

Simple Drag and Drop

Drag and drop sounds like a simple concept but is actually quite a technique to master, and easily complicated by additions to its functionality. In this lesson we will start off simple and turn it into something more complex as we travel through.

First of all you need to understand how Drag and Drop work and how to use them. Drag and Drop as the term suggests are individual commands within the actions scripting window and allow us to Drag an object around and Drop an object, obviously but the object needs to be quite specific. First we will try and implement the Drag command to get a feel for it working.

Drag

```
stop ○;  
startDrag ("");
```

I have added these commands to the first Keyframe on my timeline. 'Stop();' again out of habit, and the new command, 'StartDrag("");' (you will find the command in the Actions Book in the

ActionScripting window).

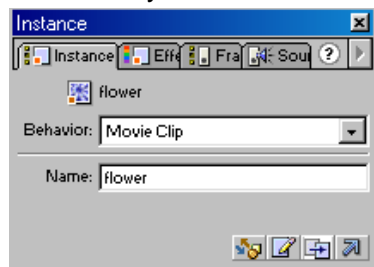
I have drawn a simple shape on the stage to illustrate the 'Drag' in action.



Try Control > Test Movie.


What you will find is that the whole movie is now attached to the mouse and you cannot shake it off. This is partly because the 'StartDrag' command is on the timeline and in this it effects the whole movie because the 'Drag' function acts on Movies, now there's a big clue!

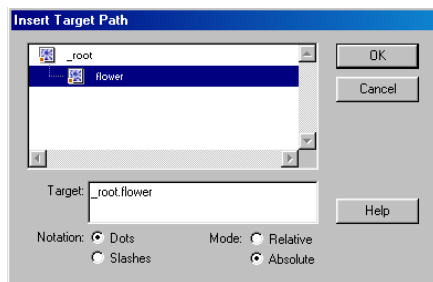
Now lets try and add another bit, I have turned the simple shape into a Movie Clip.



The instance of it on the stage I have called 'flower', now in the Keyframe change the Action Script to read.

```
Stop();  
StartDrag ("_root.flower", true);
```

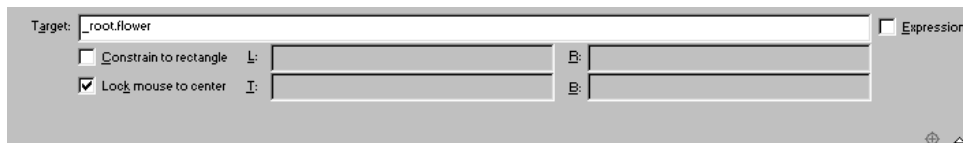
The second line is done by placing a 'Target' within the brackets, the target found by clicking on the  icon while in the 'target' box and navigating to the instance you just named. Press 'OK'.



The 'true' statement is added by ticking the 'lock Mouse to center' box.

Now try Control > Test Movie.

Now you will find that rather than the whole Flash movie moving about with the mouse only the embedded movie clip is attached to the mouse and what's more it is locked to the centre of it!



This really is the basics of Drag, but this only allows you control the movie clips automatically, what happens if you want to select an item to pick up?

Luckily for us this is possible and needs us to remove the actions we placed on the Keyframe at the start, leaving 'Stop();' though. Now we can start adding actions to the movie clip as we would if it were a button.

```
onClipEvent (mouseDown) {  
    startDrag ("");  
}
```

You will now see another new command appear automatically if you add the 'StartDrag("");' onto the Movie clip itself. It is known as 'OnClipEvent' and gives us a number of options when dealing with Movie clips, a few of which act as buttons. We will start to use these as much as the button controls now.

From the options given in the 'OnClipEvent () {' line choose 'mouseDown', this will simulate a button action of clicking down on the movie clip to activate the 'StartDrag("");'. Another point to mention here is because I am not outside of the movie clip I want to control I do not need to target it, although you may want to 'Lock to Centre'!

Try Control > Test Movie.

If you have taken the controls off the first Keyframe and put them on the movie clip then you should have the shape inactive until you click on it, from then on it is attached to your mouse.

How do I let go?

What happens if you want to Drop it? Well that should be relatively logical now.

Drop

Back in the Movie clip actions lets add the drop command. Go to the end of the scripting you just created:

```
OnClipEvent (mouseDown) {  
    StartDrag ("");  
}
```

Now double click on the 'StopDrag' option in the Actions book. It will look like this:

```
OnClipEvent(load){  
    StopDrag();  
}
```

Change the clip event to 'mouseUp' and now Try Control > Test Movie.

You have successfully created 'Drag' and 'Drop' commands which can now be implemented further.

Complicated Drag and Drop with Targets

Most drag and drop scenarios have some sort of target, the aim of which is to drag the object over the target and something happens or something loads etc.

I have made a 'bin' movie clip in my example to act as the target. I have given the name of its instance 'bin'.

Now the aim is to have a draggable and droppable item that is active until it is dropped over the 'Target' area and then we can get Flash to do something with that.

Now it starts getting a little more complex!

In the actions scripting on the flower movie clip instance let's go and add some more code. It should look something like this in the window.

```
onClipEvent (mouseDown) {
    startDrag ("", true);
}
onClipEvent (mouseUp) {
    stopDrag ();
}

if (eval(_root.flower._droptarget) ==
    _root.bin) {
    stopDrag ();
    TellTarget ("_root.flower") {
        gotoAndStop (2);
    }
}
}
```

The majority of this you should be happy with but we meet the 'if' statement for the first time and the '_droptarget', both of which have a specific way of being addressed.

if (eval(_root.flower._droptarget) == _root.bin) this is the line I am really most interested in, as it is this line that does all the work. So what's happening?

The 'if' statement is created by double clicking in the actions window under the actions book and looks like this to start with:

If (condition){}- The part in brackets is the part we need to change, Flash expecting some kind of conditional statement to activate the 'if' command.

(eval(_root.flower._droptarget) == _root.bin) – This creates the evaluative statement, hence the 'eval' at the start and then parentheses within the condition. Eval allows Flash to look at the statement and come to a conclusion, if it is not placed the condition will not be executed. '_root.flower._droptarget' tells Flash that the movie clip called flower on the main timeline is the drop target....NOT the target!

(eval(_root.flower._droptarget) == _root.bin) – Finishing off we use the equality operator '==', this means that something is equal to something else and in this example when the flower instance is directly over and dropped onto the bin instance the 'if' statement is true.

Tell Target then activates and moves the pointer in the movie clip 'flower' to the next keyframe and stops (you might want to add some stop commands in the flower movie clip to show that the '_droptarget' has occurred. I simply added a blank keyframe at keyframe (2) to simulate the item vanishing into the bin!

That is really as complicated as Drag and Drop gets apart from obviously remembering multiple objects with instance names and all of them going to different targets maybe.

The last thing to show you will allow you to create a more professional finish or even to add something as simple as a customised mouse.

Now you are happy with the drag and drop commands you can employ an extra line found down in the 'Objects' book within the action script window, navigate to Mouse and add 'Hide' to this section of the previous exercise:

```
onClipEvent (mouseDown) {  
startDrag ("", true);  
}
```

It should look something like this:

```
onClipEvent (mouseDown) {  
startDrag ("", true);  
Mouse.Hide ();  
}
```

In the same way add the 'Show' object to the second section of the code we did earlier:

```
onClipEvent (mouseUp) {  
stopDrag ();  
Mouse.Show ();  
if (eval(_root.flower._droptarget) == _root.bin) {  
stopDrag ();  
Mouse.Show ();  
TellTarget ("_root.flower") {  
gotoAndStop (2);  
}  
}  
}
```

N.B. *It needs to go in twice because the conditional statement gives flash two options, stopping the drag even if the drop target hasn't been met therefore re-establishing the mouse, and within the condition itself.*

Now try Control > Test Movie, what you will find is that on click the flower movie clip becomes the mouse and on release of the mouse the mouse arrow is restored giving better clarity to your animation and making it look more professional.

N.B. *Make sure that the 'Lock to Center' box has been ticked when using 'show' and 'hide' mouse as you will have no idea where the pointer is!*