II.2

A SIMPLE FAST MEMORY ALLOCATOR

Steve Hill
University of Kent
Canterbury, Kent, United Kingdom

This Gem describes a simple memory allocation package which can be used in place of the traditional malloc() library function. The package maintains a linked list of memory blocks from which memory is allocated in a sequential fashion. If a block is exhausted, then memory is allocated from the next block. In the case that the next block is NULL, a new block is allocated using malloc().

We call the list of memory blocks a *pool*. A pool may be freed in its entirety, or may be reset. In the former case, the library function free() is used to return all the memory allocated for the pool to the system. In the latter case, no memory is freed, but the high-water mark of the pool is reset. This allows all the data allocated in the pool to be discarded in one operation with virtually no overhead. The memory in the pool is then ready for reuse and will not have to be re-allocated.

The package allows the programmer to create multiple pools, and to switch between them.

Some advantages of this scheme are:

- Memory allocation is fast.
- Data is likely to have greater locality.
- We no longer require a *free* routine for each data structure.
- Resetting the pool is extremely simple. This might replace many calls to the free() library routine.
- Space leaks are less likely.

The principal disadvantage is:

- Individual structures cannot be freed. This might lead to greater program residency.

The package has been used successfully in a ray tracing program. Two pools were used. The first pool holds the permanent data created whilst reading the model file. The second pool is for ephemeral data created during the rendering process. This pool is reset after each pixel has been calculated.

Incorporation of the package had three significant effects. Firstly, the program ran faster. The speed-up was not spectacular, but the program spends most of its time calculating intersections, not allocating memory. Secondly, the code for many operations became simpler. This was due to the elimination of calls to free memory. Finally, all space leaks were eradicated. The program had been worked on by a number of people, and in some cases calls to the appropriate memory de-allocation functions had been forgotten. Using the package eliminated the need for these calls; hence, the space leaks were also eliminated.