Design for Interaction – Lab 1: Introduction to Flash

Introduction – What is Flash?

Flash is a powerful authoring package for developing multimedia and for the web. Flash is owned and marketed by Adobe, and is part of their suite of development tools, which also includes Illustrator, Photoshop and Premiere.

The version of Flash that you will be using depends on the campus and lab you are based in. There are many similarities between the different versions and you are able to practice on others. However, be warned that they are not compatible and that work created in one version may not be editable in an older version. Flash comes with a scripting language called ActionScript. The version of this you will be using is ActionScript 3. There are also previous versions which are not compatible with ActionScript 3, so be aware of this if studying code examples from elsewhere.

In Flash you will work on a file with a .fla extension. This is your source file and should be saved regularly. From this you produce a shockwave file (with a .swf extension). This is the file that can be viewed using either a standalone viewer or as part of a web page using the Flash Player plugin (also known as Shockwave Flash).

In this module you will learn a range of Flash capabilities to construct interfaces and respond to user input. Flash is capable of much, much more in animation, games etc. and you may study these later on your course.

Flash uses vector graphics. An important quality of a vector format such as this is that presentations may be resized on screen to fit the available space without any deterioration in quality. This is very important on the web, where we cannot predict the size of a user's browser window. A Flash movie specified to run at full browser size will fit itself automatically into the space available.

In addition to these features, Flash supports a wide range of multimedia formats for importing images and sound, including MP3 audio. It also allows a certain level of interactivity to be incorporated into presentations. The Flash interface, while not exactly trivial to learn, is generally quite intuitive, and it is fairly easy to produce useful content within a relatively short timescale.

About This Workshop

This first Flash workshop is intended to familiarise you with Flash's workings and to give you the skills required for this module. It assumes no prior knowledge of the software, so don't worry if this is all new to you. If you run into difficulty at any point, ask one of the lab demonstrators. They should be able to point out what has gone wrong and point you in the right direction. If things are not working don't just go on – stop and get help. Remember that it is not just finishing the exercises that is important, but what you learn along the way.

In these notes, Flash menu items are shown in **bold**, and menu sub-options are indicated by the > symbol. Thus, you would open a new file by selecting **File > New**.

If the commands surrounding > are not in bold they refer to entering ActionScript. More of this later.

Getting Started With Flash

First start Flash. Flash should be in a program group called something like *Adobe Production Premium CS6* (a lower number indicates an older version). You should see something like:

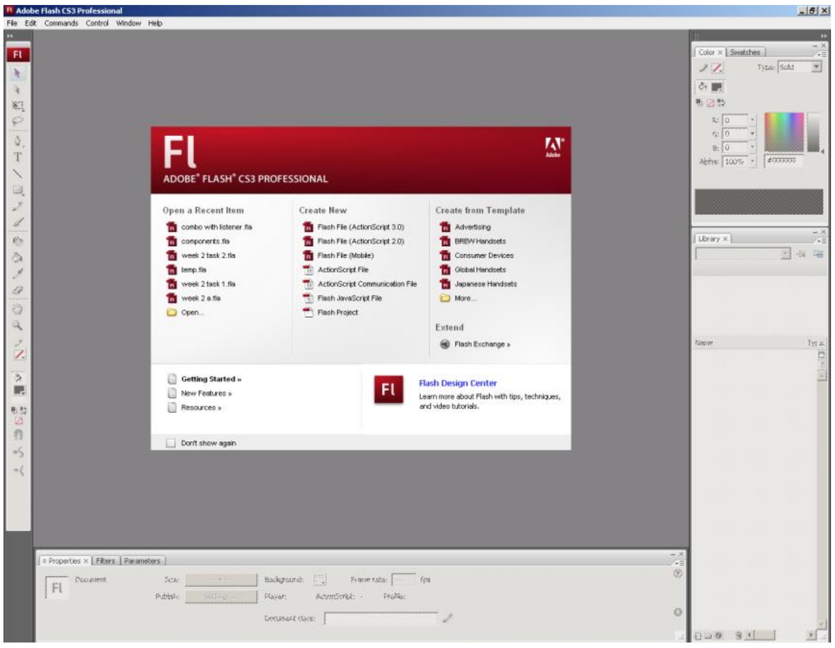
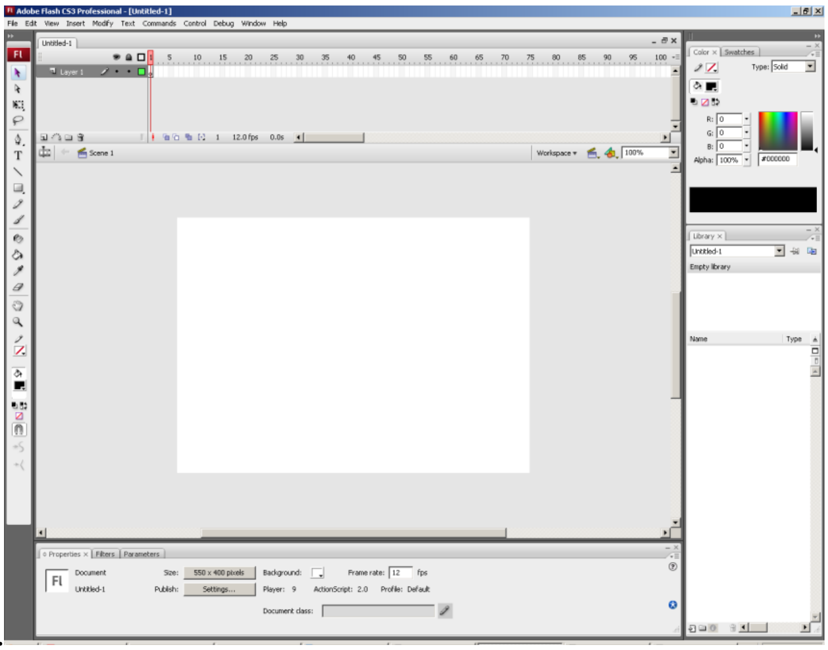


Fig 1: initial Flash screen

From the central area choose **Create New > Flash File (ActionScript 3)**.



The Flash interface can look intimidating but you will soon get to know how to use it. For this module you will not be covering all aspects of the interface.

We will begin by looking at the different components of Flash to get a flavour of how the programme works. If you don't see one of the elements described, use the **Window** menu to select the appropriate item. Identify each of the components on the screen as you read on.

The Stage

The blank area in the centre of the screen is the Stage. This is where the various graphical objects will appear as they are created and assembled.

The Stage is the area that defines the size of the movie on screen; the default size (usually) is 550x400. To modify the size of the stage you would select **Modify > Document** and change the height and width properties. For the purposes of this lab 550x400 (or greater) is fine.

The Toolbar

This should appear at the left of the screen. The Toolbar is used mainly when creating graphical objects, and its features are generally similar to those found in many graphics packages. Move the mouse over the various tools to see what they do.

The drawing tools are used to create objects in the work area that can be used as buttons or can be animated in various ways.

The Library

Objects created in Flash are stored as *symbols* in the Library. This is a useful facility because it allows you to reuse objects many times over, while altering their size, colour and various other characteristics. This has great benefits in reducing the file size of presentations. If the Library is not open select **Window > Library**.

The Properties Window

This appears below the stage. You may need to select the tab to open it. At present it refers to the document you are working on (referred to as a movie in earlier versions of Flash). As you select tools and other objects it will change to give information about them and allow you to modify their settings. The properties window can be hidden to show more of the stage.

The Actions Window

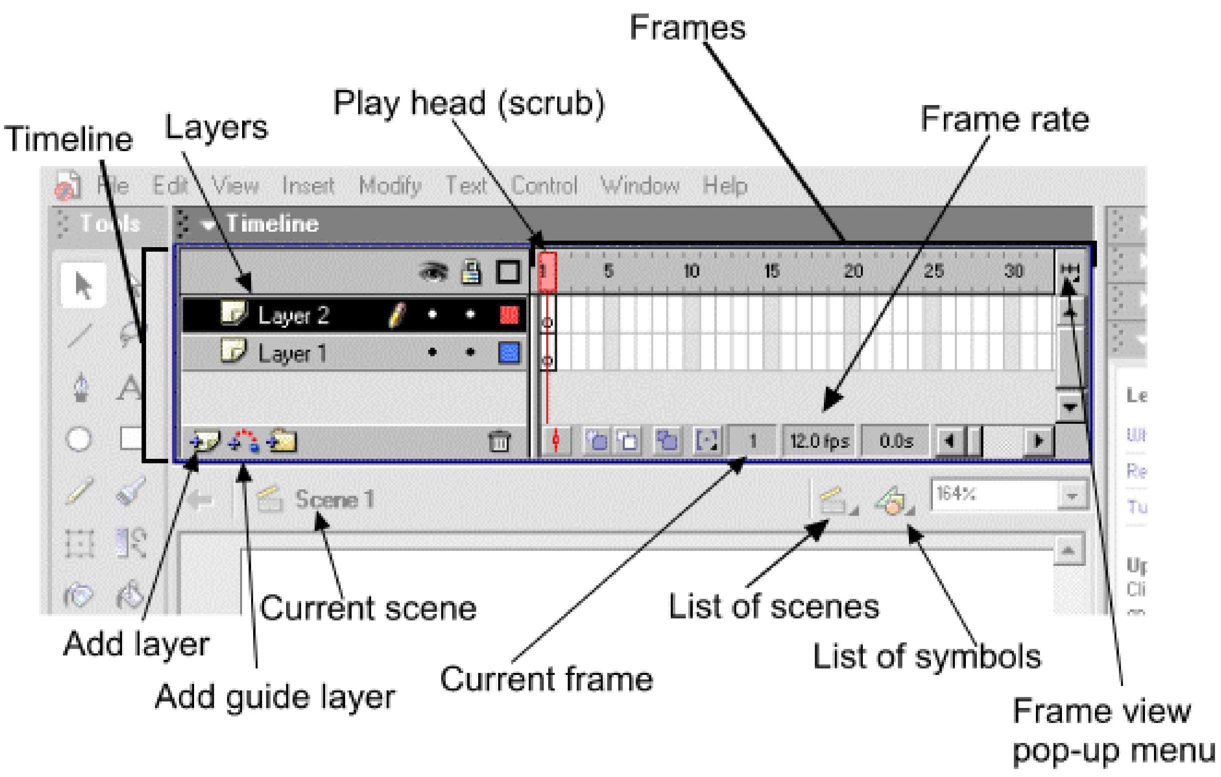
This can be opened by selecting the appropriate tab. When flash starts up the title of the tab is **Actions – Frame**.

The Timeline

This is where a Flash presentation is assembled. The Timeline is divided up horizontally into *frames*, and vertically into *layers*. Frames define the progress of the movie, with the red vertical line indicating the current *active* frame. In animation uses of flash frames are displayed sequentially. In interface work frames are used as separate displays.

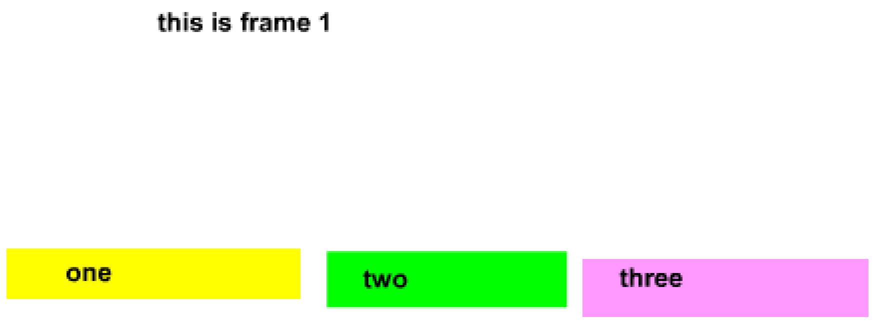
Layers allow different objects to be manipulated and animated separately in a movie. It is good practice to add a new layer for each new object as this causes no increase in file size and makes editing much easier. Note that the Timeline controls allow a layer to be both hidden from view and locked. Layers are also used for labels and actions. You will learn more about this later.

The figure below shows the key areas of the interface around the timeline.



Using Buttons to Navigate Between Screens

In this exercise you are going to create a simple navigation interface using buttons to move between different displays. It should end up looking a bit like the display below



First of all you need to create the three different screens that you will be using. This is done by putting keyframes in the timeline. The first keyframe is already there.

Click in frame 2 and using the menu that appears when you right click the mouse select Insert Keyframe. A circle will appear in the frame. Do the same in frame 3.

Now to make each of the frames look different. Select the first of the frames and from the Toolbar click on the text tool (the letter T). Notice how the properties box (underneath the stage) changes. From the dropdown menu to the left of this box make sure that static text is selected if it is not already. Now click on the stage and add some text to the top half of the screen ("Screen Number One" might be a good idea). You can alter the text size, font etc in the properties box. What has happened to the circle representing the keyframe?

Back in the timeline select frame 2. What you typed should disappear and the stage should revert to white. Add some static text ("Screen Number Two" might be a good idea) and then add something to frame 3.

Now go to **Control** > **Test Movie**. You should find that your movie loops through your 3 frames without pausing. You need to put a stop to this – exit the movie by clicking the cross in the top right hand corner to go back to editing mode. You need to add a simple piece of ActionScript to stop the movie.

Add a new layer by clicking the insert layer icon at the bottom of the layers section, and double click the new layer and change its name from *layer 2* to *actions*. We will use this layer to include frame actions. It is useful to group all frame actions into a separate layer. Here we will use an action to stop the movie in frame 1.

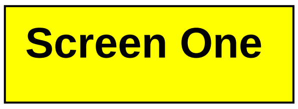
With frame 1 selected in the actions layer on the timeline click the Actions – Frame tab to open up the code window (you may need to expand this window) and type stop(); in the window.

Press `Ctrl+return`. This is another way to test your movie. Has this had the desired effect of stopping your movie in frame 1? If not, get help now.

If you haven't saved your movie already it is time to do it now.

Now add another new layer and change its name to *labels*. This layer is going to contain the target labels for the navigation system. Select frame 1 in this layer. In the properties box there is a text box that has the text <Frame Label> in it. Click in this box and enter the label *screenOne*. Insert a keyframe in frame 2 and give it the frame label *screenTwo* and likewise keyframe and label frame 3 *screenThree*.

Now to add the buttons that will allow navigation. Create another new layer and name it *navigation buttons*. Now click of frame one of the navigation buttons layer. You are going to create three buttons with actions to allow navigation to each screen. Notice that the new layers you add have extended over three frames but do not have keyframes other than in frame 1. This means that if you add something in frame 1 it will be visible in all three frames.



Click on the rectangle drawing tool and draw a yellow rectangle onto the stage. Use the illustration on the left as a guide to the size of the button. Click the text tool and add the words Screen One to the button. Using the select arrow tool click and drag over the whole button to select the rectangle and text and press `F8` to turn the object into a symbol. Select button from the behaviour options and give it the name `goOneButton`. Click OK. Your graphic is now a button. Check in the library window that it is there (**Window** > **Library**).

Repeat the procedure to create another two buttons with appropriate names. Your three buttons should now be on your stage, if not, making sure you are still on Frame 1 of the navigation layer, drag a copy of the buttons onto the stage from the library. Select each of the buttons in turn and, in the properties window, give them the instance names *go1Button*, *go2Button* and *go3Button*. These are the names of the buttons on the stage and will be how you will refer to them.

Now go back to the Actions – Frame script in frame 1, right click and choose actions. In the actions window add the following

```
go1Button.addEventListener(MouseEvent.CLICK, screenOneHandler);
go2Button.addEventListener(MouseEvent.CLICK, screenTwoHandler);
go3Button.addEventListener(MouseEvent.CLICK, screenThreeHandler);

function screenOneHandler(evt:MouseEvent):void {
      gotoAndStop("screenOne");
}

function screenTwoHandler(evt:MouseEvent):void {
      gotoAndStop("screenTwo");
}

function screenThreeHandler(evt:MouseEvent):void {
      gotoAndStop("screenThree");
}
```

The first three lines add event listeners to the buttons – this just tells the buttons that they must respond when clicked. How do the buttons know what to do when clicked? The event listener on each button refers to a function – so when *go1Button* is clicked the function `screenOneHandler` is implemented. The function tells the movie to go to and stop at the frame that has `screenOne` as a label. Don't forget that ActionScript is case sensitive and the label must exactly match what comes in between the quotes.

Now test your movie. The cursor changes to a pointing finger when over the buttons and by clicking you should be able to navigate to each of your screens with the button interface remaining in place. Is it working as expected?

Question: if you had used the gotoAndPlay action what would you have to add to the 2nd and 3rd frames?

Question: Flash allows you to navigate to frame numbers. Why is it better to use labels?

The buttons you created are functional and should make the appropriate action happen when clicked – however Flash allows the buttons to be more responsive to the user. Go back to frame 1 and double click on button 1. The button timeline should appear.

Buttons can also have various *states* attached to them that alter the way the button reacts to mouse events. These button states exist within the button's own timeline.

The timeline for a button is slightly different for that of a graphic or a movie. This timeline allows you to define properties for the button based on the whether or not the mouse pointer is over it or if the mouse button is up or down.

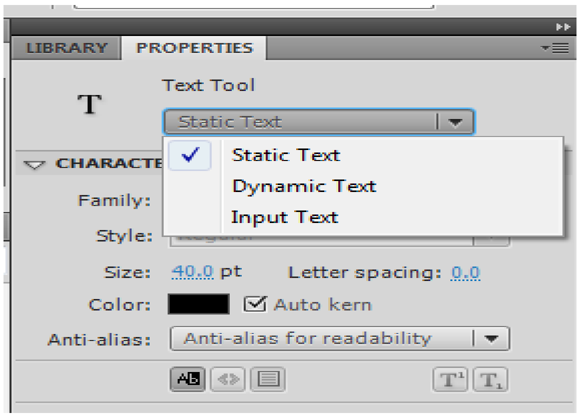
The first thing to do is right click on the Over state and insert keyframe and select the background colour of the button. This can be awkward and might need you to click on the stage and then back on the fill of the button. Now select the paint bucket tool choose a contrasting fill. Now right click in the Down state insert a keyframe, click on the background and choose a third fill. In order to give the appearance of the button being pressed for the Down state, hit the down arrow three times and then the right arrow three times your button should move. To get out of button edit mod double click on an empty piece of the stage.

The Hit state is not important for this example because the hit state is defined by the yellow rectangle. The hit state is used to define an area within which the button is active and is useful for small buttons and hotspots.

Preview the movie and check that moving the cursor over the button makes it turn red and that it turns light blue when clicked. Save your Flash File.

TEXT BOXES

Start a new Flash file. When you select the text tool from the tool palette the property inspector indicates all the options for formatting for you to determine how the text area is to behave. While most of these are fairly standard it is worth noting that among the choices is a drop-down box that allows you to define the characteristics of the text box you wish to use. You may need to expand the properties box to reveal the full range of properties.



Text Tool
Static Text
Family:
Style:
Size: 40.0 pt
Letter spacing: 0.0
Color:
Anti-alias: Anti-alias for readability

As can be seen in the drop-down list above, there are three types of text box that can be defined within a flash movie:

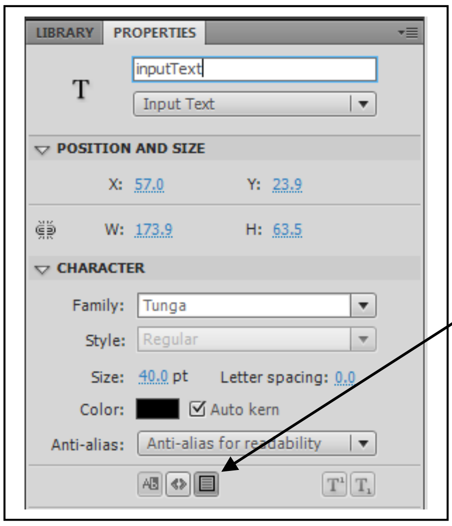
- Input Text
- Static Text
- Dynamic Text

The most common form, which you have used above, is the **Static Text** which would be used to hold labels, blocks of text, or any other textual content that is determined at design time and does not change. This is a little inflexible if you perhaps wanted to personalise a game or quiz by asking the user to enter their name that you would then display when asking questions or in a score table.

Input Text allows the user to provide textual input that can be processed by ActionScript within the movie. Each input text box can have associated with it an instance name by which it can be referenced so that text can be obtained from the box or output can be presented.

For longer messages and those of indeterminate length a **Dynamic Text** box can be used. This is not required for this module.

Text boxes allow data entry forms to be developed while the ActionScript capabilities of Flash allow data validation, formatting and other operations to be performed as we shall see in the next example.



inputText
Input Text

POSITION AND SIZE
X: 57.0 Y: 23.9
W: 173.9 H: 63.5

CHARACTER
Family: Tunga
Style: Regular
Size: 40.0 pt Letter spacing: 0.0
Color: ☑ Auto kern
Anti-alias: Anti-alias for readability

Create a text box. From the properties window select the type as input. Make sure that the text box has a **border** so that when you test the movie you can see it on the screen. In the *instance name* field of the properties window enter *inputText*.

Make another text box to the right of the first one – the properties should be the same except the *instance name* should be set to *outputText*.

Beneath the text add a button. Put the word "transfer" in the text box of your button. If the fill of the textbox is white, take off the border in the properties box. You should be able to do this now. Name your button transfer and give the button an instance name of *transferButton* in the properties window. When clicked we want this button to display whatever is in the left text box in the right text box – despite both boxes being called input they can be used for output too!

Create a new layer calling it Actions. In frame 1 of this layer open the actions window and type

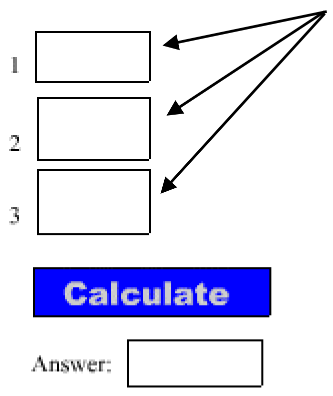
```
transferButton.addEventListener(MouseEvent.CLICK, myHandler);

function myHandler(evt:MouseEvent):void {
      outputText.text = inputText.text;
}
```

This adds the event listener to the button and calls a function when the button is clicked that takes the text from the text field named *inputText* and places it in the text field named *outputText*. Now run the movie. Enter some text into the left hand text box and press the button. The text should appear in the right hand box. Check it works and get help if it doesn't.

Exercise
The following is an example of using input text fields to allow users to enter data, a button to cause the data to be processed (using ActionScript) and a text field to output the result.

Start a new movie.



Create three text input text fields with instance names `num1`, `num2`, and `num3`. (Make sure the auto kern check box in the properties window is unchecked).

Give the field's text labels 1, 2, and 3 respectively as shown using static text fields.

Create a button called calc and with an instance name `Calculate` in the properties window.

Create an input text field with instance name `answer`.

Give this field a label as well as shown.

Your movie should be looking something like the illustration above.

In frame 1 enter the following ActionScript

```
Cvalculate.addEventListener(MouseEvent.CLICK, myHandler);


function myHandler(evt:MouseEvent):void {

      answer.text = num1.text + num2.text + num3.text;

}
```

Run the Flash movie and try out the script. What output do you get? Do you understand why?

Flash treats the textual input from the text input fields as text and not as numbers, because of this the + signs used in our script concatenate the text strings together instead of treating them like numbers and adding them up. We have to tell Flash explicitly to treat the input as numbers.

Change the function ActionScript to look like that below:

```
function myHandler(evt:MouseEvent):void {

      answer.text = String(Number(num1.text) + Number(num2.text) +
Number(num3.text));

}
```

The `Number()` function takes the text from a text field and turns it into a floating point number (if it can). The + operator then adds the numbers and they are then converted back into text (by the String function) for output.

Run the movie and enter some numbers in the fields, click Calculate and the result of the addition of the three numbers should appear in the answer field.

You can give yourself more screen area to edit your scripts with if you hide the actions list by clicking the button to the right of the list.

Previewing and Publishing the Movie

In previewing the movie, Flash will have created a compiled version, with .swf extension, in the same folder as your .fla file. You could place your movie online by writing an HTML page with this file embedded in it. However, the syntax to do this correctly is actually rather tricky, and we can make Flash do the work for us by using the Publish option. Simply go to **File > Publish**, and a new web page will be generated – in this case a file with the same name and a .html extension – with the compiled movie embedded. Check that the file plays back as expected. This code could be cut and paste to another web page.