Behaviours

Using built-in ActionScript

In this section

In this section, you will learn how to:

- attach ActionScript to objects using behaviours
- edit ActionScript generated by a behaviour
- load a movieclip using behaviours
- load a web page using behaviours

Behaviours

Behaviours are built-in scripts that you attach to an object to control how it behaves. Using behaviours is the simplest way to begin using ActionScript. They let you add interactive features without having to create the ActionScript code yourself.

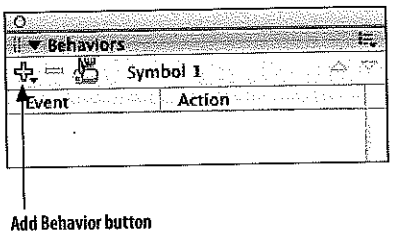
Behaviours are available for a variety of objects including movieclips, text files, video and audio files. New behaviours can also be downloaded from the Macromedia exchange website (*www.macromedia.com/exchange*).

To apply a behaviour to an object:

1 Select the object on the stage that you wish to apply the behaviour to.

2 Go to the *Behaviors* panel *(Window > Development Panels > Behaviors)* and click the *Add Behavior* button.

3 From the pop-up menu, select the behaviour you wish to apply.



exercise 2.1

Drag and drop movieclips

In this exercise you will use behaviours to allow a movieclip instance on the stage to be dragged and dropped with the mouse.

1 Open a new Flash document and save it as *Drag_and_Drop.fla.*

2 Rename layer one as *drag object.* In frame 1 of this layer, use the drawing tools to create a simple shape to drag.

3 Convert your shape into a *Movie Clip* symbol by selecting it, choosing *Modify > Convert to Symbol (F8)* and naming it appropriately.

4 In the *Properties* panel, give your movieclip the instance name *my_Drag_mc.*

5 Select the instance of your movieclip on the stage and in the *Behaviors* panel click on the *Add behavior* button.

6 From the pop-up menus that appear, choose *Movieclip > Start Dragging Movieclip.* In the dialog box that appears, select the movieclip instance name *my_Drag_mc* and make sure the relative path button is checked. Click *OK* and save your file.

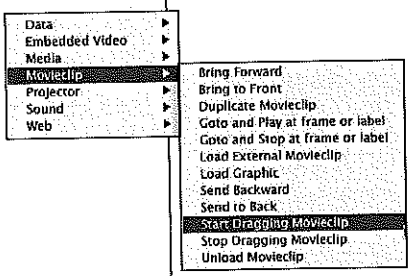
7 Test your movie (*Control > Test Movie*). When you click on your movieclip and then release the mouse button you should be able to drag your movieclip around the stage. You will not be able to drop it yet!

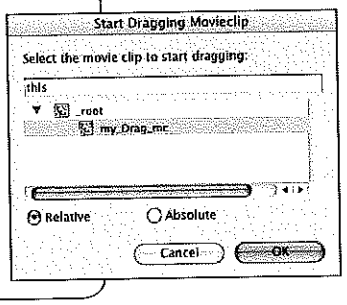
The event handler being used here is the default one *On Release,* but this is not ideal for the dragging action. This can be changed to *On Press* which would be better, you will do this next.

Continued next page...

tip **Naming movieclips**

It is recommended when naming movieclips, to finish the name with *_mc.* If you do this, when typing code into the *Actions* panel any code hints that appear will only be ones relevant to movieclips. The same applies to button names ending with *_btn.*





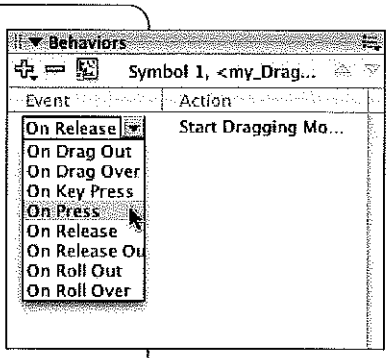
exercise 2.1

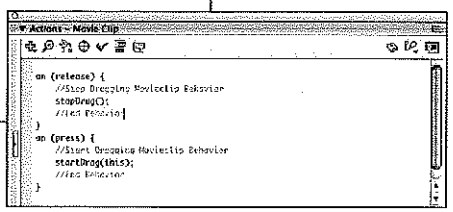
8 Return to Flash and go to the *Behaviors* panel. You will see the *On Release* event and the *Start Dragging Movieclip* action listed beside it. Click *On Release*, a pop-up will appear. Choose *On Press* to change the event that will trigger the dragging action.

9 Save and test your movie again, this time you should be able to drag the movieclip by just clicking on it.

10 To add the drop behaviour, repeat steps 5 and 6 but choose *Movieclip > Stop Dragging Movieclip*. A dialog box will appear, click *OK*. Save and test your movie.

11 To view the ActionScript code that has been automatically created for you, select the movieclip on the stage and go to the *Actions* panel *(Window > Development Panels > Actions* or *F9)*.





Interpreting the code

The following is the code that controls the dragging:

```
on (press) {
    // Start Dragging Movieclip Behavior
    startDrag(this);
    // End Behavior
}
```

- `on (press)` is the event handler that triggers the code within the curly braces when the mouse is clicked.
- Double forward slashes (`//`) are used to add comments to the code. These help the programmer, or others who may have to edit the code at a later date, to understand what the code is doing. Sensible commenting of code is good practice. ActionScript ignores these comments when the code is executed.
- `startDrag(this);` is a built-in method, that as the name suggests, allows the object to start being dragged.
- `this` is a reference to the object to which the method has been attached. In the exercise, this object was the *my_Drag_mc* instance of the movieclip.

tip Curly braces

A common error when writing code is to leave out a curly brace. Always check that the number of opening curly braces matches the number of closing ones.

exercise 2.2

Modifying code

Once a behaviour has been applied, you can édit the ActionScript code generated by the behaviour to create different effects.

1 Open your *Drag_and_Drop.fla* file (if it is not already open).

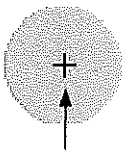
2 Select the *my_Drag_mc* movieclip instance on the stage and in the *Actions* panel, go to the code that was created when assigning the drag and drop behaviours.

3 Change `startDrag(this);` to `startDrag(this,true,0,0,200,200);`

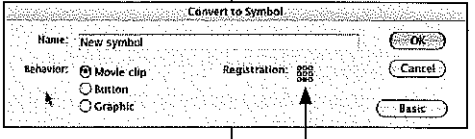
Save and test your movie. The parameters in the parentheses change the way the dragging occurs.

`true` snaps the mouse pointer to the registration point on the object (the + sign). If you put *false* the mouse just drags from where it is clicked on the object.

The numbers *0, 0, 200, 200* define the boundaries within which the registration point of the object can be moved. These numbers relate to *left, top, right* and *bottom* coordinates of the boundary in pixels.



Registration point



Clicking on one of these squares changes the position of the registration point.

4 Try changing the numbers to set different boundaries and change *true* to *false* and observe the effects. Save and test your movie again.

The hitTest method

When using drag and drop methods there is often a target you want the user to drop the object on to. By using the *hitTest* method in an *if* statement, ActionScript can determine whether an object is being dropped over a target or not.

The *hitTest* method is written like this:

```
this.hitTest(target_name);
```

where `target_name` is the instance name of the movieclip you have created as your target.

If the object you are dragging overlaps the target, the *hitTest* code will return the value 'true'. Otherwise it will return 'false'.

An *if* statement can test whether something is true or false and is written like this:

```
if (expression) {
   do this part only if the expression is true;
}
```

If the expression in the parentheses is false, the instructions in the curly braces are not actioned. You will learn more about *if* statements in a later chapter.

exercise 2.3

Using hitTest

In this exercise you will build a target into the drag and drop movie created in exercises 1.1 and 1.2, and write some code to test whether the object is dropped onto the target. If it is dropped on the target the word 'hit' will be displayed in an output box. If it is not dropped on the target, no message will be output.

1 Open your *Drag_and_Drop.fla* file from exercise 2.1 or 2.2 (if it is not already open) and add a new layer to the timeline. Name this layer *target*.

2 In frame 1 of the *target* layer, use the drawing tools to create a target shape on the stage. Convert this shape to a *Movie Clip* symbol *(Modify > Convert to Symbol* or *F8)*, naming it appropriately. In the *Properties* panel, give your movieclip the instance name *my_target*.

3 Go to frame 1 of the *drag object* layer and select the object you will be dragging and dropping. Then go to the *Actions* panel and alter the *On Release* code to:

```
on (release) {
    stopDrag();
    if (this.hitTest(_root.my_target)){
        trace("hit");
    }
}
```

The *trace* command will print the word 'hit' in an output box if the object is dropped on the target. The trace is included just to show the code works.

4 Save and test your movie.

5 The *hitTest* method can also be configured to test for a hit on a designated x and y coordinate on the stage instead of having to collide with a hit target. Go to frame 1 of the *drag object* layer, select the drag and drop object, go to the *Actions* panel and alter the *if* statement line to look like this:

```
if(this.hitTest(50,100,false)) {
```

Save and test your movie. You should get a successful hit each time any part of the object is dragged on to the point that has coordinates x=50, y=100.

6 Change *false* to *true*. Save and test your movie. A hit should only occur now when the registration point of the object matches the *hitTest* coordinates.

tip _root

When an instance name is preceded by *_root*, this means the instance is on the main (root) timeline. More on this in a later chapter.

tip Indenting code

It is good practice to indent your code to make it more readable.

Notice that in the examples provided in this guide, the closing curly brace always lines up with the beginning of the statement that contains its corresponding opening curly brace.

Object properties

In ActionScript, *properties* are things an object possesses, such as a size, a position, a colour. All properties are preceded by an underscore, for example, *_x* and *_y* represent the x and y coordinates of an object's registration point on the stage. In the following exercise you will see how the properties of an object can be altered using ActionScript.

exercise 2.4

Changing properties

In this exercise you will change the transparency of an object when it is being dragged and make the object jump to a fixed position if it is not dropped on the target.

1 Open your *Drag_and_Drop.fla* file from exercise 2.3 (if it is not already open). Modify it, if necessary, so you have a movieclip target with an instance name of *my_target*.

2 Go to frame 1 of the *drag object* layer and select the object you will be dragging and dropping. Next go to the *Actions* panel and amend the *On Press* code to:

```
on (press) {
    startDrag(this);
    this._alpha = 20;
}
```

The alpha value makes the object become semi-transparent while being dragged.

3 Also change the *On Release* code to:

```
on (release) {
    stopDrag();
    this._alpha = 100;
    if (!this.hitTest(_root.my_target)) {
        this._x = 0;
        this._y = 0;
    }
}
```

When the object is dropped, the alpha value restores its full colour.

Notice that the *if* statement contains an exclamation mark (!). This stands for 'not' and means the code in the curly braces will only be actioned if `this.hitTest (_root.my_target)` is '**not** true'. In other words the target has *not* been hit.

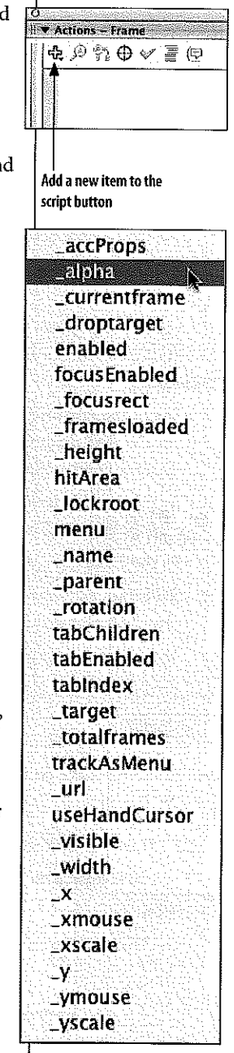
`this._x = 0;` and `this._y = 0;` make the object jump back to the top left corner if the target is missed.

4 Save and test your movie.

Extension

Experiment with this code by changing the properties of the object in different ways when it is being dragged, dropped successfully or dropped unsuccessfully. In particular you might like to try *_rotation*, *_height*, *_width* and *_visible*. For example,

```
this._rotation = 45;
this._visible = false;
```



This is a list of the properties available to movieclips. You access this list by choosing Built in Classes > Movie > MovieClip > Properties from the Add a new item to the script button in the Actions panel.