

# More on Movie Clip Control

- The following notes add a few more tips on controlling movie clips in Flash
- To recap, the idea with video is to:
  - ◆ Import it into the library
  - ◆ Add it (with all of its frames) to a movie clip symbol
  - ◆ Drag the movie clip symbol to the stage and name it
  - ◆ Use ActionScript to make buttons invoke movie clip methods for playback
- In addition to just playing and pausing, a quick scan of the other things a movie clip can do include:
  - ◆ Frame-by-frame — use *prevFrame()* and *nextFrame()* to move one frame at a time
  - ◆ Frame properties — *\_totalframes* tells you how many frames are in the movie clip, while *\_currentframe* tells you which frame is currently visible
- We can use *\_totalframes* and *\_currentframe* to implement an even finer level of control, in conjunction with additional built-in Flash components — in this example, we'll use a `UIScrollBar` and a `TextInput` to implement jump-to-any-frame functionality
  - ◆ You'll find these widgets in the Components panel

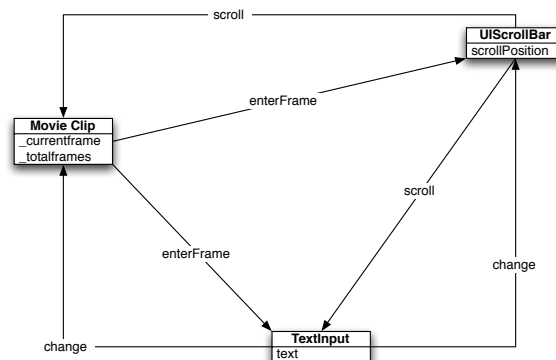
# Setting Up Direct Frame Access

- Here's the idea: by changing the `_currentframe` property of a movie clip, we can make it display any frame we choose; the acceptable range for `_currentframe` goes from 1 to `_totalframes`
- `UIScrollBar` is a convenient way to drag across a range of values; strictly speaking, it isn't meant for movie clip control, but it works, and what matters is the principle behind how we set everything up
- As an added bonus, we'll have a `TextInput` field display/change the current frame numerically

- This is what we do:

- ◆ Set up and name all three parts
- ◆ Give the `UIScrollBar` the correct minimum and maximum values (1 to `_totalframes`)
- ◆ Use ActionScript such that: (a) when the `UIScrollBar` changes value, it updates the movie clip `_currentframe` and the `TextInput`'s `text`; (b) when the `TextInput`'s `text` changes, it updates `_currentframe` and the `UIScrollBar`'s `scrollPosition`; and (c) when the movie clip displays a new frame (e.g., during playback), it updates `scrollPosition` and `text` in the other components

- This is a classic case of MVC, many times over: the widgets are the views, their properties are the models, and the arrows represent the controller



# Script Specifics

- The ActionScript fragment shown below illustrates the setup, assuming:
  - ◆ The movie clip is named *my\_mc*
  - ◆ The UIScrollBar is called *player\_scroll*
  - ◆ The TextInput is called *frameNumber*
- The script also illustrates a new way to handle events: create a new object, assign functions to that object corresponding to an event that you want to handle, then “inform” the component about this object

```
/**
 * This ActionScript should go in the first frame of the Actions layer
 * for the scene containing the movie clip you want to play.
 */

// Stop the movie clip.
my_mc.stop();

// Set up the UIScrollBar.
player_scroll.setScrollProperties(10, 1, my_mc._totalframes);

// Create the controller.
var playerController = new Object();

// Assign the scroll event function.
playerController.scroll = function(eventObject) {
    my_mc.gotoAndStop(player_scroll.scrollPosition);
    frameNumber.text = player_scroll.scrollPosition;
};

// Assign the text input change event function.
playerController.change = function(eventObject) {
    my_mc.gotoAndStop(frameNumber.text);
    player_scroll.scrollPosition = frameNumber.text;
};

// “Connect” the components to the controller.
player_scroll.addEventListener("scroll", playerController);
frameNumber.addEventListener("change", playerController);

// Assign a function to handle changes to the current frame (“enterFrame”).
my_mc.onEnterFrame = function() {
    player_scroll.scrollPosition = my_mc._currentframe;
    frameNumber.text = my_mc._currentframe;
};
```