NeXTstep Measurement Kit: Recorders

The NeXTstep Measurement Kit consists of two modules. The first, Recorders, is discussed herein. The second, Historian, is discussed in its own documentation. Together, they

provide software developers with the ability to measure and view data about how users interact with user interface items. These capabilities can be added to existing source code with a minimum of effort.

Recorders is intended to be used by programmers, not end users. Thus, this documentation assumes familiarity with object-oriented programming, the NeXT Application Kit, and the Interface Builder.

Overview

The NeXTstep Measurement Kit (NMK) extends the functionality of Buttons, Sliders, Scrollers, and Menus to keep track of usage information. Time spent manipulating an item,

as well as time between uses, is measured. The number of invocations (hits) is also logged. For Sliders and Scrollers, the package keeps a histogram which shows where, within its range of values, the indicator has been set most frequently. Buttons also keep track of when the user clicks within a button and then drags outside the button in order to cancel the click. NeXT Scrollers consist of many parts and hits in each part are counted separately.

NMK is written in Objective-C and is designed to fit seamlessly into almost any application written in accordance with the NeXT Application Kit . Specifically, NMK consists of: custom classes for database management; a recording subclass of Application; subclasses that substitute for ButtonCell, SliderCell, Scroller, and MenuCell; and custom accounting classes for the substitutes.

In order fully to understand the role of NMK, one should work both with Recorders and Historian on a simple test application. A few cycles of recording followed by viewing should

help to familiarize one with the whole created by the interaction of these two halves.

Installation

This documentation will discuss how to add NMK to an application developed using Interface Builder. It is assumed that any users writing NeXT applications without the Interface Builder will already have the background required to adapt the following instructions to their particular situations.

The first step in the process is to add the necessary NMK source files to the project. This is done using the project inspector. A project containing these files is shown below:

proj.eps ¬

In addition to the class files shown above, the file "structs.h" should be added as ".h (other)".

Interface Builder must then be told about each of the new subclasses that has been added to the project. This is done in the classes window. To add a class properly, one must do the following. First, click on the class's superclass (parent class) so that only it is highlighted. Next, drag to "Subclass" in the "Operations" pull-down menu. Edit the text field in the lower right so that it shows the name of the new subclass and hit return. Next, drag to "Parse" in the "Operations" menu and answer "OK" to the alert box which appears. The subclasses of Object are shown installed below:

classes.eps ¬

The following subclasses must be installed:

Name:	**Superclass:**	
RButtonVars	Object	
RMenuVars	Object	
RScrollerVars	Object	
RSliderVars	Object	
TranscriptLinker	Object	
TranscriptManager	Object	
ButtonCellCover	ButtonCell	
MenuCellCover	MenuCell	
SliderCellCover	SliderCell	
RApplication	Application	
ScrollerCover	Control	*(because Scroller isn't shown)*

Finally, the owner of the main nib file must be changed from "Application" to "RApplication". This is done using the attributes inspector for the "File's Owner" icon of the appropriate file. Once this has been completed, the project can be saved and built using the "Make" command.

Classes

TranscriptManager & TranscriptLinker

The two subclasses of Object, TranscriptManager and TranscriptLinker, implement the internal database and are probably not of general interest. TranscriptManager handles the

transcript files (with the .trnscrpt suffix) which store recorded data for each application between runs. Transcript files have the following icon.

trnscrpt.tiff ¬

These files are currently stored in the user's home directory, but by editing the file TranscriptManager.m, the path may be customized. TranscriptLinker takes care of the details involved with keeping records for many user interface items accurate across multiple invocations.

RApplication

This subclass of Application extends the functionality of the Application class in two ways.

First, it sends "poseAs:" messages which enable the "...Cover" classes to substitute for their superclasses. Thus, all messages intended for an object of class ButtonCell will actually go to an object of class ButtonCellCover. This may cause problems with applications which already rely on "poseAs:" messages for other features. This, however, is not standard practice except in some debugging circumstances. Second, RApplication handles the creation of a TranscriptLinker (which creates a TranscriptManager) and sends it the proper initialization and termination messages.

ButtonCellCover & RButtonVars

ButtonCellCover overrides some of the standard functionality of the ButtonCell class, which is used to implement pushbuttons, check boxes, and radio buttons. In order to do so, ButtonCellCover declares no new instance variables and no new public methods.

RButtonVars handles the storage and maintenance of recorded data for ButtonCellCover. There is one instance of RButtonVars for each ButtonCellCover, and RButtonVars communicates with TranscriptLinker. Each ButtonCellCover is identified by a "key" which consists of any text it contains, the value of its "tag", and the rectangle it occupies. ButtonCellCover operates through the "trackMouse:inRect:ofView:" method. Each time this method is invoked, the statistic "numHits" is incremented. Based on the return value of the method, "numCancel" may also be incremented. In addition, the number of seconds spent executing the method is recorded in "timeIn". The time between the end of the last invocation and the beginning of the current one is added to "timeBetween", unless at the first invocation of the current run. All of these statistic variables are maintained by RButtonVars and can be inspected using Historian.

SliderCellCover & RSliderVars

SliderCellCover and RSliderVars work analogously to ButtonCellCover and RButtonVars. The "key" for a SliderCellCover is similar to that for ButtonCellCover, but is based on the name associated with the Slider object. Note that there is no "numCancel" statistic for Sliders. An additional statistic "histogram" keeps track of value information as a ten-part counter . Specifically, every time the "trackMouse:inRect:ofView:" method returns, the value of the slider relative to its minimum and maximum values is calculated as an integer between 0 and 9. This integer is used to determine which part of the histogram to increment. The histogram is shown graphically in Historian.

ScrollerCover & RScrollerVars

ScrollerCover poses as Scroller, which is usually used for showing relatively smaller views of

large documents. Since Scrollers act as complicated Sliders, ScrollerCover adds to the features of SliderCellCover. A Scroller consists of a knob, two buttons with arrows, and what is known as a knob slot or jump area. When one of the buttons is Alternate-clicked, the buttons serve to scroll by a page, rather than a line. Scrollers generally do not have names or text associated with them and are usually identified in the "key" by their orientation: horizontal or vertical. In addition to measuring all the statistics measured by RSliderVars, RScrollerVars keeps track of the following hit counts: "numKNOB", "numDECPAGE", "numINCPAGE", "numDECLINE", "numINCLINE", and "numKNOBSLOT". These are based on the values returned by "hitPart".

MenuCellCover & RMenuVars

A MenuCell is essentially a ButtonCell which appears in a Menu. Thus, there is virtually no

difference between the behaviors of MenuCellCover and RMenuVars, and ButtonCellCover and RButtonVars.

Runtime Behavior

NMK utilizes what is known as a lazy approach to record-keeping. User interface items which are never invoked are never recorded--no "R...Vars" object is created and they will not appear in Historian. Once an interface element is invoked, its statistics will be part of the transcript file forever. If an application contains a help system, the interface objects in the help system will not be recorded until the first time a user interacts with the help system, after which time the records will be visible in Historian. If the application is run again and the help system is not used, the records in the transcript file will remain unchanged and will not be

removed.

The transcript file is updated in the "free" instance method of RApplication and thus only when an application terminates normally.

Recorders.rtfd -- user manual for recording interface items
NeXTstep Measurement Kit
by Alex Meyer <ameyer@phoenix.Princeton.EDU>
for computer science senior thesis
27 April 1992 -- created
28 April 1992 -- first draft completed
4 May 1992 -- first release