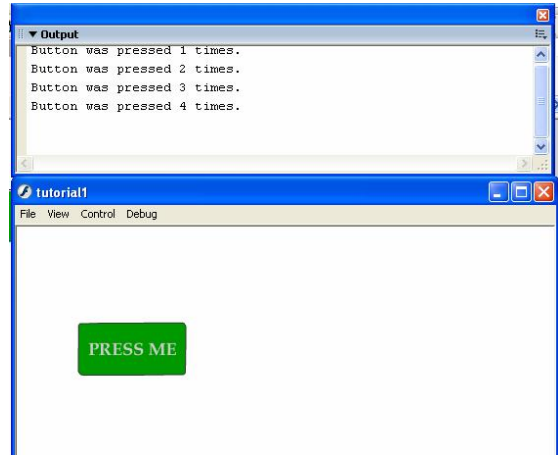


# Flash basics for teaching mathematics

## Tutorial 1. Making a button

by Doug Ensley, Shippensburg University and  
Barbara Kaskosz, University of Rhode Island

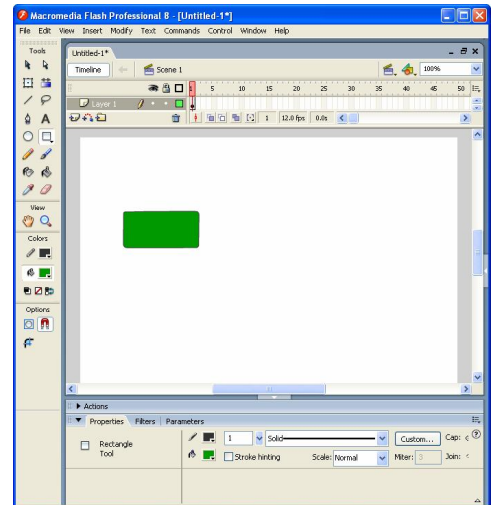
In this tutorial, we will make the Flash movie shown on the right. This simple movie consists of a button that when pressed, prints the number of times it has been pressed to the **Output** panel. Note that the **Output** panel is typically only used for tracing and debugging programs. For this reason, the output from this application can only be seen when the application is run from within the Flash authoring environment. We will learn more about how to integrate output into the movie itself in the next tutorial.



### Step 1. Draw the button

Start with a new project in the *Flash* authoring environment. You will need the **Tools** panel, the **Properties** panel and the **Stage** to be visible. All other panels can be closed if you wish. Choose the **Rectangle tool** from the **Tools** panel and draw a rectangle on the stage. Select the rectangle and change its border color, fill color, etc. so that it has the look that you want your button to have. Our example is shown below on the right.

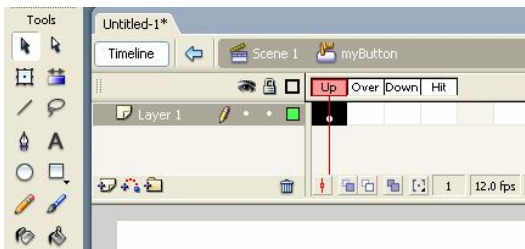
Choose the **Selection tool** (the black arrow) from the **Tools** panel, and double-click your shape or drag a selection rectangle around it to select the entire thing. Select **Convert to Symbol** from the **Modify** menu, and give this the name **myButton** and select the **Button** type. Your shape has been changed from a “graphics” object to a “button” object. This automatically endows it with special powers to detect mouse events. Your job now is to tell it how to behave as a result of these events.



### Step 2. Define the three states of the button

Double-click on the button and notice that the timeline has changed. You should drag the **Timeline** panel a bit more open if it does not already look like the screen shot shown below.

You are seeing the predefined timeline for the button, which is technically a movie clip *within* your movie. Your overall movie is considered to be in Scene 1, so you can see the hierarchy of “**myButton** within Scene 1” along the top of the **Timeline** panel. The predefined frames “Up”, “Over”, and “Down” correspond to the mouse being unclicked, rolled over, or clicked with respect to this button. The “Hit” frame is different, so we will discuss it a little later.



You must decide what you want the button to look like in each of these three states. We will leave ours as is for the “Up” state. Now click on the frame labeled “Over” in Layer 1. Select **Insert > Timeline > Keyframe** from the Flash menus, and a little dot will appear in the “Over” frame to indicate that it is now a keyframe. Make a change to your button that will indicate to the user that the button is active. For example, you can single click on the

interior of your button and change the fill color. We have chosen to double click on the border and change the line thickness from 1 to 3. This will make the border appear highlighted when the mouse is rolled over the button.

Now click on the frame labeled “Down” in Layer 1. Select **Insert > Timeline > Keyframe** from the Flash menus. Make a change to your button that will indicate to the user that the button is being clicked. We have chosen to double click on the border and hit delete to remove it altogether. You might prefer a simple color change of border or fill instead.

Finally select the frame labeled “Hit” in Layer 1, and select **Insert > Timeline > Keyframe** as before. The picture on the screen in this layer will never be seen by the user. Instead, it defines the region of the screen that will react to the mouse events. This is typically used when the button is irregularly shaped (like pure text) so that the user does not have to be precise with the mouse movements. It is also useful if you want to make invisible buttons or “hot spots” on your screen.

You have now successfully created a button. In the **Timeline** panel, click on “Scene 1” to get back to the main movie timeline. Save your file<sup>1</sup> and choose **Control > Test Movie** from the menus. Test your button by mousing over and clicking to see the different states you defined.

### Step 3. Adding text

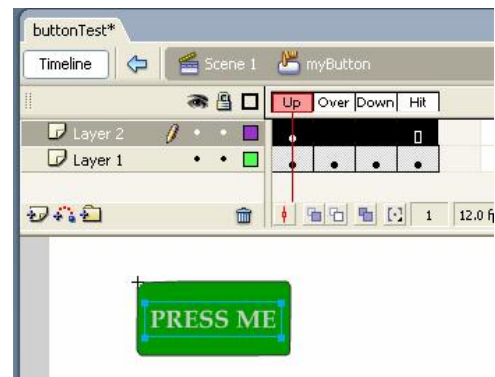
In many applications, there are several buttons on the screen, so it is not unusual to include text on the button. This can be done on the stage (i.e., simply lay a textbox on top of your button) or in the definition of the button itself. The latter method has the advantage that one could make text changes for each of the three states of the button. (For example, the text could turn red when the mouse moves over the button and green when the mouse button is down.) If we had laid a static text box with “PRESS ME” on our original rectangle before converting it to a button, then we could have gone this route.

The disadvantage to this comes from thinking ahead a bit. Typically when you have several buttons on the screen, you would like them to all be similar in size, shape or color scheme. Hence, the most typical way to make a button is to duplicate an existing button and make modifications. When there is different text attached to each frame of the button, this becomes a bit tiresome. We will use an alternate approach that makes duplication easier.

Double-click on your button in order to get the button time line open again. Click on “Layer 1” and choose **Insert > Timeline > Layer** from the menus. You will now see that a “Layer 2” has been created. Click on “Layer 2” and then choose the **Text tool** from the **Tools** panel. Draw a text box on the stage and add the text “PRESS ME.” Use the **Properties** panel to pick a reasonable font size and color and use your mouse (with the **Selection tool**) to position the text to line up over the button as shown on the right.

With this method, the text stays the same across all mouse events, but then again when you want to change the text, you have to only do it once.

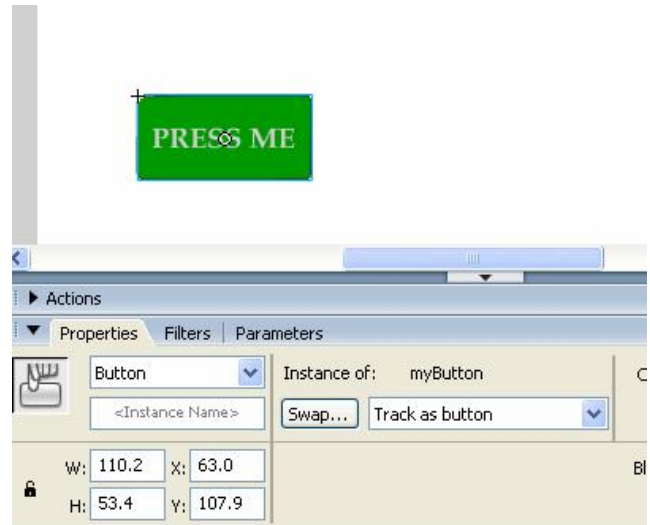
Go back to Scene 1 and save and test your movie.



<sup>1</sup> It is a good idea to create a separate folder for all of these tutorial exercises so they are easy to find later. Giving your project a meaningful name like **myButtonTest fla** will also be helpful. At the end of this tutorial, we will talk more about the files created by Flash and how this is relevant to posting your movies on a website.

#### Step 4. Making something happen

So far nothing really happens when we click on our new button. Before we address the script involved, we should be good Flash programmers and give our button an “instance name.” Single click on the button (Be sure you are in the Scene 1 timeline and not the **myButton** timeline!) and look at the **Properties** panel. The button currently has no Instance Name, so we should give it one so that it can be properly referenced from within our code. Let’s give it the name **pressBtn** to reflect its purpose. It is good to have a set naming convention in mind when you have several of these things on your stage.



Now we need to write a function (a method) to produce some outcome of a mouse click. To do this, click on Frame 1 of Layer 1, and open the **Actions** panel. (The panel should say Actions – Frame. If it says Actions – Button, then you need to click on Frame 1 of Layer 1 again.) Type the following lines of code into this panel:

```
var numPressed:Number = 0;

pressBtn.onRelease= function():Void {
    numPressed++;
    trace("Button was pressed " + String(numPressed) + " times.");
}
```

The “trace” function is typically used for showing values of variables while you are debugging a complicated program. We are using it here because it is an easy way to show output. We will see the correct way to produce output within the movie in the next tutorial.

Notice the declaration and initialization of the numeric variable **numPressed**, the “++” construct for incrementing the value of a variable, and finally the use of the “+” for string concatenation within the trace function call. These are all important ideas as we build more and more sophisticated projects.

Now save and test the movie again, and you will see the final movie functioning as we described.

#### Epilogue. A note about files

When you save this Flash project, the file is called (for example) **myButtonTest fla**. When you test it, you are creating **myButtonTest.swf** in the same directory. The swf file is the one that the Flash player browser plugin understands. To produce a website posting, you should choose **File > Publish** from the menus, and Flash will create the swf file along with an html file (filename.html) that contains it. Hence, you should post both **myButtonTest.html** AND **myButtonTest.swf** to the same directory of your website, and then direct students to **myButtonTest.html** for the activity.

There is not much point “publishing” the result of our button tutorial since the “trace” function does not do anything except within the Flash authoring environment where the **Output** panel is available. We will experiment with the html file in the next tutorial.

#### Additional resources

To see the source code for the button that we made for this tutorial, see **tutorial1 fla**.