UndoManager

By Jeff Martin (jmartin@bozell.com)

Inherits From:	Object
Declared In:	UndoManager.h

Class Description

The UndoManager object is used to handle undo and redo on a document basis in an application. Objects in the application register changes to themselves by sending an "undo" message to an undoManager instance from within an undoable method.For Example:

```
- setRadius:(float)value
{
   // Give undoManager the target for this event, then send undo message.
   [[undoManager setUndoTarget:self] setRadius:radius];

   // Set the new value and return
   radius = value; return self;
}
```

The UndoManager keeps a group (or list) of records for each "undoable" change. Each record contains a pointer to an object, an action for that object to perform and the arguments for that action. Undo is achieved by executing the target/action information in these undo records.

To add several records to an undo group, enclose the registration messages between UndoManager's -beginUndoRecordGrouping and -endUndoRecordGrouping methods. Undo messages that are not surrounded by these methods are added to the undoList in a group by themselves.

Also since sometimes changes should be ignored (ie. all of the discrete movements of a shape in a drag loop should not be undoable, just the final overall move), the messages -disableUndoRegistration and -reenableUndoRegistration can be sent to control which events are accepted into an undo record. For example, you might call [undoManager disableUndoRegistration] on a mouseDown: to prevent methods that register themselves with the undoManager to register each mouse drag, then you would call [undoManager reenableUndoRegistration] on mouseUp: with another call that would undo the drag loop (ie, [undoManager moveSelectedObjectsTo:oldPoint]).

The UndoManager allows the User to set the number of levels of undo to maintain (-setLevelsOfUndo:). When the number of records exceeds this amount, the excess is removed from the end of the list.

The UndoManager can be instructed to free target or argument data when records are removed from the undoList (whether they fall off the end of the list or are executed during an undo command). This is achieved by the -freeUndoTarget and -freeUndoArgs methods. Additionally the -copyUndoArgs method will copy any pointers that are registered with the next message (it is then assumed that the UndoManager will free them).

To trigger an Undo record, simply call -undo. This places the UndoManager into a "redo" state. All messages that are registered during an Undo are kept in a "redo" list. Thus the undo manager provides for redo as well. Redo can be triggered with the -redo message. The redo list is emptied each time a new record is added to the undo list (because, redo should only work if there were previous undos).

Clients of the undoManager can be notified when an undo is executed by adding themselves to the UndoManager's delegate list(-addUndoDelegate:). The only notification messages are -undoManagerWillUndo: and -undoManagerDidUndo: (called before and after an undo or redo). This notification is useful to do perform functions that all undo events may require, like calling -display.

Instance Variables

```
id      undoList;       // The list that holds undo records
id      redoList;       // The list that holds redo records
int     disabled;       // Whether the UndoManager is accepting events
BOOL    undoing;        // Whether the UndoManager is currently undoing
BOOL    redoing;        // Whether the UndoManager is currently redoing
BOOL    recordGrouping;    // Whether UndoRecords are being grouped
int     levelsOfUndo;   // How many levels of undo/redo to record

id      target;         // Current target of registered undo messages
unsigned int freeArgsMask; // Stores which args to free
unsigned int copyArgsMask; // Make undo manager copy pointer args
id        delegateList;     // List of objects to be notified of UM changes
```

Method Types

```
// Grouping multiple UndoRecords into an undo event
- beginUndoRecordGrouping;
- endUndoRecordGrouping;

// Disable/Reenable UndoManager to prevent events from being added to undo list
- disableUndoRegistration;
- reenableUndoRegistration;

// Setting the current target for events that are received
- setUndoTarget:object;

// Setting the target or args of the next registered method to be freed when
//   they fall off the end of the undo/redo list or are executed
- freeUndoTarget;
- freeUndoArgs;
- freeUndoArgAt:(int)pos;

// Setting the target or args of the next registered method to be freed when
//   they fall off the end of the undo/redo list
- freeUndoTargetOnRecordDiscard;
- freeUndoArgsOnRecordDiscard;
- freeUndoArgOnRecordDiscardAt:(int)pos;

// Setting the target or args of the next registered method to be freed when
//   they are executed
- freeUndoTargetOnRecordExecute;
- freeUndoArgsOnRecordExecute;
- freeUndoArgOnRecordExecuteAt:(int)pos;

// Make UndoManager copy arguments(like objects or strings) for convenience
- copyUndoArgs;
- copyUndoArgAt:(int)pos;

// Make UndoManager copy arguments and free them when record is discarded
- copyUndoArgsFreeOnDiscard;
- copyUndoArgFreeOnDiscardAt:(int)pos;

// Make UndoManager copy arguments and free them when record is executed
- copyUndoArgsFreeOnExecute;
- copyUndoArgFreeOnExecuteAt:(int)pos;

// Copies the pointer args in undoRecord as requested by copyArgsMask
- copyUndoArgsForRecord:(UndoRecord *)undoRecord;

// Overridden to capture undo/redo messages to be added to current record
- forward:(SEL)aSelector :(marg_list)argFrame;
```

```
// Removes a record from the undo/redo list and dispatches the messages in it.
- undo:sender;
- redo:sender;

// Query and set the maximum length of the undo/redo list
- (int)levelsOfUndo;
- setLevelsOfUndo:(int)value;

// These methods add and remove objects that are to receive undo notification.
- addUndoDelegate:object;
- removeUndoDelegate:object;
- sendNotification:(SEL)action;

// Used internally to free the space used for an undo/redo groups and records
- discardRecordGroup:recordGroup;
- executeRecordGroup:recordGroup;
- freeUndoRecord:(UndoRecord *)undoRecord withFreeMask:(int)mask;

// Remove and Free all records currently stored in the UndoManager
- emptyUndoManager;
```

Instance Methods

- beginUndoRecordGrouping, - endUndoRecordGrouping

These methods allow the client of the UndoManager to place several undo records in one undo "event". Undo messages that are received while not surrounded by begin/end undoRecordGrouping are placed into the undo list individually.

- disableUndoRegistration, - reenableUndoRegistration

These methods allow the undomanager to ignore events. This is useful in a case like mouse dragging, where the application only wants the final move to be undoable and not all of the intermediate drags. These should be strategically placed in blocks and can be nested.

- setUndoTarget:object

This method sets the target of following undo records. This is necessary because the sender cannot be gleaned from the forward method. Returns self.

- freeUndoTarget, - freeUndoArgs, - freeUndoArgAt:(int)pos

These methods specify whether the UndoManager should free the target and/or its args when an undo record is executed or when it exceeds the number of levels of undo. These methods set bits in the freeArgsMask representing the argument position. Position 0 is the target. Position 1-15 represent the first 15 arguments to the method. Bits 0-15 in the freeArgsMask represent args to be freed when the record is discarded, while Bits 16-31 represent args to be freed when the record is executed.

- freeUndoTargetOnRecordDiscard, - freeUndoArgsOnRecordDiscard,
- freeUndoArgOnRecordDiscardAt:(int)pos

These methods specify whether the UndoManager should free the target and/or its args when an undo record exceeds the number of levels of undo. These methods set bits 0 - 15 in the freeArgsMask representing the argument position. Position 0 is the target. Position 1-15 represent the first 15 arguments to the method.

- freeUndoTargetOnRecordExecute, - freeUndoArgsOnRecordExecute,
- freeUndoArgOnRecordExecuteAt:(int)pos

These methods specify whether the UndoManager should free the target and/or its args when an undo record is executed. These methods set bits 16 - 31 in the freeArgsMask representing the argument position. Position 16 is the target. Position 17-31 represent the first 15 arguments to the method.

- copyUndoArgs, - copyUndoArgsAt:(int)pos

copyUndoArgs and copyUndoArgsAt: tell the UndoManager to copy pointer arguments(like objects or strings) for convenience. It is assumed that the UndoManager will free them when the undoRecord is freed. Returns self.
copyUndoArgsFreeOnDiscard and copyUndoArgsFreeOnExecute are just like copyUndoArgs but they specify whether to free the args when the record is discarded or when it is executed. They all Return self.

- copyUndoArgsForRecord:(UndoRecord *)undoRecord

copyUndoArgsForRecord is used internally and does the actual work of copying the pointers.

- forward:(SEL)aSelector :(marg_list)argFrame

This method is how events are registered with the Undo Manager. When the UndoManager receives a message that it doesn't respond to, it assumes that this is an undo event and associates it with the current target. This undoRecord is then added to the undoList.

- undo:sender

This message actually triggers the UndoManager to take the top undo record and send out all of the events. The record is then freed.

- redo

This message actually triggers the UndoManager to take the top redo record and send out all of the events. The record is then freed.

- (int)levelsOfUndo, - setLevelsOfUndo:(int)value

These methods allow programmatic manipulation for the levels of undo that an UndoManager maintains.

- addUndoDelegate:object, - removeUndoDelegate:object

These methods add and remove objects that are to receive undo notification. The two notifications that are currently supported are undoManagerWillUndo: and undoManagerDidUndo (sent before and after an Undo or Redo).

- discardRecordGroup:recordGroup;
- executeRecordGroup:recordGroup;
- freeUndoRecord:(UndoRecord *)undoRecord;

These methods free the individual records in a record group. discardRecordGroup: calls freeUndoRecord:withMask: with the discard part of the freeArgsMask while executeRecordGroup: executes the records in the record group then calls freeUndoRecord:withMask: with the execute part of the the freeArgsMask.
freeUndoRecord walks through the record arguments free those that the freeMask requests to be freed. Returns self.

- emptyUndoManager

emptyUndoManager removes all of the recordGroups from the undoList and redoList. An UndoManager client might do this when a document is saved. Returns self.

- free

Frees each undo group in the undo and redo lists then it frees the undo and redo lists.

Written by: Jeff Martin (jmartin@bozell.com)