Complete)

New England Clam Chowder — **Recipe Planner**

Meal(s) ☐ Breakfast ☒ Lunch ☒ Dinner   Food Group Seafood

Course(s) ☐ Appetizer ☒ Side Dish ☒ Main Dish ☐ Dessert

Serves 4   Quantity to Serve 4   Preparation Time (hours) 0.75

Line	Amount	New Amt	Measure	Ingredient
1	2.00	2.00	Tbs	Clarified Butter
2	3.00	3.00		Hearts of celery, Chopped
3	3.00	3.00		Leeks, white part only, chopped
4	1.00	1.00		Spanish onion, chopped
5	1.00	1.00		Idaho potato, peeled and diced
6	3.00	3.00	tsp	White flour
7	20.00	20.00		Top neck clams, shelled, chopped, strained, and juice reserved
8	1.00	1.00	cup	Heavy cream
9	1.50	1.50	cup	Water

[About...] [Delete Line]

Service Temperature ☐ Cold   ☐ Room Temp   ☒ Hot

Source of Recipe   Bill Blass Dining in Manhattan

[Back] [Next] [Clear] [Delete] [Store] [Print] [Lookup] [Instructions...]

The Review

There are a lot of things I like about OV itself.   The designer is a little simpler than Visual Basic.   There aren't many add-ons though.   There are direct connections BUILT-IN for database management.   These include support for Paradox, dBase, ASCII, and other file types.

If the database doesn't exist, you can create it inside OV.   Any of the file types are supported.   Once the database is designed (in or out of OV), you simply point to the database field that should be used for each of your screen fields.   The link is than complete.

But OV is not finished!   Once the database links are specified, OV will automatically add Next, Previous, Top, Bottom, Update, Clear, Store, and other buttons to your window, if you so desire (you get to pick which ones you want, if any).

In about 10 minutes, I built a simple document tracking program, shown below.   While it just

tracks the data, with no searching or reporting yet, we'll add those functions starting next issue!

The big concept to get used to in OV is Event Trees.   In Visual Basic, you can assign functions to button clicks, etc.   In OV, you build an Event Tree, that is a graphical representation of what happens when a button is clicked, for instance.   This concept is rather straightforward once you have dealt with other similar environments, but might take a few minutes to understand if this is your first try.   As an example, here is the event tree for the button marked "WRITE" in the document tracking program shown above:

Event tree for Write
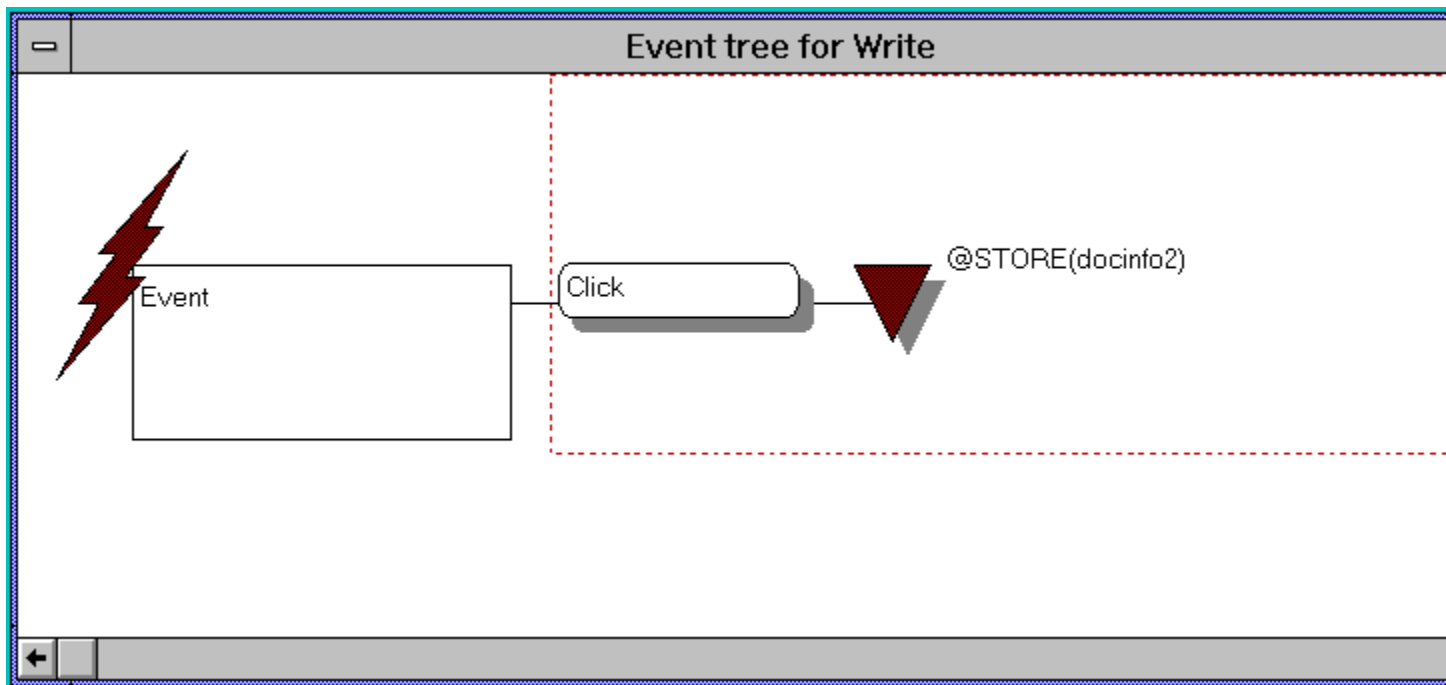
This "tree" shows that an EVENT occurred, in this case a CLICK, which calls the function STORE() with the name of the database.   This is about as difficult as it gets.

ObjectVision is coming into it's own, and with the addition of Crystal Report, to write your reporting functions, and Turbo C++ to add in your own functionality (via DLLs), this should become a serious contender in the Visual Programming Environment wars.

As a side note, there are other many other features available from the OV functions (used in the trees), including DDE links, file manipulation, and graphic objects (the fire-breather in my window above was included with IV, and added with a simple click!).

Probably the only thing I really don't like about OV is the way end-user applications are distributed.   Unlike Visual Basic (and others), there is no "EXE" version of an OV file.   You must distribute the run-time version of OV.   Depending on the type of license you buy, there may be additional costs required for the copies that you sell.   Ask your Borland rep, as this may change by the time you read this.

Coming Up

It's hard to review these types of products, because of the change in thinking patterns that they represent.   In the next few issues, I'll walk you through the development of the document tracking program I mentioned previously.   In this effort, we'll add outside functionality (through DLLs), talk to other programs with DDE, and print reports with the Crystal Reports program, all of which are included in the OV Pro package.   This should give you a better idea about how well OV will fit into your needs as a programmer, non-programmer, or manager.

[EDITOR'S NOTE: John will be joining the staff of Basically Visual Magazine and will be doing the Object Vision Beginner's column - Object Vision 101.   Look for it soon!]

TIPS, TRICKS AND GOTCHA'S

I don't know quite what to call this first one, so we'll just jump into it.
Many times when I write programs, I have to access rather large databases.   When this happens, the user ends up waiting sometimes 2-3 minutes or more.   I like to show that something is happening, rather than just showing a box that says "Searching the database...Please wait."   Users get impatient and think that the machine has hung up, when in reality, it's just trying to get all the data in.   So I've come up with a short little re-usable form that I use called Wait4It.Frm.   While digging through the WingDing font that comes with Window 3.1, I found the clock faces, and got the idea that this would make a good way to show something is happening.
The form is self-contained and calls to it are easy.   I've included a simple form to start the Wait4It form and stop it either by the user pressing a control button or waiting for a 10 second timer event.   Here is what the Wait4It form looks like...
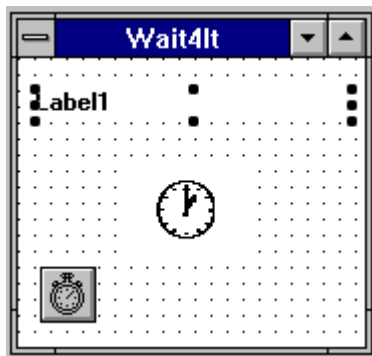
There are two Label controls...One named Prompt and the other named Clock.   The Clock label box has a FontSize of 32 and a FontName of "WingDings".   There is also a timer that is set for a 500 ms. interval (1/2 second).   The Prompt label box allows you to display what ever you want to the user.   Here is the code for the Wait4It form...

Start with declaring a shared variable called showchar that is an Integer type

```
Dim Shared showchar As Integer
```

Next under the Form_Load routine, set the default value of showchar tochr$(183), then put that value into the caption of the labelbox named Clock.

```
Sub Form_Load ()
    showchar = 183
    Clock.Caption = Chr$(showchar)
End Sub
```

Whenever a timer event occurs, add one to the showchar value and check to see if we have overrun the clock faces.   If so, reset it to the first clockface.   Then show it by displaying that as the caption for the label box called Clock.

```
Sub Timer1_Timer ()
    showchar = showchar + 1
    If showchar > 194 Then
        showchar = 183
    End If
```

```
      Clock.Caption = Chr$(showchar)
 End Sub
```
Now to use it.   Here is the code for the demo program...
```
 VERSION 2.00
 Begin Form Form1
    Caption          =    "DEMO"
    Height           =    4170
    Left             =    1740
    LinkTopic        =    "Form1"
    ScaleHeight      =    3765
    ScaleWidth       =    2715
    Top              =    1620
    Width            =    2835
    Begin CommandButton Command2
       Caption          =    "Exit Demo"
       Height           =    495
       Left             =    720
       TabIndex         =    2
       Top              =    2505
       Width            =    1215
    End
    Begin CommandButton Command1
       Caption          =    "Start Demo"
       Height           =    435
       Left             =    645
       TabIndex         =    0
       Top              =    1710
       Width            =    1395
    End
    Begin Timer Timer1
       Interval         =    10000
       Left             =    1110
       Top              =    1050
    End
    Begin Label Label2
       Alignment        =    2   'Center
       Caption          =    "From Basically Visual Magazine Issue #2"
       Height           =    480
       Left             =    225
       TabIndex         =    3
       Top              =    3135
       Width            =    2250
    End
    Begin Label Label1
       Alignment        =    2   'Center
       Caption          =    "This demo shows the use of the Wait4It module."
       Height           =    765
       Left             =    345
       TabIndex         =    1
       Top              =    150
       Width            =    1950
    End
 End
```

First I setup a shared variable called toggle.   This is used for the "Start Demo" button.

```
Dim Shared toggle As Integer
```

Here's what happens when the user clicks the button...

```
Sub Command1_Click ()
    If toggle = False Then
        toggle = True
        Command1.Caption = "Stop Demo"
        Wait4It.Prompt.Caption = "Searching Data..."
        Wait4It.Show
        Wait4It.Timer1.Enabled = True
        Timer1.Enabled = True
    Else
        toggle = False
        Command1.Caption = "Start Demo"
        Wait4It.Hide
        Wait4It.Timer1.Enabled = False
    End If
End Sub
```

As you can see, the Wait4It form is brought forward and the timer on that form is enabled, starting it.   I also start a timer on this form just to end it in 10 seconds.   You can do this from any code, not just a button click event.   To stop the Wait4It form, the timer.Enabled property is set to FALSE, stoping the timer from doing anything else... and the form is hidden.

```
Sub Command2_Click ()
    Unload Wait4It
    Unload Me
    End
End Sub

Sub Form_Load ()
    toggle = False
End Sub

Sub Timer1_Timer ()
    toggle = False
    Command1.Caption = "Start Demo"
    Wait4It.Hide
    Wait4It.Timer1.Enabled = False    'Stop the timer
    Timer1.Enabled = False
End Sub
```

The rest of the code is pretty self-explanatory.   Hope this helps someone out there.

HOW TO GET A SINGLE LINE TEXT BOX TO RESPOND TO THE ENTER KEY

It's not unusual to have a number of text boxes for data entry on a form.   One of the biggest problems for new Windows users, is getting used to pressing the TAB key rather than the ENTER key to move to the next field.
To get a text box to respond to the ENTER key is a simple matter. In the KeyPress method simply add the following code...

```
Sub Text1_KeyPress (KeyAscii As Integer)
```

```
     If KeyAscii = 13 Then
         KeyAscii = 0    'reset the entered key stroke so we don't
                         'get a beep back.
         Text2.SetFocus  ' move to the next field
     End If
 End Sub
```

If you don't want to make a call directly setting the next field to go to, you can fake Windows into thinking the user pressed the tab key by using this routine instead...

```
 Sub Text1_KeyPress (KeyAscii As Integer)

     If KeyAscii = 13 Then

         KeyAscii = 0

         SendKeys "{TAB}"    'Send a tab to go to the next control.

     End If
 End Sub
```

MORGAN SUPPLY

BBS WATCH

This month, I thought that we'd look at some of the shareware and freeware code that's available on various boards around the country.   This is just a small sample.   Contact your local Visual Basic Support board for more information.

```
23PICK.ZIP        14476 11-16-92   Source code for Computer/Player game
256PB2.ZIP        28595 11-16-92   How to load 256 colors in VB picture box
3D4VB.ZIP         16741 11-16-92   3D effect subroutine library (FramesBoxesBorders)
3DDEM.ZIP         19763 11-16-92   Demo of 3D for VB
ABCSLOTS.ZIP       9872 11-16-92   Code for Slot machine game
ACCRD1.ZIP        30520 11-16-92   Code for Accordian Solitare
ADIALR.ZIP        23787 11-16-92   Code for Phone dialing program
AFFIRM.ZIP        39729 11-16-92   Code for positive thinking program
AHELP.ZIP          5468 11-16-92   Code for Windows help (Add Help)
ALARM-D.ZIP        4441 11-16-92   VB/DOS Alarm clock example
ALTFNT-D.ZIP      11455 11-16-92   VB/DOS Alternate font graphics example
ANIMT-D.ZIP        4398 11-16-92   VB/DOS Graphics animation example
APIHELP.ZIP       64818 11-16-92   Help with Win API
APIREF.ZIP         6962 11-16-92   Xref of Win API calls to their defining DLL
APIX.ZIP            256 11-16-92   Gives instant access to Windows API declares
API_FU.ZIP        57270 11-16-92   Windows API call descriptions in cardfile format
APMMET.ZIP        24633 11-16-92   MetaFile library support (ViewPrint)
APPKT11A.ZIP      45386 11-16-92   Replacement for Program manager (Shell)
APPSHL.ZIP        33827 11-16-92   Application Shell (Includes mini Text editor)
AREACO.ZIP         9328 11-16-92   Area code program in VB
ARRANG.ZIP         6117 11-16-92   Code to arrange icons at bottom of screen
More [Y,n,t,=]? =ARRAY-D.ZIP       4278 11-16-92   VB/DOS Arrays example
ARRAY.ZIP          8641 11-16-92   Code for using API fast I/O routines
ASCII-D.ZIP        4008 11-16-92   VB/DOS ASCII graphics example
ASSOC.ZIP         12643 11-16-92   List associations between File/App
AUTOV10.ZIP       56889 11-16-92   Code for Auto maintenance
BAR-D.ZIP          3921 11-16-92   VB/DOS Pro Bar chart example
BARS3.ZIP         45857 11-16-92   BarCode printing code
BASECVRT.ZIP      13268 11-16-92   Converts numbers between bases
BC_CDK.ZIP        10224 11-16-92   Doc using BC++/IDE w/MS VBASIC Control Dev.kit
BDEMO.ZIP         24219 11-16-92   Code to create buttons containing pics
BINARY-D.ZIP       4439 11-16-92   VB/DOS Binary tree example
BITS_DLL.ZIP      13561 11-16-92   DLL supplies bit operations
BLNKBLNK.ZIP      45725 11-28-92   Screen Blanker/Out to lunch program w/Src (VB WIN)
BMPKIT.ZIP        25215 11-16-92   Routines for .BMP files within Visual Basic
BOOKS_VB.ZIP       4472 11-16-92   Text file listing 22 VisualBasic books
BTDEMO.ZIP        47413 11-16-92   Demo of 3D custom tools
BTRTES.ZIP        50984 11-16-92   Code to access Btrieve files
BTVB.ZIP           6524 11-16-92   Code for BTRIEVE files in Visual Basic
BURGLR.ZIP        15586 11-16-92   Examples of how to animate ICO's
BUTTON.ZIP        13169 11-16-92   Make differant buttons
BV0442.ZIP        36837 11-16-92   Huge array support for MicroSoft Visual Basic
BV0447.ZIP        45215 11-16-92   WINAPI.TXT: Win 3.0 API Declarations for VB
CALC-D.ZIP         4137 11-16-92   VB/DOS Windowed calculator example
CALLBT.ZIP        27260 11-19-92   Btrieve example code for VBasic
CARD10.ZIP        38842 11-16-92   DLL for Card game programming in VB
CBLIST.ZIP        32576 11-16-92   Custom Control to create groups in single control
CCFACT.ZIP        76250 11-16-92   Interactive development tools
```

```
CFIXVB.ZIP        7043 11-16-92   DLL to fix decimal places in currency values
CHK4EXE.DIZ       3911 12-12-92   How to check for instance of same program being loaded
twice
CHOYCE.ZIP       17977 11-16-92   Associate several programs with one extension
CLBOX1.ZIP        3567 11-16-92   Incremental list box
CLOCK-D.ZIP       3888 11-16-92   VB/DOS Vector graphics example
CLOCK.ZIP        23543 11-16-92   Another VB clock
CLP25610.ZIP     17910 11-16-92   BitMap viewer
CLPSIB.ZIP       15380 11-16-92   Set WS_CLIPSIBLINGS style bit for all forms
CLPTL.ZIP       194398 11-16-92   Save/Load/Print/Display from ClipBoard
CM01.ZIP         63579 11-16-92   Phone/Address manager in VB
COLCLIP.ZIP       9380 11-16-92   Select RGB Fore/Background pasting with Clipboard
COMDEM.ZIP       53951 11-16-92   Terminal program with code
COMPS.ZIP       120820 11-16-92   Store and retrieve real estate info
COPYDESK.ZIP      3997 11-16-92   Example of how to copy DeskTop to Visual Basic
CRC32-VB.ZIP      8742 11-16-92   DLL w/DOC's to check file CRC
CRISPY.ZIP       22855 11-16-92   Solitaire variation (Need VBCARDS.DLL from CARDS10.ZIP)
CURLIBM.ZIP      28440 11-16-92   Libraries of cursor animations. (Create you own)
CURSMAN.ZIP      11512 11-16-92   How to control the cursor in VB
CUSCON.ZIP       65500 11-16-92   3 custom controls (GaugeListState)
DB-D.ZIP          6978 11-16-92   VB/DOS DataBase example
DB_HDR.ZIP       13166 11-16-92   HowTo read dBase file structure for QEVB
DD1A.ZIP          6031 11-16-92   Drag/Drop how to for VB
DDEUTI.ZIP        7097 11-16-92   VB tool to test DDE code
DLAYDR.ZIP       11556 11-16-92   Code to time Mouse Drag/Drop event
DLL4VB.ZIP        6158 11-16-92   How to write simple DLL and use it in VB
DLLMGR.ZIP       15385 11-16-92   Load/Unload DLL,VBX,FON,EXE,DRV files
DODOS.ZIP        30344 11-16-92   Utility to provide runtime arguments for DOS
DOSBUT.ZIP        4282 11-16-92   Code for drop to DOS from Windows
DOSHEL-D.ZIP      4529 11-16-92   VB/DOS System shell example
DRAGFO.ZIP        5834 11-16-92   Code to drag a Form or TextBox
DROPDO.ZIP        6669 11-16-92   Code for Combo drop down box
DRWSCR.ZIP        7718 11-16-92   Text on how to direct printed text
DSCAN.ZIP         6822 11-16-92   VB Code to scan dir's for File.Name
DSL100.ZIP       13670 11-16-92   VB Code for Shell to DOS
DXFDLL.ZIP       56322 11-16-92   DLL to create AutoVAD DXF format files
EDITDEMO.ZIP     22887 11-16-92   Code for several methods of In/Output
ENDPRN.ZIP        5264 11-16-92   Code to control PrintManager
ENUMFONT.ZIP     22495 11-16-92   Call Windows API Font functions
ETDEMO.ZIP        3906 11-16-92   Code for EditTool, which creates custom controls
EZHELP.ZIP       14243 11-16-92   Visual Basic adjunct to add outline help without MS Editor
FBR12C.ZIP       50626 11-16-92   Code for Application launcher
FILBX2.ZIP       11152 11-16-92   Code for FileOpen box like VB's environment
FILEBO.ZIP        5337 11-16-92   Example of old style Dir/File list
FILEBOX.ZIP      11810 11-16-92   FileBox example for VB
FILEBOX2.ZIP      5876 11-16-92   MAK Windows style File_Open box
FILER1.ZIP       19060 11-16-92   Code for Find/View/Delete of files
FINDAP.ZIP       12045 11-16-92   Multiple instance presentation with code
FNDWND.ZIP        6448 11-16-92   Code to find a form
FNTS43.ZIP      256863 11-16-92   Font viewing and cataloging
FNTVIEW.ZIP      18300 11-16-92   Code to control/View Fonts
FOC001.ZIP       16554 11-16-92   How to control focus
FOC002.ZIP       15109 11-16-92   Version 2 of Focus code with new features
FONTRO.ZIP        6410 11-16-92   Print text sideways from VB
FOPEN.ZIP        11549 11-16-92   Advanced FileOpen code
```

```
FRAME.ZIP        12441 11-16-92   Draw 3D frames around controls
FVIEW3.ZIP      111538 11-16-92   Font viewer
FXLAUN.ZIP        8670 11-16-92   Code for Metz software menu system
GRAF20.ZIP       53394 11-16-92   Display .BMP and .PCX files 2/16/256 color
GRDRTN.ZIP       11474 11-16-92   Code for grid manipulation
GRID.ZIP         25071 11-16-92   Code to test the properties of controls
GRID2.ZIP        24472 11-16-92   Enhancement to MS's GRID.VBX with better control
GRPH11.ZIP        9833 11-16-92   Custom control, displays clr.bmp
HEADERS1.ZIP    119738 11-16-92   WIN v3.1 Header files for calling Win API's
HLPKEY.ZIP        6229 11-16-92   Sets task-specific F1 hook
HOWTO.ZIP       101312 11-16-92   Very good "how to's" for VB
HPENV.ZIP        24568 11-16-92   Code for advanced Envelope printer
HPESC.ZIP         5348 11-16-92   Text on sending printer esc seq to HP
HUGE.ZIP         24039 11-16-92   VB & DLL for creating Arrays <64
HUGE2.ZIP        24302 11-16-92   Mod's to MS's Huge Array DLL
HUGEARAY.ZIP     19377 11-16-92   DLL with functions for creation and more
HUGEARR.ZIP      23184 11-16-92   HUGEARR.DLL functions for array manipulation <> 64k
HUGSTR.ZIP       43374 11-16-92   Mod to MS's Huge Array DLL to support large strings
HYPER-D.ZIP       4657 11-16-92   VB/DOS Hyper text example
ICBROW.ZIP        4479 11-16-92   Code to control USER.EXE DrawIcon
ICOBMP.ZIP       10591 11-16-92   Convert ICO to BMP in VB
ICONDLL2.ZIP     11766 11-16-92   Program to build Icon DLL of your own
ICONMAK.ZIP       4261 11-16-92   Demonstration of Icon drawing in VB
ICONVIEW.ZIP      7406 11-16-92   Code for Icon viewing and save to ClipBrd
ICONWRKS.ZIP    101297 11-16-92   MAK files to edit and save ICON's
ICOTIMER.ZIP      3514 11-16-92   How to animate the ICON in a minumized form.
ICOXTR.ZIP        4235 11-16-92   Use API to control Icons
IDVB.ZIP         11309 11-16-92   Determine ID's of controls during Dev process
INPOUT.ZIP        7427 11-16-92   InpOut.dll: INP and OUT Replacement
INPUT.ZIP        20480 11-16-92   Custom text box for VB
ISAM-D.ZIP        4584 11-16-92   VB/DOS IndexedSequentialAccessMethod Dbase example
IVB9112.ZIP       7624 11-16-92   VB examples
IVB9201.ZIP       5691 11-16-92   VB examples
IVB9202.ZIP       6912 11-16-92   VB examples
IVB9206.ZIP      13889 11-16-92   Even more good examples
JEOPKE.ZIP       14382 11-16-92   Code for Jeopardy TV show scoring program
JNAL10.ZIP      164097 11-16-92   Information manager
KEYTEST.ZIP       5372 11-16-92   Code for HexaDecimal keypress display
KWI.ZIP          25502 11-16-92   Printer routines
LABTAB.ZIP        8654 11-16-92   Code to use label controls to display data
LANDAU.ZIP       22273 11-16-92   Code for Drag/Drop Cut/Copy/Paste
LBSRCH.ZIP        8964 11-16-92   Code to search list box for match
LB_FUN.ZIP        8011 11-16-92   Code to demonstrate Send/Message API commands
LFORM-D.ZIP       4547 11-16-92   VB/DOS Load form example
LINE-D.ZIP        3924 11-16-92   VB/DOS Line chart example
LISTDR.ZIP        8596 11-16-92   Code to drag text line from location/location
LOOKHERE.ZIP     11453 11-16-92   Font control/Print code
LTBDEM.ZIP       11378 11-16-92   Create linked TextBox/ListBox code
LZSSLI.ZIP       21472 11-16-92   Code for compression of files
METASTUF.ZIP     28672 11-16-92   MetaFile tidbits
METER.ZIP        24251 11-16-92   Shows disk space left
METER1.ZIP        5551 11-16-92   Gauge scroll code
METRIX.ZIP       62353 11-16-92   Custom control analog meter with code
MHCOMM.ZIP      105030 11-16-92   Code for Comm transfers up to 19,200bps with protocols
MHELP.ZIP         7669 11-16-92   DLL with PEEK/POKE/INP/OUT and more
```

```
MODAL.ZIP         3881 11-16-92   Example of Windows Modal
MOREAPI.ZIP      13092 11-16-92   Code for additional API functions
MOUREL-D.ZIP      4295 11-16-92   VB/DOS Graphics mouse and keyboard handling example
MOVEFORM.DIZ       908 12-12-92   How to locate a form to the Lower RH screen corner
MOVTXT.ZIP        8709 11-16-92   How to move Pic control / Drag Text control
MSGBOX.ZIP       24957 11-16-92   VB MsgBox Places corresponding code on Clipboard
MSGFIX.ZIP        2983 11-20-92   Advanced MsgBox example
MSJ0992.ZIP      68555 11-16-92   MicroSoft Visual Basic journal September 1992
MSJV6-1.ZIP     236122 11-16-92   MSJ source code vol 6 no 1
MSJV6-4.ZIP     303782 11-16-92   MSJ source code vol 6 no 4
MULPIK.ZIP        8582 11-16-92   Multiple selection Listbox for VB
MULTIPIC.ZIP     10736 11-16-92   Code/.BMPs/ICOs for 3-D buttons
MULTIPIK.ZIP     11993 11-16-92   Multiple selection List box for Visual Basic
MULTSEL2.ZIP     10114 11-16-92   VBX/Code add on for ListBox control (Multi)
MYMEM1.ZIP       40861 11-16-92   Code for memory game
NETPRN.ZIP       12604 11-16-92   Code for linking LAN printers
NOCRUN.ZIP       15196 11-16-92   Code for numbers game with timers
NST14S.ZIP       60648 11-16-92   Code to change startup RLE file
NUMBER.ZIP        5441 11-16-92   Additional math routines for VB
NUMFUN.ZIP        4204 11-16-92   Fun with numbers (Text code)
NUMGAME.ZIP      17416 11-16-92   Code for childrens number game
ORDER.ZIP         7688 11-16-92   Code for professional looking Order form
PAD-D.ZIP         5130 11-16-92   VB/DOS NotePad example
PAINT-D.ZIP       5053 11-16-92   VB/DOS Graphics paint example
PANEL-D.ZIP       4290 11-16-92   VB/DOS Popup control panel example
PDOXDE.ZIP       17783 11-16-92   Two routines for working with PDOX Engine
PDOXENG.ZIP      17919 11-16-92   Example of howto use the Paradox DBF engine w/VB
PHONE-D.ZIP       4356 01-08-93   VB/DOS Phone book example
PIE-D.ZIP         3924 01-08-93   VB/DOS Pie chart example
POLYGO.ZIP        7795 11-16-92   Code for Polygon drawing in sizes/colors
PRCLP2.ZIP        6122 11-16-92   Print contents of ClipBoard
PRNTCB.ZIP       13089 11-16-92   Print clipboard (MicroSoft source)
PROADD.ZIP       91904 11-16-92   Efficient way to use MS's Pro Toolkit
SCROLL2.ZIP      19580 11-16-92   Scrolling Demonstration
VBCTRL10.ZIP    137524 11-16-92   Set of custom controls (Includes VBX's and Code)
VBDB.ZIP         81540 11-16-92   Data Base engine that supports dBase files in VB
VBDB110.ZIP      82401 11-16-92   VB dB engine w/src
VBDEVERS.ZIP     11419 11-16-92   List of developers and publishers
VBDIA.ZIP         9564 11-16-92   Arrow control(DLL and Code included)
VBDOS.ZIP        12474 11-16-92   DLL to call DOS functions with source code
AGIGO2.ZIP        6276 01-17-93   VBWIN-How to search records in VB 1.0 / 2.0
MATCH.ZIP         6162 01-17-93   VBWIN - How to do extensive pattern matching in VB
MHCTL.ZIP         3432 01-17-93   VBWIN-How to get a menu's ControlHandle
NOMO3.ZIP         6953 01-17-93   VBDOS-Resident int 104 BIOS extension for VBDOS v1.0
OVLRUN.ZIP        5388 01-17-93   VBDOS-Bug Fix, Use of "RUN ProgramName$" w/Overlays
VB-OOP.ZIP       10206 01-17-93   VBWIN-OOP for VB (Object VB)
```

```
VERSION 2.00
Begin Form Traffic
   Caption         =    "Traffic Light Simulation"
   Height          =    4500
   Icon            =    TRAFFIC.FRX:0000
   Left            =    2175
   LinkTopic       =    "Form1"
   ScaleHeight     =    4095
   ScaleWidth      =    3345
   Top             =    1395
   Width           =    3465
   Begin Timer Timer1
      Left            =    2880
      Top             =    2160
   End
   Begin CommandButton Exit
      Caption         =    "Exit!"
      Height          =    495
      Left            =    720
      TabIndex        =    4
      Top             =    2760
      Width           =    1815
   End
   Begin CommandButton About
      Caption         =    "About"
      Height          =    495
      Left            =    720
      TabIndex        =    3
      Top             =    2160
      Width           =    1815
   End
   Begin CommandButton Toggle
      Caption         =    "Start Simulation"
      Height          =    495
      Left            =    720
      TabIndex        =    2
      Top             =    1560
      Width           =    1815
   End
   Begin Image Red
      Height          =    480
      Left            =    2880
      Picture         =    TRAFFIC.FRX:0302
      Top             =    1080
      Visible         =    0    'False
      Width           =    480
   End
   Begin Image Yellow
      Height          =    480
      Left            =    2880
      Picture         =    TRAFFIC.FRX:0604
      Top             =    600
      Visible         =    0    'False
```

```
         Width           =    480
      End
      Begin Image Green
         Height          =    480
         Left            =    2880
         Picture         =    TRAFFIC.FRX:0906
         Top             =    120
         Visible         =    0    'False
         Width           =    480
      End
      Begin Label Label2
         Alignment       =    2  'Center
         Caption         =    "E/W"
         Height          =    255
         Left            =    1800
         TabIndex        =    1
         Top             =    360
         Width           =    495
      End
      Begin Label Label1
         Alignment       =    2  'Center
         Caption         =    "N/S"
         Height          =    255
         Left            =    840
         TabIndex        =    0
         Top             =    360
         Width           =    495
      End
      Begin Image E_W
         Height          =    480
         Left            =    1800
         Picture         =    TRAFFIC.FRX:0C08
         Top             =    720
         Width           =    480
      End
      Begin Image N_S
         Height          =    480
         Left            =    840
         Picture         =    TRAFFIC.FRX:0F0A
         Top             =    720
         Width           =    480
      End
   End
End
'Here is where we set our variables.
'The Dim Shared statement allows these variables to be called
'from anywhere in this form module

Dim Shared ToggleFlag As Integer     'Boolean True/False (Is the simulation running?)
Dim Shared NS_Go As Integer          'Boolean True/False (Is the North/South light the active
one?)
Dim Shared LightState As Integer     'Green = 1 Yellow = 2

'Here we set two constant values.  These numbers can not be changed
'by the program.  Constants are nice because you can simply refer
'to them by name.  This makes much more sense when you are trying
```

```vb
'to read the code (if you give them meaningful names).

Const GreenTime = 5000              'Time a green light is on
Const YellowTime = 2000             'Time a yellow light is on

Sub About_Click ()
'This event is called whenever the user clicks on the second
'button.

    AboutBox.Show       'Show the about box
End Sub

Sub Exit_Click ()
'This event is called whenever the user clicks on the third
'button --- End the program!

    Unload Me           'Unload the form and release the memory
                        'The "Me" in the above statement refers to
                        'the current form.  This is new in VB 2.0
    End                 'End the program
End Sub

Sub Form_Load ()
'Here we set our program defaults.  This routine is run whenever
'we load the form.  Since this is the first form in our program,
'everything is setup here.

    Timer1.Enabled = False              'Turn off the timer
    Toggle.Caption = "Start Simulation" 'Set the caption for button1
    ToggleFlag = False                  'Set ToggleFlag to the default FALSE
End Sub

Sub Timer1_Timer ()
'This is the timer event.  This event is called when the
'number of milliseconds set in the TimerX.Interval property
'has been reached.

    If NS_Go = True Then                     'If the North/South light is
                                             'the active light
        If LightState = 1 Then          'Green Light
            LightState = 2              'Set up for Yellow
            N_S.Picture = Yellow.Picture   'change the light to yellow
            Timer1.Interval = YellowTime   'set the timer to YellowTime
        Else                            'LightState MUST = 2 (Yellow)
            LightState = 1              'Reset LightState to 1 (Green)
            N_S.Picture = Red.Picture      'Turn the N_S light to Red
            E_W.Picture = Green.Picture    'and the E_W light to green
            Timer1.Interval = GreenTime    'Reset the timer to GreenTime
            NS_Go = False                  'Set East/West as the active light
        End If
    Else
        If LightState = 1 Then          'Green Light
            LightState = 2              'Set up for yellow
            E_W.Picture = Yellow.Picture   'Change the light to Yellow
            Timer1.Interval = YellowTime   'Set the timer to YellowTime
```

```
        Else                              'LightState MUST = 2 (Yellow)
            LightState = 1                'Reset LightState to 1 (Green)
            E_W.Picture = Red.Picture     'Turn the E_W light to Red
            N_S.Picture = Green.Picture   'and the N_S light to Green
            Timer1.Interval = GreenTime   'Reset the timer again to GreenTime
            NS_Go = True                  'Set North/South as the Active light
        End If
    End If
End Sub


Sub Toggle_Click ()
'This event is called whenever the user clicks on the
'"Start Simulation/Stop Simulation" button.
    If ToggleFlag = False Then           'If simulation is stopped...
        ToggleFlag = True                'Set the ToggleFlag to TRUE
        Toggle.Caption = "Stop Simulation" 'Set the caption on the button
        Timer1.Interval = GreenTime      'Set the timer interval to GreenTime
        NS_Go = True                     'Make the North/South light the active one
        LightState = 1                   'Set LightState to 1 (Green)
        N_S.Picture = Green.Picture      'Make sure that we are showing a Green Light
        E_W.Picture = Red.Picture        'and Red Light in the proper places
        Timer1.Enabled = True            'Start the simulation timer
    Else
        ToggleFlag = FLASE               'Set the ToggleFlag to False
        Timer1.Enabled = False           'Turn off the timer
        Toggle.Caption = "Start Simulation" 'Change the button caption back
    End If
End Sub
```

```
VERSION 2.00
Begin Form Aboutbox
    BackColor       =   &H00FF0000&
    Caption         =   "About Traffic Light Simulation"
    Height          =   3525
    Left            =   1020
    LinkTopic       =   "Form2"
    ScaleHeight     =   3120
    ScaleWidth      =   4965
    Top             =   1770
    Width           =   5085
    Begin CommandButton Command1
        Caption         =   "Return"
        Height          =   495
        Left            =   1680
        TabIndex        =   2
        Top             =   2280
        Width           =   1695
    End
    Begin Label Label2
        Alignment       =   2  'Center
        BackColor       =   &H00FF8080&
        BackStyle       =   0  'Transparent
        Caption         =   "This program is released into the PUBLIC DOMAIN by G.D. Walters and
Basically Visual Magazine."
        ForeColor       =   &H00FFFFFF&
        Height          =   735
        Left            =   600
        TabIndex        =   1
        Top             =   1320
        Width           =   3975
    End
    Begin Label Label1
        Alignment       =   2  'Center
        BackColor       =   &H00FFC0C0&
        BackStyle       =   0  'Transparent
        Caption         =   "Traffic Light Simulation program was written by G.D. Walters for
Basically Visual Magazine Issue 2 as part of the Beginning Basics column."
        ForeColor       =   &H00FFFFFF&
        Height          =   855
        Left            =   240
        TabIndex        =   0
        Top             =   360
        Width           =   4455
    End
End
'This is the ABOUTBOX variable declaration area...
'We have none to declare!


Sub Command1_Click ()
'Unload the form and release the memory

    Unload Me   '"Me" refers to the current form.  New in VB 2.0
```

```
End Sub
```

```
TRAFFIC.FRM
ABOUTBOX.FRM
ProjWinSize=152,402,248,215
ProjWinShow=2
Title="TRAFFIC"
ExeName="TRAFFIC.EXE"
Path="C:\VB"
```

BINARY is a 1 or 0.   Another way to look at it is TRUE (1) and FALSE (0).   Visual Basic defines TRUE as -1 and FALSE as 0.   However, if you simply think in TRUE and FALSE, you will be OK.

Enter our first Ca$h contest

    Show off your code and take a chance at winning our first Cash contest.
The contest is open to any Visual Basic for Windows programmer. You must create your application using
Visual Basic for Windows v1.0 or v2.0, and all code submitted must be your own original work.

    Here are the rules, and legalities:

 > The application must be your own work.
 >   It must be created in Visual Basic for Windows version 1.0 or 2.0.
 >   You may use only one Form, and or one Module for the application.
 >   The compiled EXE must be 20k or less in size. You do not have to
       include the EXE with your entry.
 * <u>Original source code remains the property of its creator.</u>

    We are not sure how big the Cash pot will be as of this printing because it is still growing. Every effort is
being made to make this pot attractive and we assure you there will be a handsome cash award for the
winner. The Winner will be selected by a panel of Visual Basic for Windows developers from several parts
of the U.S.. Entry deadline is March 15th 1993 and the Winner will be announced in the April 1993 issue
of Basically Visual Magazine.
    note: The contest is void where prohibited by law. Taxes on the prize money is the sole responsibility of
the prize winner.

 FORWARD ALL ENTRIES TO:

   The Light House
    1:2380/410   USR HST 14.4
    Flatrock, MI
or
   The Programmer's Pit BBS
    1:270/612   V32Bis
    York, PA

or MAIL YOUR ENTRY TO:

   The Light House
    23260 Woodruff Suite #4

Flatrock, MI   48134

OR
   The Programmer's Pit BBS
   P.O. Box 1214
   York, PA   17405-1214