Drawing Text

You will likely want to draw some text from time to time. This is useful for printing status messages, displaying the score, and lots of other things. You could make a bitmap for every word/number that you might possibly want to draw...but that would probably not be much fun. Fortunately, we have added a new library function just for this purpose. Its prototype is in game.inc:

```
DrawStr PROTO myStr:PTR BYTE, x:DWORD, y:DWORD, color:DWORD
```

So the short story is that it takes a null terminated string and draws it on the screen at the starting position (x,y) and draws it on top of whatever is currently there with the chosen color. So if I had defined a string somewhere in the data section like:

```
whatStr BYTE “what now?”, 0
```

and then did this:

```
invoke DrawStr, offset whatStr, 20, 20, 0ffh
```

You would draw a nice simple string in white (ff). This function handles all of your standard ASCII printing needs except for special characters (e.g. line feed, carriage return, backspace). What if you need to print numbers? Well, you can also print formatted strings with a little help from `wsprintf`. This is a C function which can “prints” any formatted string into a buffer. Formatting allows you to output a mixture of plain text, numbers (different bases), characters, or strings. The formatting works essentially the same as it does for C printf and friends. The C prototype for the function looks something like this, note the variable arguments via the ...:

```
int __cdecl wsprintf(
  _Out_ LPTSTR  lpOut,
  _In_  LPCTSTR lpFmt,
  _In_          ...
);
```

To use it you would first need to declare your output and format strings in .DATA like this:

```
fmtStr BYTE "time: %d", 0
outStr BYTE 256 DUP(0)
```

Then you could do something like this:

```
rdtsc
push eax
push offset fmtStr
push offset outStr
call wsprintf
add esp, 12
invoke DrawStr, offset outStr, 300, 300, 255
```

This would print the time in cycles. Note the use of the calle stack adjustment. In general, this function takes a variable number of arguments (at least 2 for the format and output strings and

one for each formatted parameter). Don’t forget to add a null terminator to the output string, push arguments onto the stack in reverse order. Take care to use the appropriate stack adjustment -- if you don’t it will most likely cause your program to crash. This function is in the user32 library, so you will want to have the right includes:

```
include \masm32\include\user32.inc
includelib \masm32\lib\user32.lib
```