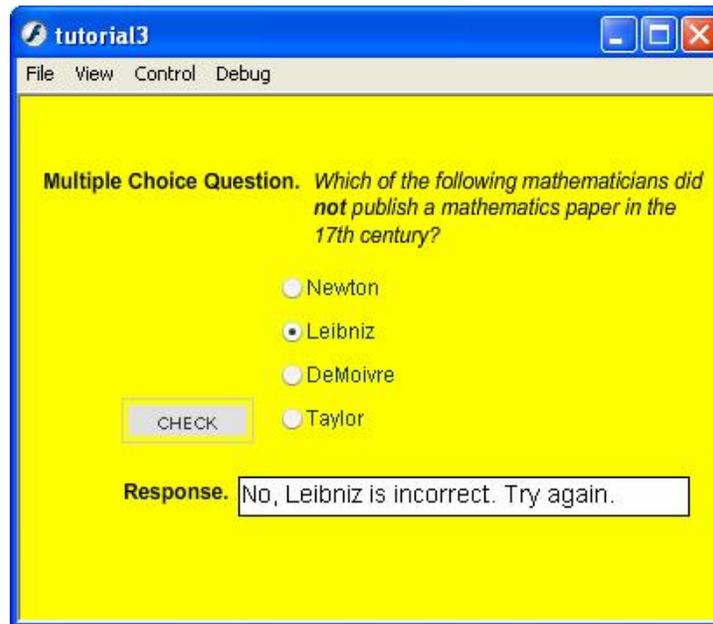


Flash basics for mathematics applets

Tutorial 3. Using the radio button component

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

In this tutorial, we will make the Flash movie shown below. This applet provides the basic functionality of a multiple choice question and gives appropriate feedback for correct and incorrect answers.



Step 1. Add all textboxes to the stage

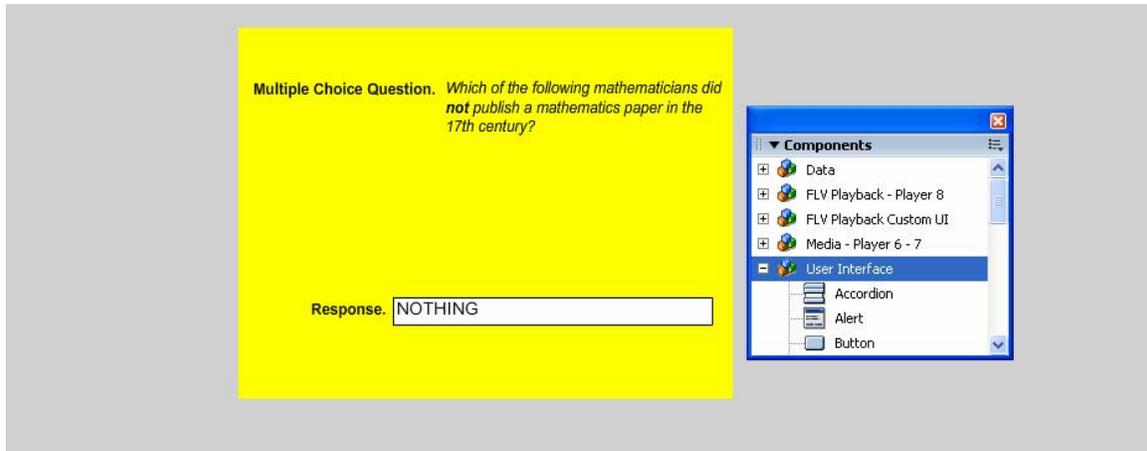
Start with a new project in the *Flash* authoring environment. You will need the **Tools** panel, the **Properties** panel and the **Stage** to be visible. All other panels can be closed if you wish. Just as you did in Tutorial 2, click on an empty section of the stage, and in the **Properties** panel set the size of the movie (400 by 300 pixels works well if you keep your text to 12-14 pt fonts), and choose a nice background color to appropriately contrast with your text. As before, save your file¹ in the tutorials folder you have created.

We will now add static textboxes to hold the question and label the “Response” box. Create a dynamic textbox called **messageBox** and line it up with the other text as shown in the example above. I like to include some text (like the word “NOTHING”) in my dynamic text boxes so that I can make appropriate decisions regarding font and alignment. The script that we write will dynamically clear all text boxes at run time, so it doesn’t matter what you put in the box. If you are unsure how to do this, see Tutorial 2, “Reading input text boxes and writing to dynamic text boxes.”

¹ 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 MultipleChoice will also be helpful. At the end of Tutorial 1, we talked about the files created by Flash and how this is relevant to posting your movies on a website.

Step 2. Create the radio buttons

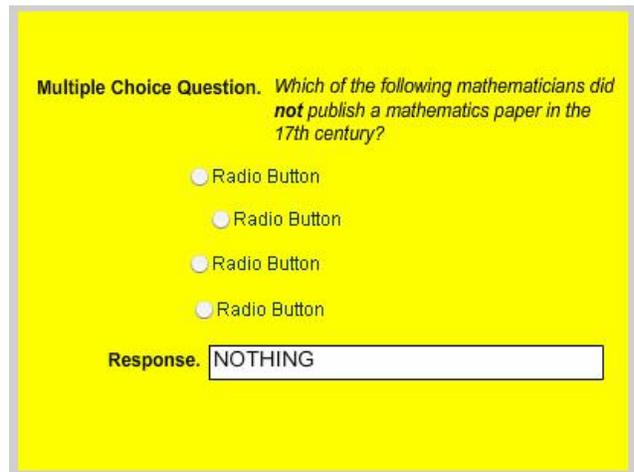
We will make a group of four radio buttons on the stage. Select **Components** from the **Windows** menu item, and expand the “User Interface” list. Your screen should look like the following:



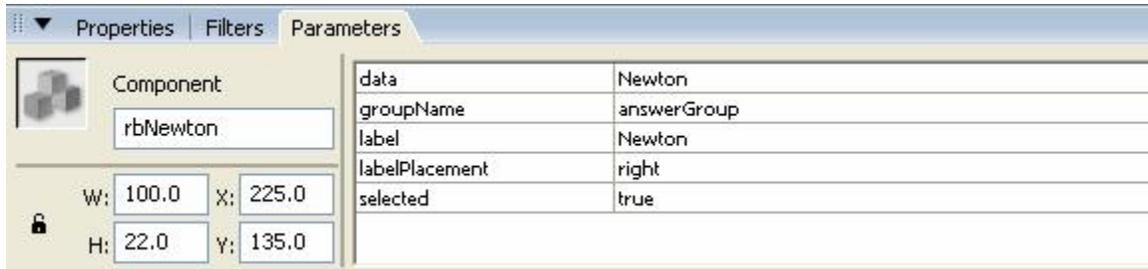
Scroll down the list of User Interface Components until you come to the Radio Button item. Click on this item and drag it onto your stage in approximately the correct position. (We will line everything up exactly later.) Repeat this process three more times so that you have four instances on the stage. Close the **Components** panel, and your stage will now look like this the screen shot shown on the right.

Now open the **Properties** panel and select the Parameters tab. When you click on the first radio button, you will see all of the relevant properties and parameters of that button. You should first specify an instance name (I used **rbNewton**) and X- and Y- position coordinates. I used X = 225 and Y = 135, but your placement will depend upon the rest of your layout². You can also set the height and width of the button/text field. The latter is useful when your text is too long to fit in the default (100 px) width. In my example, the defaults are fine.

The remaining parameters should be filled in as shown below. We include a few notes about what each of these fields means.



² When aligning a group of objects, I usually place the first one by dragging it with the mouse and then adjusting the coordinates to that they are nice whole numbers. I then use coordinates in the Properties panel to line and space the others correctly.



Notes:

1. The **data** field contains the value that will be returned when this button is selected. We have chosen to return the exact string that is in the radio button label. It is also common to return simple numerical values to represent which of the radio buttons (1, 2, 3 or 4) was selected.
2. The **groupName** field allows the four buttons to behave as a unit. All four radio buttons must have the same group name for this to happen. We have used **answerGroup** as our groupName.
3. The **label** field will contain the exact words that appear next to the button. The relative placement of the label to the button is determined by the **labelPlacement** field.
4. The **selected** parameter indicates whether that radio button is initially selected. It only makes sense for one radio button in a group to be selected, so since we have made this one *true*, the other three will be *false*.

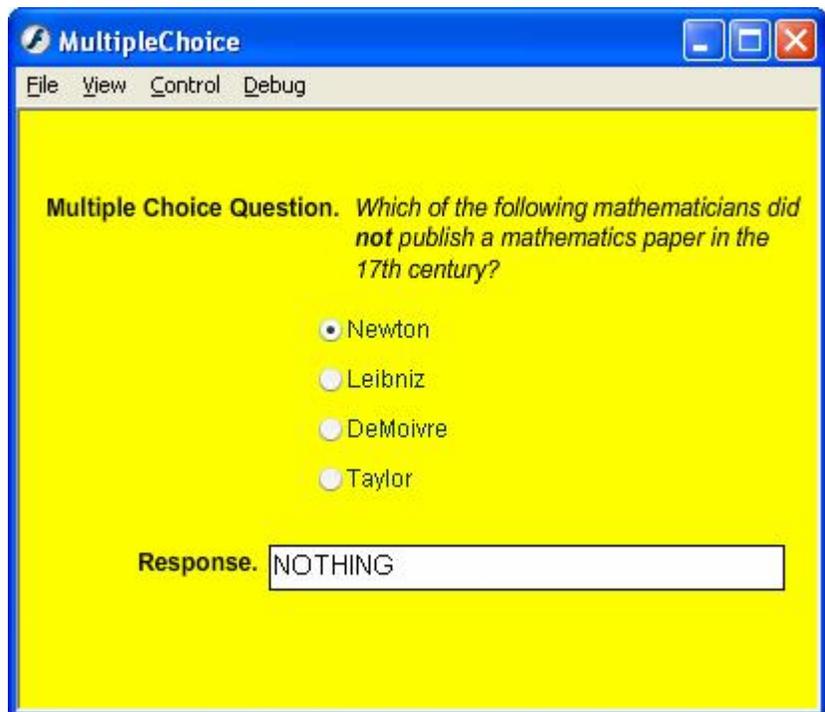
Repeat the process for the other three buttons, being sure to give them unique instance names, the same **groupName** as you gave the first button, identical **label** and **data** fields, and each with **selected** set to false. By using the same x-coordinate as the first button (225 in my case) and y-coordinates increasing by a fixed amount (like 25 px), you will get good alignment and spacing.

At this point, if you save and test your movie, you will see fully functioning radio buttons as shown at the right.

Step 3. Add a button

We now need a button that can be pressed to check the answer chosen by the user. Even though we are now experts at making buttons from scratch, for this application we will take a shortcut that you might find useful in the future.

Under the Windows menu item, select Common Libraries > Buttons. This repository contains a dizzying



array of predefined buttons that we can use. Some are quite fancy, but we will choose a plain style that does not clash with the overall plain look of our applet. Double click on the folder³ “buttons rect flat,” click on the “rectangle flat grey,” and drag it onto the stage in an appropriate location.

Single click on the button and give it an instance name like **buttonCheck**. Double click on the button to open the button timeline. Choose the layer labeled **text** (you might have to click on the picture of a padlock to unlock the **text** layer for editing) and change the text to read “CHECK” instead of “Enter”. If the text layer has more than one keyframe, you will have to make the change in each one.

If you return to the timeline for the main movie (Scene 1), you can save and test your movie to see that the button functions correctly relative to mouse events. Of course, nothing actually happens yet because we have not yet written the script.

Step 4. Add the script

The script for this movie is very simple. First click on Layer 1 of the timeline, and choose Insert > Timeline > Layer. Double click the layer labels to give the new layer the name “Scripts” and the old layer a descriptive name like “On Stage.” Click Frame 1 of the Scripts layer, and open the **Actions** panel in preparation of writing some code. Here is the full script for this movie. Once again, you do not need to type comments (following the //). If you do not wish to type *any* of this, you can open the file **FullScript.txt**, and copy and paste the script into your movie at this time.

```
messageBox.text = ""; // Clears the Response box
var correctAnswer:String = "Taylor"; // Sets correct answer for comparison
// with value from radio button group.

buttonCheck.onRelease = function():Void {
    var userAnswer:String = answerGroup.getValue(); // userAnswer is the value
                                                    // from the data field of
                                                    // the selected button in
                                                    // answerGroup.

    // Print a happy message if the userAnswer matches the correctAnswer.
    // Print a sad message otherwise.

    if (userAnswer == correctAnswer) {
        messageBox.text = "Yes, " + userAnswer + " is the correct answer!";
    }
    else {
        messageBox.text = "No, " + userAnswer + " is incorrect. Try again.";
    }
}
```

We make special note of the use of the “+” symbol for string concatenation. This allows the response to the user to be customized to his or her answer.

Save and test your movie, and you will see a full working version of the applet we described at the beginning of this tutorial. This applet is easy to customize for multiple choice questions of any type.

Additional resources

To see the source code for the applet we made for this tutorial, open the file **tutorial3 fla** in Flash.

³ If this your Buttons library does not have these exact items, shop around to find any flat button with accompanying text or just make your own as described in Tutorial 1.