Subject: Re: Surprising Odds and Ends
Posted by promashkin on Mon, 14 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

I agree on how widespread Heap_gc needs to be in your code. To me, it is a command line thing that is useful to see if your program is "leaking", which may happen. I run it after quitting my programs while in development, to see if it finds something (I guess, help, /heap would do it but then I'd have to clean up anyway). I have, however, a related note. When my program crashes or freezes (or I had a bug that caused "leaking"), I call "Clear IDL" from the "RUN" menu. In essence, it is the same as resetting Xmanager (thus killing all widgets) and running Heap_gc, /Verbose:

widget_control, /reset & close, /all & heap_gc, /verbose & retall

I noticed that sometimes the first "Clear IDL" call does not print out any pointer (or object) information. Immediate second call, however, does find lost heap variables. I guess this is exactly what David explained. Somewhere a managed widget reflection still exists, and the first call resets Xmanager and kills that widget, releasing the State structure. You'd think Heap_gc would wipe the pointers in it. However, RETALL is the last command, and Heap_gc does not find any leaks when it is called from a last routine's level. Second "Clear IDL" finishes the pointers off.
It seems that moving RETALL to the third position in "Clear IDL" call would make sense. I know, I know, mucho guys don't use them wimpy menus. They have RETALL and HEAP_GC programmed into their keyboards :-)
However, I found one nice thing about Run menu: it is available even when your event handler gets into an infinite loop by (your) mistake, and does not even respond to keyboard break event.
Cheers,
Pavel