

Interactive Pan Flash Movie tutorial by R. Berdan (Intermediate-Advanced)

This tutorial is based on the **Agile VR Flash VR movie Author: Brent Thomson** |
Website: <http://www.agilestudios.com> originally taken from **Flashkit.com** web site.

This tutorial will show how to build panoramic VR movies that move left or right as you drag your mouse over them. The main advantage of using Flash is that the movies will be much faster to load them Quick time movies and the dimensions of the movie are under your control as well as any interface elements such as buttons you want to add. I recommend you complete the first tutorial on a simple VR pan before trying this one.

First you need a panorama – you can download one from my web site or else where if you don't already have one. There are many software programs that will help you make pans e.g. Photoshop elements II, Live picture, Photoshop using layer masks etc.

Start with a pan image (e.g. rainforest_web_pan.jpg width 952 pixels, height 200 pixels)

1. Start Flash, Modify>Document – set 15 fps, width 400 height 200, leave background white.
2. Add 3 more layers and label them as follows from bottom to top:

Layer 1 Actions
Layer 2 Update
Layer 3 Hidden
Layer 4 Panorama

Inside each layer insert an additional keyframe so each layer has 2 keyframes.

3. Select keyframe 1 of Layer 1 (Actions layer), right click on the keyframe select>actions and add the action:

```
ifFrameLoaded (2) {  
    nextFrame();  
}
```

4. Select Keyframe 2 of Layer 1 and add the following action:

```
startDrag("/hidden", true);  
setProperty("/movie2", _x, Number(getProperty("/movie1", _x))+952);  
setProperty("/movie2", _y, getProperty("/movie1", _y));  
stop();
```

The +952 is the width of your panorama picture if yours is different you must change this value.

Hidden is an invisible button and movie1 and movie2 are movie clip instance names you will create.

Movie clips have properties whose values you can set and retrieve dynamically with ActionScript. Changing and reading these properties can alter the appearance and identity of a movie clip and is a key to creating interactivity. For example, the following script uses the `setProperty` action to set the transparency (alpha setting) of the `navigationBar` instance to 10:

```
setProperty("navigationBar", _alpha, 10);
```

```
getProperty(instancename , property)
```

Parameters

`instancename` The instance name of a movie clip for which the property is being retrieved.

`property` A property of a movie clip.

Function; returns the value of the specified `property` for the movie clip `instancename`.

ExampleThe following example retrieves the horizontal axis coordinate (`_x`) for the movie clip `myMovie` and assigns it to the variable `myMovieX`:
`myMovieX = getProperty(_root.myMovie, _x);`



This is what your layers palette should look like.

5. Place your cursor in the second keyframe of the Update Layer. Select Insert>new symbol> select movie clip give it name update.

You will be take to movie edit mode. Insert 2 keyframes. You are gong to add action script to these keyframes that updates and controls the pan movies.

In keyframe one add the action: `stop()`

In keyframe two add the action

```
tellTarget ("..") {
    cur_x1 = getProperty("/movie1", _x);
    cur_x2 = getProperty("/movie2", _x);
    if (Number(cur_x1)<Number(-952)) {
        setProperty("movie1", _x, Number(cur_x2)+952);
    }
    if (Number(cur_x2)<Number(-952)) {
        setProperty("movie2", _x, Number(cur_x1)+952);
    }
    if (Number(cur_x1)>952) {
        setProperty("movie1", _x, cur_x2-952);
    }
    if (Number(cur_x2)>952) {
        setProperty("movie2", _x, cur_x1-952);
    }
}
```

The value of 952 represents the width of your panoramic picture in pixels.

You are getting the x coordinate position of your movie clips and if they move setting the property or the new x coordinate position

6. Go to frame 3 and add the following action script.

```
tellTarget ("..") {  
    x = getProperty("/hidden", _x);  
    offset_x = (start_x-x)/8; // sets sensitivity of mouse drag 1 more sensitive  
    cur_x1 = getProperty("/movie1", _x);  
    cur_x2 = getProperty("/movie2", _x);  
    setProperty("/movie1", _x, Number(cur_x1)+Number(offset_x));  
    setProperty("/movie2", _x, Number(cur_x2)+Number(offset_x));  
}  
gotoAndPlay(2);
```

This says measure the new coordinate x position then go back to Frame 2 and loop. The code basically is how far you dragged the movie.

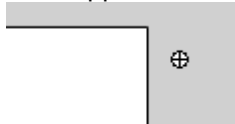
Explanation according to Brent:

There are two identical movie clips that contain the image. There is an invisible button that detects when the viewer presses the mouse button. There is an invisible movie clip being dragged that supplies the mouse's current position. When the user presses his/her mouse button, the invisible button sets two variables according to the x-value of the mouse at the instant the button was pressed (it gets these from the invisible dragging movie). It also tells another invisible movie clip to play. This second movie clip takes the initial mouse coordinates and compares them to the current position of the mouse. From this simple calculation, the movie clip comes up with an x-offset for the image movies. The movie clip then updates the position of both image movie clips by that values, checks if one of the movies is too far off of the screen, and then loops. Frame one of the update clip has a stop for when the user does not have his/her mouse button pressed. The second and third frame contain the actual code that makes the movie pan. Frame two checks to see if either image movie is far enough the screen that it can be situated on the opposite side. This makes it look like the image is one continuous picture

7. After adding the action script above return to the main movie stage (hit left point arrow beside Scene 1).

Drag a copy of the update movie clip onto the stage beside the main movie (i.e. off to the side) and give it the instance name **update**.

It will appear like a circle beside you main movie, a cross appears if it is selected.



8. Select the 2nd keyframe of layer Hidden, then select Insert>New Symbol>movie clip call it Hidden. In the movie clip scene draw a small circle, select it, Insert>Button. Right click on the button and add the following action script.

```

on (press) {
    tellTarget("../") {
        start_x = getProperty("/hidden", _x);
    }
    tellTarget("../update") {
        gotoAndPlay(2);
    }
}
on (release, releaseOutside) {
    tellTarget("../update") {
        gotoAndStop(1);
    }
}
}

```

Return to the main scene and drag an instance of the invisible button onto the stage but outside the main movie background li.e. beside the Update movie clip) exact position not critical.

Give the movieclip the instance name **hidden** in the properties tool box.

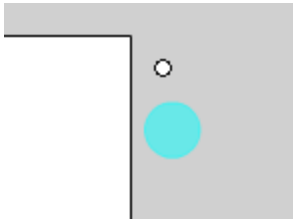


Diagram shows the update movie clip and the hidden button (blue) on screen off the main movie background (white area). Make sure you have keyframe 2 of the hidden layer selected when you drag the hidden movie clip on to the stage.

Note: to view the script in the invisible button or movie clip you need to double click on the main symbols in the library movie.

9. Now we are ready to import our panoramic photo. Select Insert>New Symbol>Movie clip then select File>Import and select your panoramic movie. Position the movie so that its upper left corner is set to 0.0 in the movieclip scene.



Return to the main movie stage.

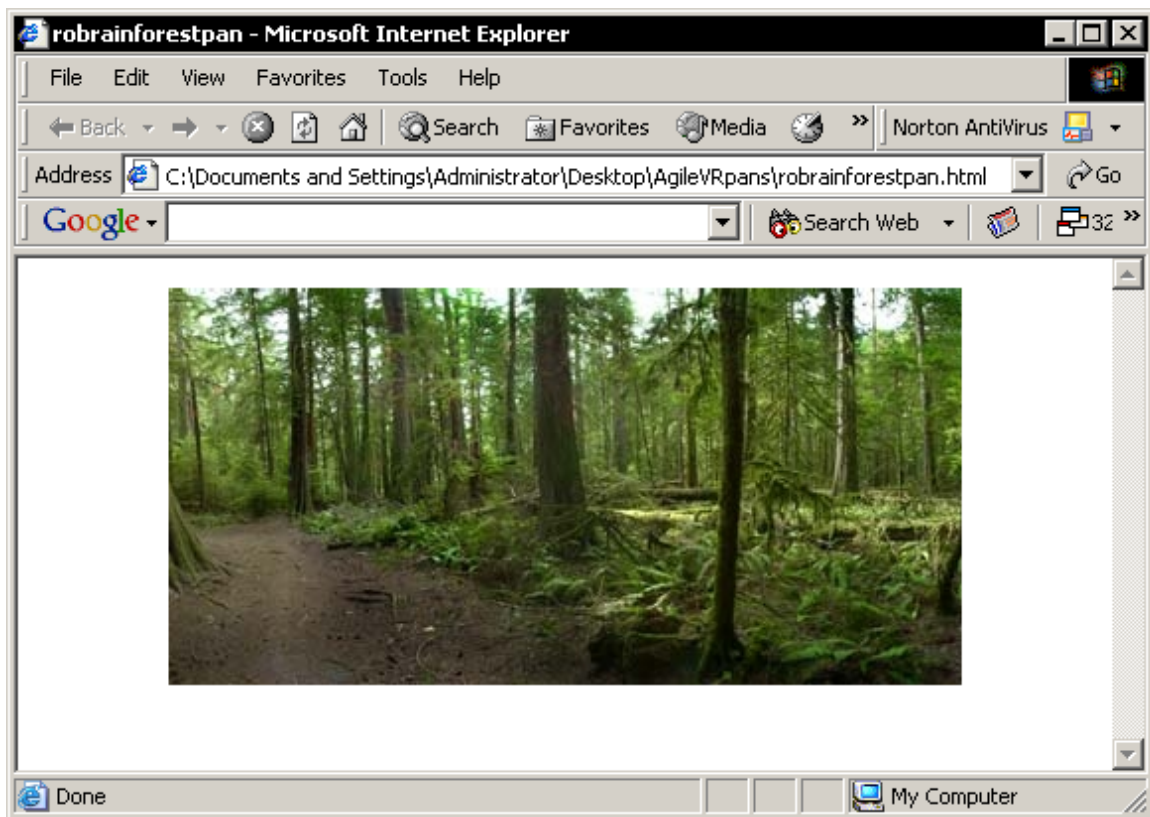
10. Select the 2nd Keyframe of the panorama layer, then drag and instance of your panorma movie clip onto the stage so that it is position at x,y 0,0 i.e. the top left corner matches exactly with the top left corner of your main movie. Check the coordinates in the properties box to be sure. Give this movie the instance name **movie1**

Then drag a second instance of the panoramic movie onto the stage and give it the instance name **movie2** Where you put the second movie doesn't matter because you set the property of movie1 and movie 2 to be relative to each other in the 2nd keyframe of the actions layer.

```
setProperty("/movie2", _x, Number(getProperty("/movie1", _x))+952);  
setProperty("/movie2", _y, getProperty("/movie1", _y));
```

If you have done everything correctly – select Control Test movie and move your mouse over the pan – it should move right and left. If not check you did not leave off an instance name on the hidden or update movie clips. Also check them for type case.

You are ready to save your movie and then publish it. You will need to center the movie by adding <center></center> in the HTML – this a bug that Macromedia seemed never when you publish a movie.



If you move your mouse over the movie it pans left or right. This a basic vanilla movie you can add text, a fancy border or anything else you want. A % preloader bar is nice especially for larger VR movies. This movie is 132 KB in size has better image quality than a java applet, and loads a lot faster.

Adding interactive buttons

Some useful buttons that you might like to add to your VR movie clip are: zoom in, zoom out, and reset size back to the original size.

Below I will show you how to add buttons and some scripts – however you should note that the zoom feature is limited in value and when you zoom into the picture the pan no longer has the correct stitch coordinates. Furthermore the zoom occurs from the upper left corner (x=0, y=0) of the movie. In this regard Quicktime movies are superior until someone comes up with a code fix.

Insert a new layer on your main movie and call it buttons. Then add a second keyframe. Select this keyframe and drag 3 buttons onto your main movie. (You are welcome to make your own).

Button 1. Zoom in

Button 2. Zoom out

Button 3. Reset – take movie back to original size

Window>Common Libraries>Buttons



Here is a series of buttons I modified using the arrow circle buttons.

Select the first (zoom in button with + symbol), right click on the button, Select>actions and add the following code:

```
on (press) {  
    _root.movie1._xscale += 5;  
    _root.movie1._yscale += 5;  
    _root.movie2._xscale += 5;  
    _root.movie2._yscale += 5;  
}
```

Basically this script says to increase the x,y size of the both movie clips 1 and 2 by 5 each time you press the button. It works but its not a smooth zoom.

```

on (press) {

    _root.movie_yscale= 100;
    _root.movie_xscale= 100;
    _root.movie1._xscale -= 5;
    _root.movie1._yscale -= 5;
    _root.movie2._xscale -= 5;
    _root.movie2._yscale = _root.movie2._yscale - 5;

if (_root.movie1._yscale <= 100 && _root.movie1._xscale <=100 && _root.movie2._yscale <= 100
&& _root.movie2._xscale <=100)
{
    _root.movie1._yscale = 100;
    _root.movie1._xscale = 100;
    _root.movie2._yscale = 100;
    _root.movie2._xscale = 100;
    // don't permit the movie to zoom below 100%
}
}

```

Notice:

```

_root.movie2._xscale -= 5;
_root.movie2._yscale = _root.movie2._yscale - 5;

```

these two lines are different ways of saying the same thing take the movie size and subtract 5 from its xy coordinate. -= or += is programmer short form for the longer version on the second line above. In bold type.

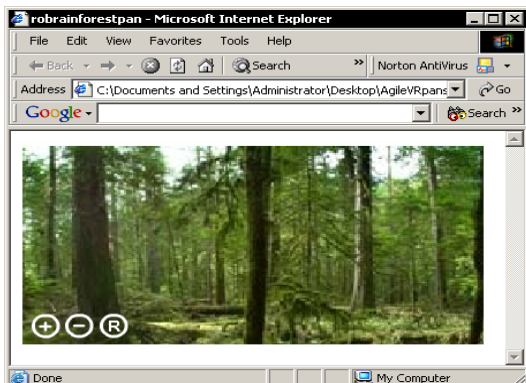
Fore the reset button add the following action script.

```

on (press) {
// reset movie to initial size
_root.movie1._yscale = 100;
_root.movie1._xscale = 100;
_root.movie2._yscale = 100;
_root.movie2._xscale = 100;
}

```

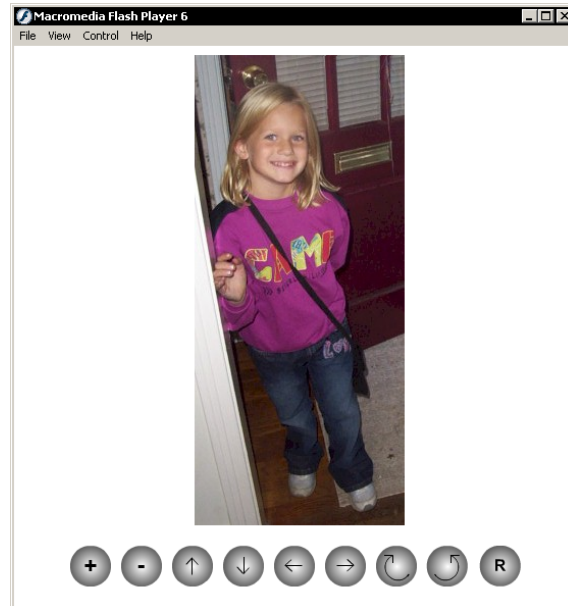
The buttons above only zoom in or out when you click on them so you have to click several times. To make a continuous zoom button we need to create another invisible movie clip that loops and includes a zoom function. The problem I ran into is that when you zoom the size of the movie changes so they no longer register properly at their ends. To fix this we would have to convert the movie size into a variable that changes with amount of zoom.



Note button changed to include R for reset size.

Adding buttons that zoom smoothly. The code is taken from another tutorial I found on Flashkit.com. Image control Pad by Sam Bous. (file called scale_move_image2 fla)

There is no tutorial but you can download the the .fla file and it includes a variety of buttons such as rotate, move and zoom. For a better understanding of how it works see simple flash pan tutorial.



I am only going to cover the smooth zoom in and out buttons. The move buttons work great but additional code must be added to restrict the movement of your panoramas to the viewing area.

1. First make or drag two button onto the stage into buttons layer keyframe 2. (I used the arrow buttons in a circle one pointing up and the other down – had to rotate one button so it pointed town)
2. Add the following action script to each button. First the up (zoom in button) right click on the button and add the following action script.

```
on (press) {  
    _root.StoredActions.gotoAndPlay("increase");  
}  
on (release, releaseOutside) {  
    _root.StoredActions.gotoAndStop(1);  
}
```

// StoredActions is an invisible movie clip we will create shortly

3. Add the following script to the zoom out (decrease zoom) button.

```
on (press) {  
    _root.StoredActions.gotoAndPlay("decrease");  
}  
on (release, releaseOutside) {  
    _root.StoredActions.gotoAndStop(1);  
}
```

- Next we need to create a movie clip that we will store various actions in. The movie clip plays continuously so that when you click on the zoom button the script is continuously executed until you release the button – providing a smooth zoom in and out. You will give the movie clip the instance name **StoredActions** which you can see in the button code above. Select >Insert>New Symbol>Movie Clip call it ActionClip. Your screen will open in the actionclip Scene. Select keyframe 1 and add the action `stop()`;
- In Layer 1 of your action clip movie add a second keyframe and in the properties box give it the frame name “increase” this is where the `gotoAndPlay(“increase”)` statement above is calling this frame to play. Right click on the 2nd keyframe and add the following action script.

```
_root.movie1._xscale += 5;
_root.movie1._yscale += 5;
_root.movie2._xscale += 5;
_root.movie2._yscale += 5;
```

This code is the same as our code for the initial buttons.

```
_root.movie1._xscale = _root.movie1._xscale + 10;
```

the line above is the same as

```
_root.movie1._xscale += 5;
```

the latter method is just a short cut method for writing the longer version above.

If you were controlling a single movie you would only need two lines of code to increase the x and y scale of the object. However our pan has two movies - instance names **movie1** and **movie2** - so we need to double up the code so both movies are scaled. The number 5 controls the amount of zoom, higher numbers will zoom faster and lower numbers zoom slower or less amounts. You can think of as 5 increase picture size 5% and a value of 10 would increase the picture 10%.

- Create a third keyframe and add the following action script.

```
gotoAndPlay(2);
```

This is your loop when you press the button, it zooms in, movie goes to the next frame (keyframe 3) which then sends it back to keyframe 2 and executes the action again until stop pressing on the button. This creates a smooth zoom effect.

- Create another keyframe at 4, give it the instance name **decrease** and then add the following action script.

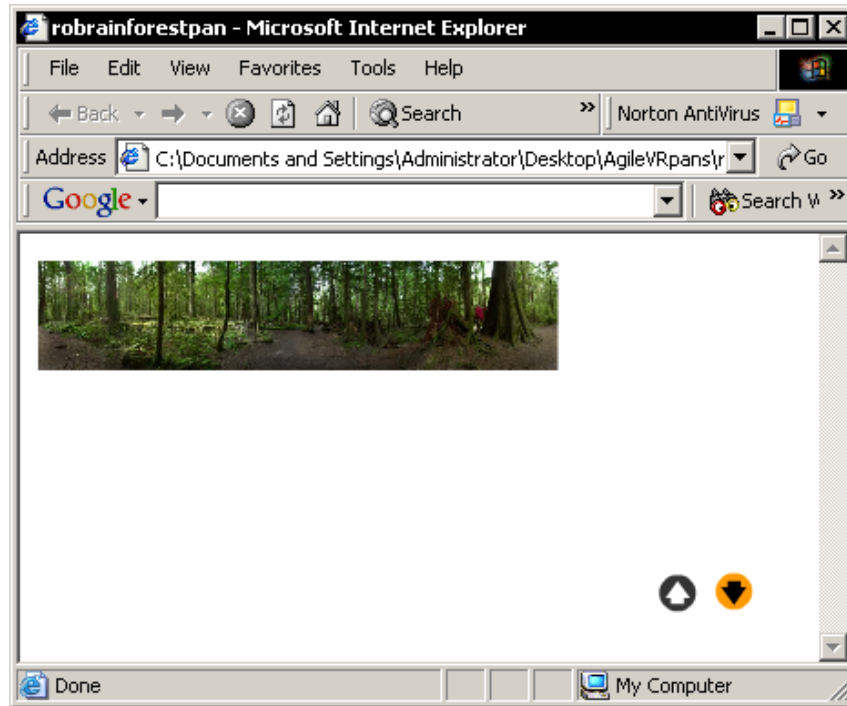
```
_root.movie1._xscale -= 5;
_root.movie1._yscale -= 5;
_root.movie2._xscale -= 5;
_root.movie2._yscale -= 5;
```

- Now add a 5th keyframe and add the action script:

```
GotoAndPlay(4);
```

8. Return to the main movie and then open your library. Drag the Action clip onto the stage – it does not have to be on the visible movie area. Give the instance name **StoredActions**

Save your movie, publish and then test your buttons. You should see your panoramic movie zoom in or out smoothly depending on which button you select. However there is one problem the zoom out feature has no limits so your movie could disappear if you keep zooming out.



Zooming out the movie continues to shrink. So we have to add limits.

To set the limits on how small we want the user to zoom out to we need to add the following code to the ActionClip movie frame 4.

```
_root.movie1._xscale -= 5;  
_root.movie1._yscale -= 5;  
_root.movie2._xscale -= 5;  
_root.movie2._yscale -= 5;
```

```
if (_root.movie1._yscale <= 100 && _root.movie1._xscale <=100 && _root.movie2._yscale  
<= 100 && _root.movie2._xscale <=100)  
{  
  _root.movie1._yscale = 100;  
  _root.movie1._xscale = 100;  
  _root.movie2._yscale = 100;  
  _root.movie2._xscale = 100;  
  // don't permit the movie to zoom below 100%  
}
```

The bold code above prevents the movie from zooming out below 100%. Your movie is ready to go. There are many aesthetic things you can do to make the movie clip more attractive – this tutorial only focused on the coding required to make the movies. Hope you enjoy. RB

Comments – this movie still has some things that need to be improved upon, I welcome your input or I will tackle them when I have more time.

1. There is no ability to move the movie up or down – this is easy to implement with buttons however one would also have to restrict the movement to the boundaries of the movie clip or your movie could move off the screen.

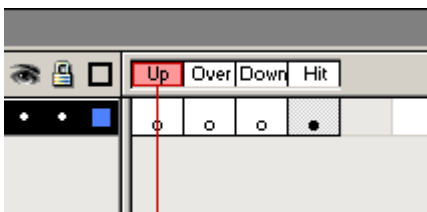
```
_root.movie1._y = _root.movie1._y - 10;  
if (_root.movie1._y <= -(_root.movie1._height/2)) {  
    setProperty("_root.movie1", _y, 550+(_root.controlme._height/2));  
}
```

if you add another button “UP” and add the code to a new keyframe in the ActonClip movie this will move the pan up. However this will continue to move the pan up until its off the viewing screen! Additional code needs to be added to restrict the up movement to the frame and tie the movement to the overall zoom amount. Similarly the code below moves the pan down.

```
_root.controlme._y = _root.controlme._y + 10;  
if (_root.controlme._y >= 550+(_root.controlme._height/2)) {  
    setProperty("_root.controlme", _y, -(_root.controlme._height/2));  
}
```

2. There is no feature to drag the zoomed in pan movie so the person viewing the movie can move the zoomed in pan up and down like you can using Quicktime movies.
3. If you pan photo is a high resolution image you could publish the movie so that it stretches (i.e. as percentage height and width in the Publish settings) this would make your pan expand to fit the computer window. Try it you can make full screen movies.
4. You can add hot spots (hyperlinks) to the movie clip so that when you click on the movie it loads another panorama or jumps to another web site.

To do this double click on movie clip one. Add a new layer call it invisible button. Draw a rectangle on top of the spot you want to turn into a hyperlink. Select the rectangle>Insert New symbol>button. You will make an invisible button. Double click on the button you made to go into button editing mode:

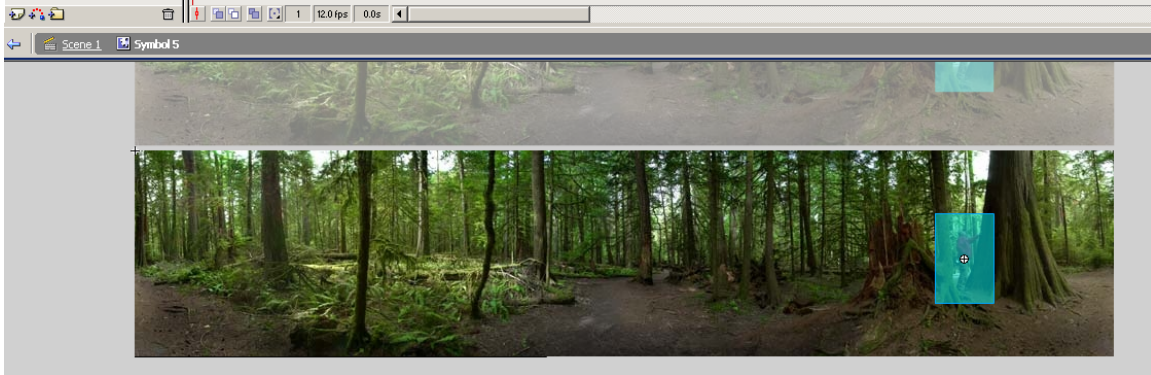


Add keyframes to Over, Down, then select Up, and delete the graphic, repeat with Over and Down. This is so your button is invisible. Leave the Hit state as this is the area that will respond to the on press event. Return to the movieclip by select the left arrow beside scene 1 (under layers) but do not return to the main movie. Select the invisible button and add the following action.

```

on (press) {
    getURL(http://www.scienceandart.org", "_blank");
}

```



My pan movies showing the location of the “hotspot” where I created an invisible button. You could also add loadMovie(“pano1.swf”) to load a new movie in its place (with same dimensions).

This will link to my web site and do so in a new window. Modify as you please. You could also load a new panorama here as well to produce a walk through illusioning for example by clicing on a door you move into another room. After adding the script return to the main movie, Save, publish and test your movie. If you click on my father you will be linked to my web site in my demo movie.

5. Zooming in modifies the overall dimension of the two movie clips and they no longer form a seamless pan– to fix this one would have to set the height and width parameters as variables and tie the dimensions of these variables to the zoom in and out values. I will do this when I get some time.
6. The code uses Tell Target this could be updated to use with.
7. You could modify the code so that the movie is rotating when it first loads. Tricky because the movie stops when you release the mouse and mouse down triggers the movie.

Flash Panoramic Movie



To build your own panoramic movies using Flash see tutorial.