**Programmers with too much time on their hands**

I finally got around to reading the Sept. 14, 1993 *PC Magazine*, and came across the article ªMultithreading and Graphics Under Windows NTº by Charles Petzold. It describes a simple multi-threaded app: four windows are continually updated to display an increasing sequence of numbers, an increasing sequence of fibonacci numbers, an increasing series of primes, and a series of circles of random size drawn in random places.

As I'm pretty annoyed by all the hype surrounding NT, I figured I'd do it under NeXTSTEP. This is the result. It does nothing useful, really, just what's described above.

**Using the Program**

Start it up. Select the Run menu item. Look at the pretty pictures and text. When you've had enough, select the Stop menu item. Repeat as needed.

**Compare and Contrast**

So, what's the difference between the NeXTSTEP version and the NT version? The NeXTSTEP version is more functional, basically. You can copy & paste the text from the windows, print, fax, edit the text, and this help file is around. The NeXT version uses distributed objects as a communication tool, so it would be extremely easy to write a distributed app that wrote data back to the display windows. (Basically, you'd just have to cut & paste the code in the treads to a new app and compile). So, in theory, you could have some high-zoot piece of iron calculating bond yields or whatever,and updating your display on a humble Intel PC.

The NeXT version uses distributed objects to serialize output to the window server. The appkit is not thread-safe (making it so would be a big performance hitÐyou'd constantly be checking for mutex locks and the like), so only one thread can access it at a time. We do this by creating a server object,

and having the threads message the server. The server handles the client requests in order, ensuring that only one thread is writing to the screen at a time. Petzold says that ªexperimentation seems to show that Windows NT properly serializes access to the graphics functions.º So it's unclear to me that NT is supposed to be able to have simultaneous access to the window server from multiple threads, or if this just happens to work by happenstance and luck, like most PC software.

Output to the window continues during window dragging and the like. The server gets and dispatches remote events by looking during the main event loop.

The line count is pretty close to being the same for both programs. Petzold's looks to be around 300 lines long, eyeball estimate, while this runs around 600 according to wc. But I have a lot of comments and a coding style that uses a lot of whitespace. The number of lines with semicolons is around 200. I'll leave it to the reader to determine which is

more understandable.

Don't try to compare the two on speed. I haven't optimized this at all; it's pretty naive in a some places that count.

**Highlights**

·*Distributed Objects.* Fun stuff: write four client apps, and have them message the server object, each updating one of the views in the window. Five different processes communicating seamlessly. Tres qool.

·*Threads.* Amusing. Uses the cthreads package to fork and detach functions.

·*Drawing.* Hack some postscript and get more interesting pictures to display in window # 4.

Don McGregor
mcgredo@prism.cs.orst.edu