Welcome
To the Windows Graphics Trainer
By *Denthor* of *Asphyxia*

Part 1

Introduction

Greetings all. It's been a while since my last trainer, mostly because of work. I am now a Software Developer (sound impressive?), and spend lazy days programming applications in Visual C++. Anyway, a few weeks back I read a message in a Usenet group from Scott Dodds, asking if there was a simple Asphyxia VGA trainer-like series for Windows. We couldn't seem to find one, so I decided to reincarnate the series for Windows.

The plusses for this are obvious. Have you *tried* getting a job as a DOS coder recently? Yet as a Windows programmer I had no end of opportunities waiting for me. Also, when I code something for SVGA, I like to know that ALL hardware on the planet is supported. And don't even get me started on multithreading.

Okay, so I've heard all the arguments against Windows before. Most of them are from people who don't actually know how the whole thing works. My favorite one is "Windows coders suck because Windows does everything for you. You just call the base libraries to get everything from spreadsheets to phong shaded graphics… takes two seconds…" The person who believes that deserves our sympathy. I know some amazingly intelligent Windows programmers (a lot of who migrated from DOS), and although the challenges are different, there are still some formidable challenges. That is why the clever ones are in demand.

Anyway, this series will follow pretty much the same format as the original, text section explaining everything and then the working sample source afterwards.

But wait, there is a bonus! Seeing as though this is for Windows (and at least the first few will be for 16 bit Win 3.1 type-stuff), I am using Word, with (wait for it…) a spell checker! Yes, all those legions of people who wrote to me correcting spelling mistakes and grammatical errors for the original series will have to attack my flawed logic process instead ;)

The sample code has been written and compiled in MSVC++ v1.52c 16-bit, and will run fine under Win 3.1 … the code should run without much modification under any Windows C compiler.

Easy Windows Stuff

This isn't a "How to code for Windows" tutorial, it's a "How to code graphics for Windows" tutorial. As such, I'm going to only lightly touch on all the base stuff… getting a book on it might not be a bad idea.

When you run your program, whatever is in the WinMain function is run.

```
int PASCAL WinMain (HANDLE hInstance, HANDLE hPrevInstance,
      LPSTR lpszCmdParam, int nCmdShow)
```

This is like main() in C or the main BEGIN in Pascal. In this, you create your window by registering its class with RegisterClass (a standard API function). In the class variable, you set the attributes of the window (has it got a menu, when do you want to get notifications, what is the background color etc.)

You then use CreateWindow with certain parameters to create the window (all this is in the example program, you will see how basic it is)

If you just left the procedure now, that would be the end, your application would terminate fine. But we want our window to hang around, so we construct a message-loop. Like so :

```
      MSG Message;
      while (GetMessage (&Message, 0, 0, 0))
      {
            TranslateMessage (&Message);
            DispatchMessage (&Message);
      }
```

GetMessage hangs around letting all other open windows do their thing until it receives a message meant for your application. Message is then filled in with the details, and the other two commands take care of it. If your specified a WndProc in your class, it will get called.

The WndProc is like a callback function… whenever your application gets a message, you get the chance to look at it first and do something about it. The format is :

```
long FAR PASCAL WndProc (HWND hWnd, WORD iMessage, WORD wParam,
      LONG lParam)
```

Windows always sends messages with these parameters. iMessage is the message such as WM_PAINT (to repaint the view). wParam and lParam change depending on what the message is.

That's about all there is to a small little application… with the above you have a window with a title bar, min/max/close buttons etc.

--

Drawing something to screen

To draw to a Windows view with normal API, you need to get its Device Context (HDC). The easiest way to do this is via GetDC…

```
      HDC pDC = GetDC(hWnd);
      ...
      ReleaseDC(hWnd, pDC);          // Release it, there are limited
                              //  DC's (about 5) in Win3.1
```

While you have an HDC, you can use all of the windows standard functions, e.g.

```
   SetPixel (pDC, nX, nY, RGB (0, 0, 255));
```

RGB is a macro, which returns a COLORREF with the specified red, green and blue attributes.

There are lots of other functions, such as MoveTo, LineTo, FillRect etc. Always remember though, that these are API calls, and aren't very fast… but we will use them for now.

The star field

Now we want a star field in our window. To do this I create a class. A class is like a group of variables and functions all put together, a bit like a souped up Record in Pascal. Here is an example :

```
class CBob
{
public:      // Other classes can see this data
   int m_nVar;
   void SetVar (int nVal) {m_nVar = nVal; }
};
```

Elsewhere you could say

```
CBob  myBob;
myBob.SetVar (100);
if (myBob.m_nVar == 100)
{
   ...       // This will get called
}
```

You don't have to have the function in the class structure, like so :

```
class CBob
{
public:      // Other classes can see this data
   int m_nVar;
   void SetVar (int nVal);
};

void CBob::SetVar (int nVal)
{
      m_nVar = nVal;
}
```

It's all the same.

Anyway, I have written a little CStarField class, which sets up, moves and draws our stars. But how do we get animation? One answer is to tell the system to send us a WM_TIMER message every few milliseconds. Every time we get this message, we move the stars a bit and redraw them. We also draw them in the WM_PAINT message in case they have stopped and we still want to show them.

```
      case WM_TIMER :
      {
            if (wParam == 1)
            {
                  g_csStars.MoveStarField (0, 0, -3);
                  HDC pDC = GetDC(hWnd);
                  g_csStars.DrawStarField (pDC);
                  ReleaseDC(hWnd, pDC);
            }
      }
```

Don't forget to kill the timer before you leave the application!

Another way involves the main message loop, and using PeekMessage instead of GetMessage. PeekMessage doesn't wait until the application gets a message, so we can update at least one frame every message that gets sent in Windows.

```
        while (TRUE)
        {
                g_csStars.MoveStarField (0, 0, -3);
                HDC pDC = GetDC(hWnd);
                g_csStars.DrawStarField (pDC);
                ReleaseDC(hWnd, pDC);

                if (PeekMessage (&Message, NULL, 0, 0, PM_REMOVE))
                {
                        if (Message.message == WM_QUIT)
                                return Message.wParam;

                        TranslateMessage (&Message);
                        DispatchMessage (&Message);
                }
        }
```

The above as the message loop will be much faster and smoother, but will put a greater strain on Windows… other applications will run slightly slower.

The CStarField is simple enough to grasp, if you have any trouble with the starfield theory, read VGA Tutorial 13 for information.

In Closing

So now we know how to do simple animation in Windows. But there is a lot left to learn... how to handle palettes like in DOS, how to draw bitmaps, how to do transparency and so much more. I'll be going over this in future tuts.

I am a fervent believer in object orientation (as you can see from the sample source code). I am a little less proud to admit that I often use MFC and the ClassWizards to get things done quickly... distributing a 600k DLL file of someone else's source code with my applications doesn't appeal to me. Anyway, I don't think I will use any MFC in any of these documents, it's a bit _too_ Microsoft specific, and I know that a lot of people use Borland and Watcom too.

In this archive I am including a little "How Windows Works" document I wrote a while back for some friends... check it out.

I don't have a Web page... but it is coming soon. Until then, you can mail me at dexter@iafrica.com ...

Byeeee.....
  Denthor