Pritsker Corporation

The work I did at Pritsker Corporation consisted of designing, programming, testing and documenting an animation system that effortlessly provided animations on a SLAM II simulation network. The system was called the Direct Network Animator, and it was an extension to the SLAMSYSTEM environment. Using Microsoft C and Windows SDK, I wrote about 7,000 lines of C for this project. For the Fortran portion of the project, I wrote 4,000 lines of code.

SLAM II is a simulation package that does combined discrete, continu­ous and network simulations. It is a general purpose simulation language that finds uses primarily in manufacturing and logistics planning. Input to SLAM II consists of a text file and an optional file of Fortran func­tions that can be linked in. Output from SLAM II consists of a text file containing informa­tion on resource utilizations, queue lengths, etc. SLAM II is written in Fortran.

SLAMSYSTEM is a user friendly graphics interface that lays on top of SLAM II. It provides utilities for managing projects, building simulations graphically and displaying results graphi­cally. SLAMSYSTEM is written in C for the Windows 3.0 environ­ment.

In constructing the animator, I used the Network Builder, the program that is used to build the network graphically, as the basis for the animator. This gave the animator a look and feel that was consistent with the builder. Also, it contained routines that built the text input file used by SLAM II. These routines were used in building some of the mapping information between SLAM II and SLAMSYSTEM.

No connection existed between the graphic network objects, the generated SLAM II text input statements and the memory image of the SLAM II simulation network. Successfully achieving these mappings allowed informa­tion inside the SLAM II process to be mapped to the graphics information and vice versa. Since the animation ran after the simulation was completed and SLAM II did not maintain certain information needed by the animator, the animator had to maintain the state of the network (i.e. where entities were located, and the states of the network objects). Trace information consisting of records for each entity transac­tion and network object state changes were collected in a file while the SLAM II simulation ran. Producing the trace required numerous changes to the code of SLAM II.

For the animation, trace records were pulled out of the trace file and used to update the internal representation of the network model. The internal representation consisted of a generic network object for each node and activity in the network. Each generic network object contained a list of entities and state variables. Processing the trace records caused changes in the internal representation. As the internal representation changed, the image of the displayed network object also changed. Each type of network object (i.e. queue, resource, activity, etc.) had a set of graphic methods associated with it. These methods dif­fered for each type of network object and were responsible for displaying the changing internal representa­tions.

The animator consisted of a client window that displayed the visible portion of the network. Scroll bars allowed the user to move around the network model. Also, the user could zoom in and out of the model while the animation ran. A modeless dialogue box displayed the current time in the simulation; it was always present.

The animator used bitmapped graphics. Two types of graphics existed in the animation: 1) objects that would remain station­ary and would later disappear, 2) objects that would move along linear multi-segment paths at various rates. I maintained the animation display by using two bitmaps. One contained the background image of the visible portion of the network. The other contained a background image with the animation symbols superim­posed. The second bitmap was copied to the client window of the animator for each animation update. A timer was set to advance each frame of the animation. The animator had interesting functionality. The user could change the animation parameters, jump around in simulation time and inspect the attribute values of entities flowing through the system. Entities are the units of information that traverse a SLAM II network. They contain attributes that are integers and floating point numbers. The user inspect­ed the entities by clicking the mouse on the graphic symbol representing the entity. For the system, I wrote a memory allocator that gave the applica­tion a dynamic memory scheme that was portable.