This tutorial is a continuation of tutorial #3. The goal of this tutorial is to add an “on” state to the top navigation, so that the user knows what page they are on by glancing at the top navigation bar. We will do this using a variable – a powerful technique that is commonly used in flash website development.

setup

If you completed tutorial #3, your timeline for the “main” scene should look something like this:

1. The first thing that we need to do is make a movie clip called mc_rollstate. Inside this movie clip we will make the “on” states for each of our 7 pages.

2. Inside of this movie clip, we need to add keyframes and labels for each of our 7 sections. The easiest way to do this is to copy and paste the actions and labels layers from the mc_bodycontent movie clip. To do this, highlight the two layers inside mc_bodycontent, then copy all frames (EDIT > COPY FRAMES) then paste them into your new mc_rollstate movie clip.

3. Now we need to rename each of the labels with the prefix “roll_”. For instance the label “home” will become “roll_home”. We will do this so that we do not have any duplicate labels inside of our flash file.

When you are done, your timeline for mc_rollstate should look like this:
4. Next we need to add graphics for the roll state. The easiest way to do this, is to copy our buttons layer from the main timeline and paste them into a layer inside of mc_rollstate. We will do this because we already resized the buttons to fit each of the top nav sections, so we might as well copy and paste from there. But… first we need to temporarily put place a keyframe into the first frame of our but_universal button. We need to do this so that we can later break it apart to become our “on” state, or roll state graphic. This will make sense as we go through the following steps.

First open up but_universal and drag the “over” state keyframe into the first frame, the “up” state. It should look this this when you are done:

Next, copy the buttons layer from your main timeline and paste it into mc_rollstate.

Next, we need to make keyframes on this layer for each of our sections, then delete all of the buttons except for the one for that particular section. When you are done, your timeline for mc_rollstate should look like this:

The penultimate step of this process of making our “on” state graphics, is to break apart each of the keyframes, which are now buttons (but)univ). When you are done each should be a broken apart graphic that looks like this:

The last step is to open up but_univ and return it back to its previous state by removing the “up” keyframe and moving it to the “over” keyframe – thus returning it to its original state:

5. Now, we need to bring this movie clip onto a layer in the main timeline. In this case, let’s make a new layer and put it below the buttons layer and above the nav_bak layer. The easiest way add the mc_rollstate movie clip to this layer is to simply drag and drop it from the library. Once it is on the canvas, position it where it should be – if snapping is turned on, this should be very easy – then be sure to name the layer “mc_rollstate” and the instance of this move clip to be “mc_rollstate”. It is crucial that you name the instance of this, and every movie clip, whenever you will need to “talk” to it. If you have a hard time selecting it, simply lock all of the other layers so that you can select it easily.

6. Next, we need each button to “talk” to this movie clip and tell it which frame it needs to be on. The simplest and easiest way to do this is to simply add the line:

```javascript
_root.mc_rollstate.gotoAndStop("roll_home");
```
…to the actions panel inside of the on(release) brackets. You will need to add this to each one. Add this line of code to each button, renaming the targeted frame label for each one according, and then test your movie – it should work.

7. We are not done yet! Even though this works, let’s cleanup our code a little and see if we can do this a little more efficiently. One way to do this, is to use a variable instead of typing “home”, or “section 1” to each line of code.

To do this, we need to first declare a variable, then give it a value. We will do this in the main timeline at frame #1. But first, let’s move the “default” label keyframe, and it’s accompanying stop action layer keyframe over to frame #2 – in order to open up frame #1 so that we can declare our variable there. We want to do this so as to give us a place to name our variable when a user first comes to the home page.

After you move everything over, your “main” timeline should look like this:

8. Next, in frame #1 of your actions layer, lets open up the actions panel and make a new variable called “sectionID”. This variable will come in handy later. In this case, when the user first comes to frame 2 of this main timeline, they will be at the home page, therefore, let’s give section ID a value of “home”. Makes sense, right? We do this by typing this at frame#1 of your actions layer on the “main” timeline:

[sectionID="home"]

9. Now that we declared this variable, let’s return to our buttons, and use this variable in such a way that allows us to save us from renaming each action multiple times. For each button, inside the on(release) brackets, let’s change the value of sectionID to the appropriate section (“section1”, “section2”, etc.), and then let’s fix each of our movie clip lines to use the variable instead of hand coding each one. We do this using a programming technique known as concatenating strings. In computer science and programming practices, this is a very common thing in most all languages. Lucky for us, actionscript allows us to do this quite easily using the “add” command. Let’s change our code to look like this for each button:

Congratulations, you just used some advanced programming in order to create a relatively simple effect. In our next tutorial we will use this sectionID variable to add transitions in between sections.