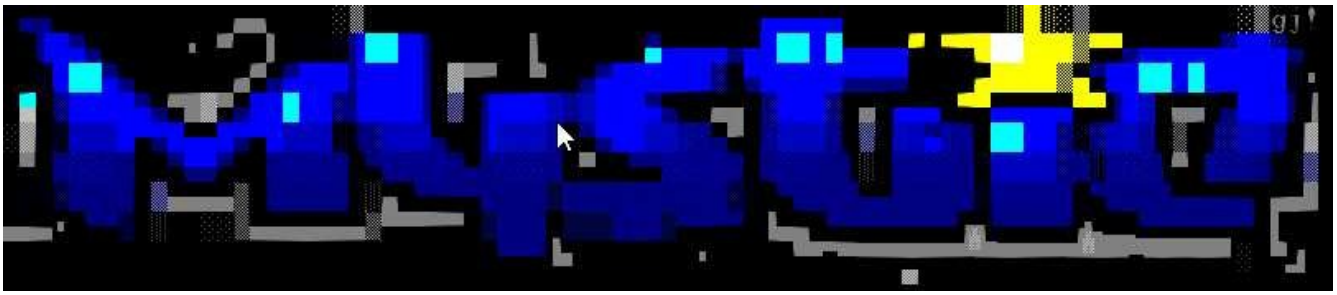

MPL Training
Lesson 14
Black Panther

Real MPL Example

-=-=-=-=-=-=-=-

For this lesson, let's take a look at something we would like to accomplish, and try to figure out how we can write some code that will do what we want it to.

-=-=-=-=-=-=-=-

Let's say, we would like to have text look like it's scrolling onto the screen from the side. Perhaps it's the Mystic Login prompt. (yep, this one's for you Paulie!)

We'll start with a very basic starting point for an MPL:

```
1   Uses
2     Cfg
3
4   Var
5
6   Begin
7
8   End
```

So, when we start we have a blank screen, so the first thing we need to do, is display an ANSI file. This could be something with a logo, or something that will identify your BBS.

```
1   Uses
2     Cfg
3
4   Var
5
6   Begin
7     DispFile('preuser')
8   End
```

That was pretty simple. Now, it will display an ANSI file, but we still need to get a scrolling login prompt.

If we were able slow the animation of the prompt down, we would see that there is one letter appearing on the screen in each step. So, we would probably start with the last letter appearing on the screen, and then move that over, and bring in the second to last letter.

Let's take a look at some code:

```
1   Uses
2     Cfg
3
4   Var
5
6   Begin
7     ClrScr
8     DispFile('preuser')
9     GotoXy(1,23)
10    Write(' ')
11    Delay(100)
12    GotoXY(1,23)
13    Write(': ')
14    Delay(100)
15    GotoXy(1,23)
16    Write('s:')
17  End
```

I'm sure you can see what's happening here, but I always like to clear the screen before doing anything, so I added the 'ClrScr' command. We then display our ANSI file.

The next line, 'GotoXY' will move the cursor to the X,Y coordinates that we pass in the parameters. So, the 1 is the first column on the left side of the screen, and the 23 is the 23rd line from the top. On an 80x25 display, this will be the third line up from the bottom. This is important to remember, because we will also need to add a prompt for the password prompt!

The 'Write' function is just writing a blank space. This is ensuring there is nothing there before we start to print to the screen.

The 'Delay' is basically a pause for MPL. It will stop the program for as long as you want it to, based on the parameters given. In this case, we will be pausing for 100 milliseconds. It's not a long time, but you want it to 'feel' as though the text is moving.

We then start the process over again, by moving the cursor back to 1,23. If we don't do this each time, the cursor would be at 2,23, and would create a mess of letters on the screen.

This code is very usable, and in fact was how my first iteration of this MPL was written. However, there is a better way to do this. What would happen if I wanted to change the delay time? I would have to change it in a bunch of places within the code. Why don't we use that lonely 'Var' in the code.

```
1   Uses
2     Cfg
3
4   Var
5     DelayTime : Byte=100
6     X         : Byte=1
7     Y         : Byte=23
8
9   Begin
10    ClrScr
11    DispFile('preuser')
12    GotoXy(X,Y)
13    Write(' ')
14    Delay(DelayTime)
15    GotoXY(X,Y)
16    Write(': ')
17    Delay(DelayTime)
18    GotoXy(X,Y)
19    Write('s:')
20  End
```

There we go. Now, if we want to change the delay time, or the location coordinates, we only need to change it in one place.

```
Uses
  Cfg

Var
  DelayTime : Byte=100
  X         : Byte=1
  Y         : Byte=23

Begin
  ClrScr
  DispFile('preuser')
  GotoXy(X,Y)
  Write(' ')
  Delay(DelayTime)
  GotoXY(X,Y)
  Write(': ')
  Delay(DelayTime)
  GotoXy(X,Y)
  Write('s:')
  Delay(delaytime)
  gotoxy(X,Y);
  write('as: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('ias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('lias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write(' alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('r alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('ur alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('our alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('your alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write(' your alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('r your alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('er your alias: ')
  Delay(delaytime)
  gotoxy(X,Y);
  write('ter your alias: ')
  Delay(delaytime)
```

```
60      gotoxy(X,Y);
61      write('nter your alias: ')
62      Delay(delaytime)
63      gotoxy(X,Y);
64      write('Enter your alias: ')
65      Delay(delaytime)
66      gotoxy(X,Y);
67      write('Enter your alias: |IF')
68  End
```

There we go! This will print ‘Enter your alias: ‘ on the screen, scrolling from the left side.

While this probably isn’t the ‘Best’ way to do this MPL, it works just fine. I probably could have done this inside a loop, and passed the text to the loop in a string, but why? Sometimes it’s just fine to do things the way you feel like doing them. In programming, there really is no ‘Wrong’ way of doing something. (some people may argue that point, but they would be ‘Wrong’) :)

What you need to do, when you have something that you want to happen in your MPL program, is to break it down into it’s simplest elements. In this case, it’s printing one letter onto the screen, each step in the process.

I may have used this analogy before, but it really makes sense when working with programming.

How do you eat an elephant?
One bite at a time.

Don’t look at the entire project, and feel overwhelmed at how complex it might be. Break it down into smaller and smaller segments, and focus on one segment at a time.

Oh, in case your wondering. The ‘|IF’ at the end of the last ‘Write’ statement, will not show a data entry field in the next command. In this case, when Mystic would show the blue box where the user enters their name, will not show. It will just be the black portion of the screen.

Also, you can use MCI color codes when doing a process like this. In my case, what I’m running on CRBBS right now, the last ‘Write’ line looks like:

Write(‘|11E|09nter |11y|09our |11a|09lias|01:|11 |IF’)

The last ‘|11’, right before the ‘|IF’, sets the color of the text field the user will be typing. So, in this case when the user types in their name, it will show in cyan.

Now, the password prompt is done using the same process. It just doesn’t display an ANSI, or clear the screen. It also prints to coordinates of 1,24.

Oh, and Paulie. For now, I’m not going to show how I’m displaying the ANSI falling onto the screen. :)