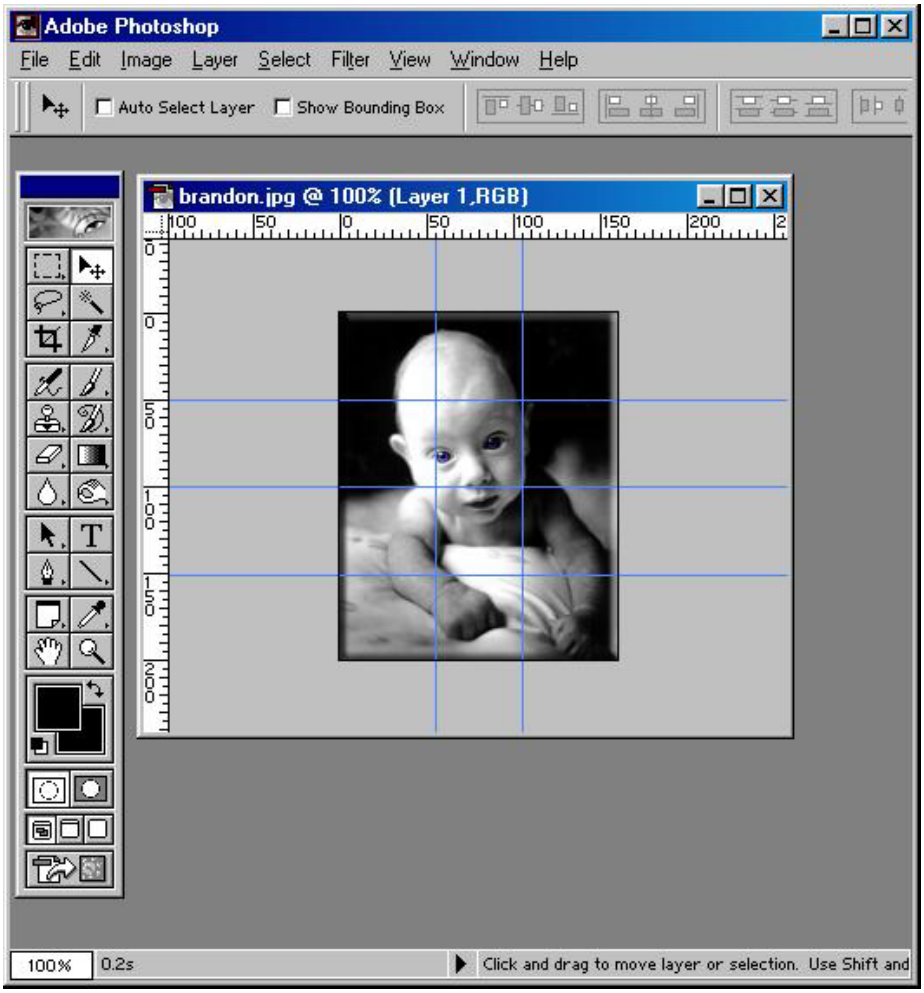
Creating Drag and Drop Objects that snap into Place – Make a Puzzle

FLASH MX Tutorial by R. Berdan Nov 9, 2002

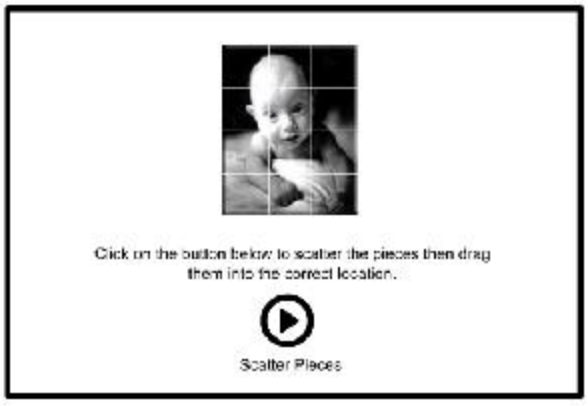
In order to make a puzzle where you can drag the pieces into place, you will first need to select a picture and slide it into pieces. You can use an image editor like Adobe photoshop, create some guides on top, then use the marquee tool to select each square copy it and paste it into a new file. Then save the graphics – I recommend you name them something like puz1.jpg, puz2.jpg etc. You may want to start with a really simple puzzle that contains only 4 or 9 pieces. The one below uses 12. Make sure you first set the image resolution of the large image using image size command to **72 dpi** first before you begin copying pieces. Note this tutorial assumes you are already familiar with basic menus in Flash.



Create a Folder and save each puzzle piece into the folder. Also save a copy of the completed image into the folder as well.

Start Flash MX – to create the puzzle we are going to use two scenes. The first scene will include a picture of the completed puzzle and a button below it that will take you to Scene 2, which contains the scattered puzzle pieces and as a guide a faded picture of the completed puzzle. When the user drags a piece of the puzzle and lets go – the puzzle will snap into place. In other words – simply clicking on a puzzle piece will make it jump to its correct position.

Below is a screen shot of scene 1. I included a black bounding box and the finished picture – you may wish to use a larger image. The text instructs the user to click on the button below in order to randomize the puzzle pieces. The arrow in a circle button was taken from the button library in Flash. Before you add a script to the button select Insert>Scene to add a second scene to your movie.



Right Click on the button, from the drop down menu select actions and add the following script in Scene 1.

```
on (release)
{
        gotoAndPlay("Scene 2", 1)   // go to scene 2 frame 1
}
```

In scene 1 select a frame or create a layer and call it actions and the add the following script to the first frame.

```
stop();
```

This script prevents the movie from jumping to scene 2 until the user clicks on the button.

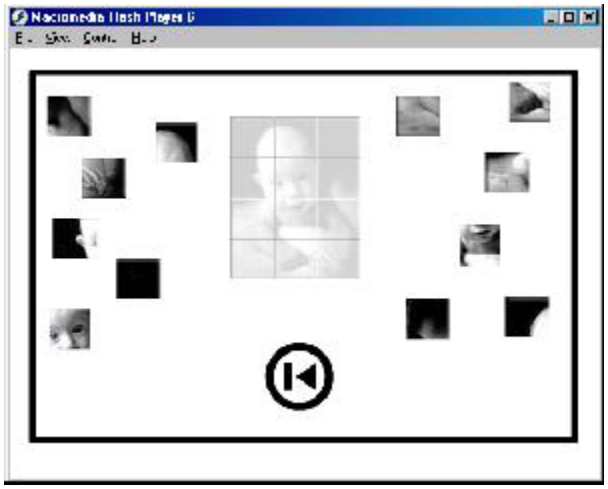
Now go to scene 2 and add a stop action to the first frame - so the movie does not try to loop back to the first scene.

Scene 2 –will contain the puzzle pieces the user can drag into place.

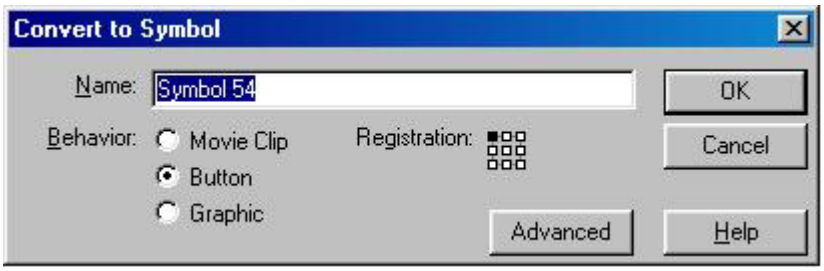
1) First import the completed puzzle image into layer one and position it near the center of the stage. Select the image, Insert>convert to symbol. Using the properties box modify alpha so the image is just visible and can act as a guide of where to place the puzzle pieces. You may want to lock this layer.

2) Insert a another layer in the movie – call it puzzle. With this layer selected File>import all the puzzle pieces onto the stage. I.e. puz1.jpg, puz2.jpg etc. You may want to remove them from the

stage, open your movie library and drag one on the stage at a time so you know which piece is which. See image below.



3) When you drag a puzzle piece onto the movie, Select it>Convert to symbol> Select button and it is very important that you make sure the registration for the button is in the top left corner. See image below.



If the Registration is not set like that above with the dark square in the top left, when you determine you puzzle x,y coordinates – the puzzle pieces will be offset up and to the left and will not come together over top of your guide image.

When you convert the first puzzle piece to a button, give it the instance name **puz1** in the properties box. Repeat for every piece and give it an instance name e.g. **puz2, puz3** etc.

4) Now drag each puzzle piece carefully over top of your guide image so that the pieces form a single image with no lines or spaces between them. This is what we want the finished puzzle to look like.

5) Starting with the first puzzle pieces, top left corner – record on a piece of paper the x,y coordinates of the puzzle pieces. The coordinates will appear in the properties box when you click on the puzzle piece. You should have a list that looks something like the table on the next page.

Puzzle Piece	x coordinate	y coordinate
puz1	206.3	63.4
puz2	246.8	63.4
puz3	284.8	63.4
puz4	206.3	101.3
puz5	247.1	101.3
puz6	284.9	101.3
puz7	206.6	139
puz8	246.8	139
puz9	284.6	139
puz10	206.1	176.9
puz11	246.8	176.9
puz12	284.8	176.9

Record the x,y coordinates for every piece in the puzzle when it is over top of the guide image. .

6) Now you will add an action script to each piece so that it will become draggable and when the user releases the piece it will snap into place in the coordinates. You will discover just clicking on a piece and releasing will cause the piece to snap into place.

```
on (press)
{
        startDrag("puz4");
}
on (release)
{
        stopDrag();
        setProperty("puz4", _x,"206.1")
        setProperty("puz4", _y,"101.3")
}
```

The script above is shown for puzzle piece 4 with the instance name **puz4 and causes the piece to snap to the x,y coordinates** provided which you include from the table of values you created for each piece when it was in the proper position.

setProperty( ) command works in Flash 4 and higher

setProperty("target",property,value/expression)

Description
Action; changes a property value of a movie clip or button as the movie plays.

Example
This statement sets the _alpha property of a movie clip named star to 30% when the button is clicked:

```
on(release) {
        setProperty("star", _alpha, "30");
}
```

7) You are almost ready to test the movie, however before you do drag each puzzle piece off the guide photo and place them some where on the stage like image on page 3. Now save your

movie, then select Control>Test movie. Click and drag the pieces to form the finished puzzle. If you want to randomize the puzzle you can add a button with the action script

on (release) {
gotoAndPlay("Scene 1", 1) }

this will restart the movie. If you want a challenge add a button and script that randomizes the puzzle pieces so the user can start again. Publish the movie.
To Scramle pieces randomly – stole this script from the Actions Script tutorial in Flash MX

Add a button to the stage scene 2

And add this script to the button

```
on (release)
{
        Scramble();   // call function Scramble
}
```

To the first actions frame in scene 2 add the following script

```
function Scramble()
{

        for (var i = 1; i<=12; i++) {
                with (this["puz"+i]) {
                        _x = random(425) + 25; // add 25 pixels from top
                        _y = random(300) + 25;  // add 25 pixels from left edge

                        //_rotation = Math.floor(Math.random()*4)*90;
                }
        }
}
```

The script uses a for loop to loop through the number of puzzle pieces, if you have more or less pieces set the value i<= then the number of pieces.

With (this[“nameofyourpuzzlepice” + i]) // takes each piece and loops through the entire series

_x = random(425) creates a random number between 0-424 (425 possibilities), then I added 25 so no x value is less than 25 pixels from the left edge. Not strictly required but I did this so all pieces stay within the black bounding box in my movie.

_y = random(300) _+ 25 creates a random y coordinate between 0-299 and adds 25.

Note in the FlashMX tutorial they use Math.floor(Math.random()) which is bit more complicated. Math.random generates random numbers between 0 and 1 i.e. 0.12654 and must be multiplied to get pixel value then rounded down (Math.floor()) does this.

I commented out the line _rotation - since line of code would rotate each piece randomly and there is no way to correct the rotation in the script we are using. I have left it in for instructional purposes only in case you wanted to rotate pieces randomly.

At this point anyone can solve the puzzle since you simply click on it and it pops into place. To make it even harder we will now define that a puzzle piece will only snap into place if it is dragged within the boundaries of the picture. See the code below. Add this script to every puzzle piece.

```
on (press)
{
        startDrag("puz5");
}
on (release)
{
        stopDrag();
        if (puz5._x >200 && puz5._x < 325 && puz5._y >50 && puz5._y <215)
        // if puzzle pieces is within the boundaries of the puzzle then snap!
        {
        setProperty("puz5", _x,"247.1")
        setProperty("puz5", _y,"101.3")
        }
}
```

To refine this one step further you could set the boundaries for every puzzle piece and only have the piece snap into place when it was within 5 or 10 pixels of its correct position. I know what your are thinking this is a lot of coding. You could do it once using a complicated loop and function – see the Flash Actions script tutorial – it is a bit complicated but it works.

As with all programs this one has a bug that my son discovered. If you are dragging a puzzle piece and click on the scramble button – one puzzle piece remains stuck to the mouse.

To fix this bug add stopDrag() to the Scramble() function like below.

```
function Scramble()
{

        (var i = 1; i<=12; i++)
        {
                with (this["puz"+i])
                {
                        _x = random(425) + 25; // add 25 pixels from top
                        _y = random(300) + 25;  // add 25 pixels from left edge
                        stopDrag();
                }
        }
}
```

Finally –We need to write one more function – one that detects when all the pieces are in place and sends a message to the dynamic text box in the lower right corner – saying “Congratulations”.

we need a conditional test to see if every piece is set to its correct x,y coordinate.

```
if ((puz1._x == 206.6 && puz1._y == 63.9) && (puz2._x == 63.9) )
{
message.text = “congratulations”;
}
```