About Undo in Draw

This new version of Draw implements a multi-level undo feature. The purpose of this example is to demonstrate how to add undo to an already existing application. We've designed the undo code so that the parts not directly concerned with Draw can be easily incorporated into your application. This document focuses on the Draw specific details of the undo code, rather than the reusable undo components. Before reading this document you should read the more general Undo documentation and maybe look over the reference documentation for the Change and ChangeManager classes. If you are not already familiar with the Draw example you will probably also want to read the more general Draw README document before diving in here.

Structure

This example is broken up into one main project and three subprojects, as follows:

Draw project	RW`original Draw example classes, slightly modified to work with the undo system.
undo.subproj	The reusable Change and ChangeManager classes.
textUndo.subproj	The reusable UndoText class and its supporting classes.
graphicsUndo.subproj	The custom subclasses of Change created for Draw specific changes such as resizing a graphic.

This example is broken up into one main project and three subprojects.

Classes in the Draw project

```
Responder
    Application
        DrawApp
    PageLayout
        DrawPageLayout
    View
        GraphicView
        GridView
        Ruler
        ScrollView
            SyncScrollView

Responder
```

```
    ChangeManager
        DrawDocument

Inspector
Graphic
    Circle
    Group
    Image
    PSGraphic
    Rectangle
    TextGraphic
    Tiff
    Scribble
        Polygon
    Line
        Curve
```

Classes in undo.subproj

```
Change
    MultipleChange
Responder
    ChangeManager
```

Classes in textUndo.subproj

```
Responder
    View
    Text
        UndoText
Change
    TextChange
        TextSelChange
            DeleteTextChange
            PasteTextChange
            TextSelColorChange
            WholeTextChange
                TypingTextChange
TextSelection
    CutSelection
```

Classes in graphicsUndo.subproj

```
Change
    CreateGraphicsChange
    EndEditingGraphicsChange
    GridChange
    PerformTextGraphicsChange
    StartEditingGraphicsChange
    UngroupGraphicsChange              -- uses OrderChangeDetail
    GraphicsChange
        DeleteGraphicsChange           -- uses OrderChangeDetail
            CutGraphicsChange          -- uses OrderChangeDetail
        GroupGraphicsChange            -- uses OrderChangeDetail
        LockGraphicsChange
        PasteGraphicsChange
        ReorderGraphicsChange          -- uses OrderChangeDetail
            BringToFrontGraphicsChange RWases OrderChangeDetail
            SendToBackGraphicsChange   -- uses OrderChangeDetail
        ResizeGraphicsChange
        UnlockGraphicsChange
        SimpleGraphicsChange
            AlignGraphicsChange        -- uses DimensionsChangeDetail
            AspectRatioGraphicsChange  -- uses DimensionsChangeDetail
            DimensionsGraphicsChange   -- uses DimensionsChangeDetail
            MoveGraphicsChange         -- uses MoveChangeDetail
            ArrowGraphicsChange        -- uses ArrowChangeDetail
            FillGraphicsChange         -- uses FillChangeDetail
            LineCapGraphicsChange      -- uses LineCapChangeDetail
            LineColorGraphicsChange    -- uses LineColorChangeDetail
            LineJoinGraphicsChange     -- uses LineJoinChangeDetail
            LineWidthGraphicsChange    -- uses LineWidthChangeDetail
            TextColorGraphicsChange    -- uses TextColorChangeDetail

ChangeDetail
    ArrowChangeDetail                  -- uses hierarchical ChangeDetails
    DimensionsChangeDetail
    FillChangeDetail                   -- uses hierarchical ChangeDetails
    LineCapChangeDetail                -- uses hierarchical ChangeDetails
    LineColorChangeDetail              -- uses hierarchical ChangeDetails
    LineJoinChangeDetail               -- uses hierarchical ChangeDetails
    LineWidthChangeDetail              -- uses hierarchical ChangeDetails
    MoveChangeDetail
```

```
OrderChangeDetail
TextColorChangeDetail                -- uses hierarchical ChangeDetails
```

The graphicsUndo subproject
The remainder of this document focuses on the undoing graphics changes. As we mentioned, the original Draw classes and the reusable undo classes are discussed in other documents.

The graphicsUndo subproject contains a couple dozen subclasses of Change which together undo almost everything that a user can do to the graphics in DrawÐthings like grouping graphics and changing their fill color. They don't undo things that the user does to documents, windows, or the applicationÐthings like bringing up the inspector, changing the page layout, changing the contents of the pasteboard, or showing the ruler. In general the things that users do to the graphics change their state and irreversible without an undo feature, whereas actions like showing the ruler are easily reversible anyway.

One common action which is noRWbdoable is changing which graphics are selected. We decided not to make this action undoable because it happens so frequently, is non-destructive, and the user probably wouldn't want to have to explicitly undo it in order to undo other ªrealº changes. In hindsight the undo code might have been somewhat shorter and simpler if we had made selection actions undoable. As it is we must always explicitly record the graphics that were currently selected when an action was performed, whereas if selections changes were undoable we would always be able to simply act on the current selection when undoing and redoing changes. A general rule of thumb is that your code will be simpler if you make *everything* undoable so that you don't have to worry about side effects caused by changes in state in the application since the change was originally done.

The graphics changes
Most of the undoable user actions have their own special subclass of Change. For instance the FillColorGraphicsChange class does nothing but handle changes to the fill color of the currently selected graphics. A few of the classes are more general. For instance DeleteGraphicsChange handles explicit user deletions but is also used by ServicesGraphicsChange when a services call results in the deletion of the currently selected graphics.

GraphicsChange and ChangeDetail
Most of the classes that handle graphics changes are subclasses of the abstract superclass

GraphicsChange. GraphicsChange works hand in hand with the ChangeDetail class. GraphicsChange assumes that the change may need to keep track of state information for each and every graphic involved in the change. For example, if the user changes the fill color of a selection of graphics then every graphic involved will need an instance of ChangeDetail to remember what its color was before the change. Just as there is a particular subclass of GraphicsChange that knows about fill color changes (FillColorGraphicsChange) there is also a particular subclass of ChangeDetail that knows about fill color changes (FillColorChangeDetail).

GraphicsChange provides a few basic services to its subclasses. It maintains a list of the graphics involved in the change and a list of the changeDetails associated with these graphics. The GraphicsChange **saveBeforeChange** method creates the list of graphics and initializes its contents to match the GraphicView's slist. The **saveBeforeChange** method creates the list ofRWPngeDetails and inserts one newly created ChangeDetail for each graphic in the graphics list. The particular subclass of ChangeDetail that's instantiated is chosen using [self changeDetailClass], so that FillColorGraphicsChange can override **changeDetailClass** to specify that it wants to use FillColorChangeDetail instances.

The default **undoChange** method will work for most subclasses of GraphicsChange. The **undoChange** method notes the bounds of the graphics before and after the change is actually undone and then redisplays the affected bounds and updates the inspector. To actually undo the change the **undoChange** method calls the **undoDetails** method, which subclasses must override to restore the state of the graphics and the graphicView. The default **redoChange** method is identical to the **undoChange** method except that it calls **redoDetails** instead of **undoDetails**.

SimpleGraphicsChange

Many of the graphics change classes are derived from the SimpleGraphicsChange class. SimpleGraphicsChange is a subclass of GraphicsChange, which provides a **saveBeforeChange** method to initialize the change with a set of graphics and changeDetails, and provides **undoChange** and **redoChange** methods that handle redisplaying the affected graphics. For many changes all that remains is to simply record the state of each graphic before the change and then toggle the state of each graphic with every undo and redo. The methods in SimpleGraphicsChange do just that. The **saveBeforeChange** method gives every changeDetail a chance to record the state of its graphic before the change. The **undoDetails** and **redoDetails**

methods are called by the GraphicsChange **undoChange** and **redoChange** methods and they simply tell every changeDetail to toggle the state of its graphic. The subclasses of SimpleGraphicsChange can be alarmingly short. See ArrowGraphicsChange for typical example.

Diving in

You should now have enough background to be comfortable diving into the source. We suggest starting with ArrowGraphicsChange because it's a simple, representative change. Have a look at the ArrowGraphicsChange class, its superclasses SimpleGraphicsChange and GraphicsChange, and its ChangeDetail class ARWQChangeDetail. You may then want to move on to the change classes that don't descend from GraphicsChange. They're more complicated but demonstrate the use of features like the **incorporateChange:** and **subsumeChange:** methods. Good luck!