Q: What are timer events useful for? How are they different from timed entries?

A: These two different event-handling features are often confused because of the similarities of their names. However, they are intended for different uses. A timed entry is used for scheduling **regular** periodic activities in your application. See ../AppKit/timed_entries.rtf for more information about timed entries. A timer event is used in conjunction with a modal loop when an application must continue to do something even when no user events are being received. Modal loops are used to **temporarily** circumvent the primary application loop. The loop is triggered by an event such as a mouse down event, and is terminated when a specific event is encountered, such as a mouse up event.

The scroll buttons in the standard NeXT Scroller use a timer event in a modal loop to scroll continuously while the mouse button is held down. When the Scroller receives a mouse down event on a scroll button, it begins to scroll the contents of the view. While the user is simply holding down the mouse button, no events are generated. The application must continue to scroll the view, even though the events have stopped. This is when timer events come into play. Once you start a timer, it will insert timer events into the queue at regular time intervals. These events are ªdummyº events Ðthe user has not actually done something (moved the mouse, hit a key, let the mouse up, etc) but the event indicates that a certain time interval has passed since the last event. Turn on the timer as the modal loop begins, and turn it off when the modal loop terminates. (If you forget to turn it off, you may suffer performance problems because of the extra event processing!) During the execution of the modal loop, you call **getNextEvent**: with the appropriate event mask to receive timer events as they are generated and do the desired processing for each one. Here is an example of a modal loop which implements the scrolling behavior described above:

- mouseDown:(NXEvent *) thisEvent

```
{
        int                     shouldLoop = YES;
        int                     oldMask;
        NXTrackingTimer  myTimer;
        NXEvent                 *nextEvent, lastEvent;

        oldMask = [window addToEventMask:NX_LMOUSEDRAGGEDMASK];
        lastEvent = thisEvent;
        NXBeginTimer(&myTimer, 0.05, 0.05);

        while (shouldLoop) {
                nextEvent = [NXApp getNextEvent:(NX_LMOUSEUPMASK
                                                                        | NX_LMOUSEDRAGGEDMASK
                                                                        | NX_TIMERMASK)];
                switch (nextEvent->type) {
                case NX_LMOUSEUP:
                        shouldLoop = NO;
                        break;
                case NX_LMOUSEDRAGGED:
                        lastEvent = *nextEvent;
                        break;
                case NX_TIMER:
                        [self autoscroll:&lastEvent];
                        break;
                default:
                        break;
                }
        }
```

```
        NXEndTimer(&myTimer);
        [window setEventMask:oldMask];
        return self;
}
```

The code segment above was taken directly from the Concepts manual of the NeXT System Reference Manual. For more complete information about modal loops and timer events, please read Concepts Chapter 7.

In **/NextDeveloper/Examples/AppKit** there are two example programs (**ScrollDoodScroll** and **Draw**) which show how to use timer events with modal loops and give you an impression as to why they are useful.

Valid for 1.0, 2.0, 3.0

QA651