

Flash MX Radio Tutorial program by Juan Rivera Interface by Robert Berdan June 2002

Introduction

This tutorial is designed to teach you how to create a small radio in Macromedia Flash MX. The radio works by having a handle that you can drag in order to “tune” it. When you move it between stations, you get static, static mixed with music, and static-free music if you adjust it. The following steps will teach you how to make a clone of the one that I have provided.

Part 1: Building the Interface

1. Open radio.fla as a library we will be using resources from there.
2. Resize your movie canvas to 400x200
3. Create a new layer and drag the symbol “FM Dial Build” into it. Use the info palette to precisely position the center of it at coordinates x: 154 and y: 100. This is to make sure that it’s properly centered (no portions of pixels) and that it works with the actionscript for the draggable handle (this will be explained later).
4. Create a new layer on top of the one that you just created and drag the symbol “Dial” into it. Use the info palette to position it at coordinates x: 146 and y: 104. Again, this is for the purposes of not breaking the actionscript.
5. Your interface is now ready to be programmed.

Part 2: Importing the Sounds

I have provided the music files that we will be using for this tutorial in a separate directory called “sounds”. You will now have to import them into your flash movie by following some simple steps.

1. Import the file “sounds/A_Cleft-Lee_I_G_373.wav” into your flash movie. After it is imported, right click it in the library and click “Linkage...” in the context menu that comes up. Under “Identifier” type “sound1”. Make sure that “Export for Actionscript” and “Export in first frame” are checked. This is important. What we just did is called “exporting an object”. Basically, it allows flash to make reference to an object that hasn’t been placed in a movie. I will be referring to these instructions again so please keep this in mind.
2. Import the file “sounds/audiolab-Audiolab-675.wav” and follow the same instructions as above to export it as “sound2”.
3. Import the file “sounds/FBI01-Kan_Muft-669.wav” and export it as “sound3”
4. Import the file “sounds/Matri-calpomat-4002.wav” and export it as “sound4”
5. Import the file “sounds/Violin-Bass-OSAMA.wav” and export it as “sound5”
6. Import the file “sounds/static.wav” and export it as “soundStatic”

Part 3: Setting up the sounds

Now that we have imported the sounds, we have to actually set them up and get them to start playing. In order to do this, we have to use actionscript to manipulate sound objects that we will create in this section. We *MUST* use sound objects and not just drag them into the movie because the use of objects allows us to keep control of the sounds throughout the movie.

Another problem with flash is that in order to create the sound objects and get them to work properly, we have to create empty movie clips and link the sound objects to those movie clips. The reason we have to do this is because Flash will use a sound object to control every single playing sound in the entire movie and thus will render our radio unusable because sounds cannot be individually controlled. Therefore, the first step in this section would be to create a movie clip, leave it empty, and assign it a different instance name for each sound. This is not the most intuitive way to do it, but it's the only way that works.

1. Create a blank movie clip in your library. Call it "EmptyMov".
2. Drag 6 copies of your new symbol "EmptyMov" outside the working canvas. The empty movies will appear as large white dots when dragged outside. This is to assist you in clicking and manipulating the movie clips. Do not be alarmed.
3. Click on one of the newly created white dots and give it an instance name of "movA". These are being created for the purpose of linking control of the sound to the clips so that we can control sounds individually. Assign the remaining clips instance names of "movB", "movC", "movD", "movE", and "movStatic" respectively. Once you have done this, there is no reason to touch the movie clips again.
4. Here is where the tricky part begins. You will need to actually create sound objects out of the movie clips and the only way to do this is by using actionscript. Start off by creating a new layer in your timeline and naming it "Sound Init"
5. Copy the following actionscript into the layer "Sound Init":

```
// Here we Initialize the sound (required for the engine to work properly)
_root.movA.a = new Sound(movA);
_root.movB.b = new Sound(movB);
_root.movC.c = new Sound(movC);
_root.movD.d = new Sound(movD);
_root.movE.e = new Sound(movE);
_root.movStatic.a = new Sound(movStatic);

// Here we attach sounds to our sound objects
_root.movA.a.attachSound("sound1");
_root.movB.b.attachSound("sound2");
_root.movC.c.attachSound("sound3");
_root.movD.d.attachSound("sound4");
_root.movE.e.attachSound("sound5");
_root.movStatic.a.attachSound("soundStatic");

// Here we set all of our sound objects to zero volume, large loop
_root.movA.a.setVolume(0);
_root.movB.b.setVolume(0);
```

```

_root.movC.c.setVolume(0);
_root.movD.d.setVolume(0);
_root.movE.e.setVolume(0);
_root.movStatic.a.setVolume(0);

// start all sounds
_root.movA.a.start(0,999);
_root.movB.b.start(0,999);
_root.movC.c.start(0,999);
_root.movD.d.start(0,999);
_root.movE.e.start(0,999);
_root.movStatic.a.start(0,999);

```

6. The above code initializes the sound and creates sound objects, attaches sounds to our newly created sound objects, sets the initial volume of all of our sounds to 0 (so we don't hear them) and starts all the sounds playing.

Part 4: Setting up the slider

After the sound is all set up and initialized, we need to set up the slider" so that we can "tune in" to the different radio stations. This is best accomplished by using a draggable layer so that we can keep track of the coordinates and use them as a reference point. In this section of the tutorial we will use actionscript to make the "handle" draggable and to report back a number from 1-100 (which we will later use in the sound engine)

1. Double click on the handle that we imported into our project earlier. This handle is actually a button within a movie clip. The button is so that you can attach an interaction to it and the movie clip is so that you can keep track of it's position on the screen. Once you have double clicked it and are editing the "Dial" symbol, click on the handle again and insert the following actionscript:

```

on (press) {
    startDrag(this, false, 146, getProperty("",_y), 246, getProperty("",_y));
}
on (release, dragOut) {
    stopDrag();
}

```

The startDrag() listed above is particularly important to the function of our movie. Basically, it limits the range of the slider to 100 pixels (i.e. it starts at x: 146 and may continue up to y: 246). This number is later used for our main "engine"

Part 5: Setting up the "engine"

Due to the way Flash MX works (i.e. not the way it *should*), we sometimes have to do strange things in order to get things to work the way we expect them to. This is another one of those not-so-intuitive steps that we have to do just to make the radio effect work properly.

First, we started off by creating the sound objects all in their own little movie clips. This time, we need to create one additional movie clip that will be controlled by actionscript that will control the other sound movie clips via actionscript calls. It's not as difficult as it sounds...

1. Create a new movie clip in your library and call it "MovieCon". This clip doesn't have to have anything in it. In fact, it should remain completely empty.

2. Drag a copy of the newly created "MovieCon" outside the canvas. It will turn into a large white dot to aid with user interaction. Click it and give it an instance name of "controller"
3. Double click the movie you just dragged onto the canvas and select the first frame. This movie will be used to control the other sound movies via actionscript. There is really no other way around this.

Insert the following action script in Frame 1:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(65);
```

Above: Sets volume of music to 0, sets volume of static to 65

Here comes the tedious part, you need to create a new blank keyframe and insert the following into Frame 2:

```
_root.movA.a.setVolume(70);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(50);
```

Above: Sets volume of sound1 to 70 and volume of static to 50 (gives the effect of the song not being tuned in correctly because it plays both the song and the static)

Continue inserting keyframes and pasting in actionscript until you reach the end of this section:

Frame 3:

```
_root.movA.a.setVolume(100);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(0);
```

Above: Plays the song without any static, gives the effect of the song being completely in tune.

Frame 4:

```
_root.movA.a.setVolume(70);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(50);
```

Frame 5:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(65);
```

Frame 6:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(50);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(50);
```

Frame 7:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(100);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(0);
```

Frame 8:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(70);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(50);
```

Frame 9:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(65);
```

Frame 10:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(70);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(50);
```

Frame 11:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(100);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(0);
```

Frame 12:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(70);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(50);
```

Frame 13:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(65);
```

Frame 14:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(70);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(50);
```

Frame 15:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(100);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(0);
```

Frame 16:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(70);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(50);
```

Frame 17:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(65);
```

Frame 18:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(70);  
_root.movStatic.a.setVolume(50);
```

Frame 19:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(100);  
_root.movStatic.a.setVolume(0);
```

Frame 20:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(70);  
_root.movStatic.a.setVolume(50);
```

Frame 21:

```
_root.movA.a.setVolume(0);  
_root.movB.b.setVolume(0);  
_root.movC.c.setVolume(0);  
_root.movD.d.setVolume(0);  
_root.movE.e.setVolume(0);  
_root.movStatic.a.setVolume(65);
```

The tedious part is now done. It's important to understand what is going on in the code above.

Basically, in order to get the effect of semi-realistic radio, we have to have pure static (when nothing is in tune), static+music (when the music is *almost* in tune) and music with no static (when the song is in tune). All the above does is control movies so that this effect is achieved. The pattern goes as follows:

- a. Pure Static, nothing in tune
- b. Song1 + Static, almost in tune
- c. Song1 no static, in tune
- d. Song1 + Static, almost in tune
- e. Pure Static, nothing in tune
- f. Song2 + Static, almost in tune
- g. Song2 no static, in tune
- h. Song2 + Static, almost in tune
- i. Etc...

Once the above steps are completed (you can't skip a step or else it won't work) the main engine is basically ready... Only a little more actionscript and the radio will be functional.

Part 6: Finalizing the radio and driving the "engine"

Now that the main engine is ready, your radio is almost finished. All that's left is a little actionscript that will control the engine (i.e. tell it when to play specific sounds). We are going to write some actionscript that will report back the position of the slider (from a scale of 1-100) and some code that will control the engine (which will in turn control the sound).

1. Click on the handle that we created earlier (the base of the slider, make sure not to double click) and insert the following actionscript.:

```
onClipEvent (enterFrame) {
    _root.pos = this._x;
    curPos = this._x-146;

    //Engine Code
    if((curPos >= 0) and (curPos <=2)) {
        tellTarget(_root.controller) { gotoAndStop(1); }
    }
    if((curPos >= 3) and (curPos <=6)) {
        tellTarget(_root.controller) { gotoAndStop(2); }
    }
    if((curPos >= 7) and (curPos <=13)) {
        tellTarget(_root.controller) { gotoAndStop(3); }
    }
    if((curPos >= 14) and (curPos <=17)) {
        tellTarget(_root.controller) { gotoAndStop(4); }
    }
    if((curPos >= 18) and (curPos <=22)) {
        tellTarget(_root.controller) { gotoAndStop(5); }
    }
    if((curPos >= 23) and (curPos <=26)) {
        tellTarget(_root.controller) { gotoAndStop(6); }
    }
    if((curPos >= 27) and (curPos <=33)) {
```

```

        tellTarget(_root.controller) { gotoAndStop(7); }
    }
    if((curPos >= 34) and (curPos <=37)) {
        tellTarget(_root.controller) { gotoAndStop(8); }
    }
    if((curPos >= 38) and (curPos <=42)) {
        tellTarget(_root.controller) { gotoAndStop(9); }
    }
    if((curPos >= 43) and (curPos <=46)) {
        tellTarget(_root.controller) { gotoAndStop(10); }
    }
    if((curPos >= 47) and (curPos <=53)) {
        tellTarget(_root.controller) { gotoAndStop(11); }
    }
    if((curPos >= 54) and (curPos <=57)) {
        tellTarget(_root.controller) { gotoAndStop(12); }
    }
    if((curPos >= 58) and (curPos <=62)) {
        tellTarget(_root.controller) { gotoAndStop(13); }
    }
    if((curPos >= 63) and (curPos <=66)) {
        tellTarget(_root.controller) { gotoAndStop(14); }
    }
    if((curPos >= 67) and (curPos <=73)) {
        tellTarget(_root.controller) { gotoAndStop(15); }
    }
    if((curPos >= 74) and (curPos <=77)) {
        tellTarget(_root.controller) { gotoAndStop(16); }
    }
    if((curPos >= 78) and (curPos <=82)) {
        tellTarget(_root.controller) { gotoAndStop(17); }
    }
    if((curPos >= 83) and (curPos <=86)) {
        tellTarget(_root.controller) { gotoAndStop(18); }
    }
    if((curPos >= 87) and (curPos <=93)) {
        tellTarget(_root.controller) { gotoAndStop(19); }
    }
    if((curPos >= 94) and (curPos <=97)) {
        tellTarget(_root.controller) { gotoAndStop(20); }
    }
    if((curPos >= 98) and (curPos <=100)) {
        tellTarget(_root.controller) { gotoAndStop(21); }
    }
}

```

I'll take a moment to explain what the above code does.

The “onClipEvent(enterFrame)” statement tells flash to interpret all the code beneath it every time your mouse enters the area that the handle occupies. For us, this makes our radio tune in whenever we drag the slider around.

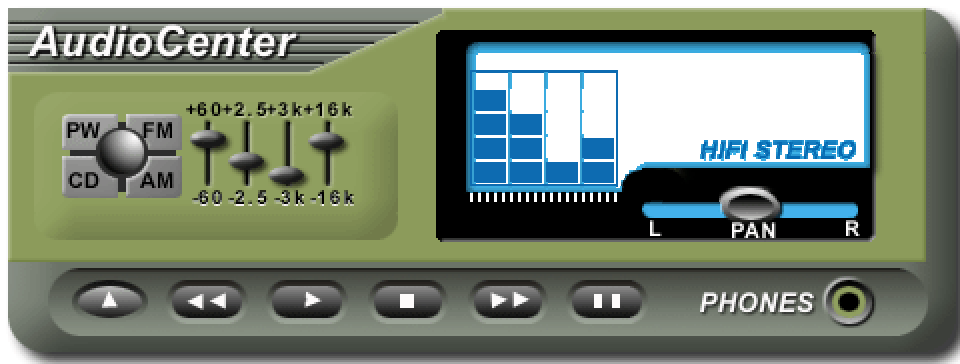
The “curPos = this._x-146” statement gets us a number from 1-100 that we will use to control the rest of the radio. this._x is a property of the movie that returns the position of the x-coordinate. You'll notice that this._x-146 will return the current coordinate of x minus 100 (this will make the range of x from 146-246 minus 100. Remember when I

mentioned earlier on in the tutorial that it's important to align the slider at x-coordinate 146?). This will effectively give us the number from 1-100 that we are about to use.

```
if((curPos >= 0) and (curPos <=2)) {  
    tellTarget(_root.controller) { gotoAndStop(1); }  
}
```

The above code and all other statements like it will control the engine. Basically, all that it does is tells the engine (that we created in Step 5) to go to specific frames, which in turn controls the various sound objects that we have created.

Your Radio is now complete.



You can design your own interface for the radio – if you make your flash movie larger and move the music bar and the handle you will need to change the action script to account for the new x coordinates.

Specifically if you move the radio station bar to x = 512 and the handle to 504 (8 pixels to the left) – edit the handle script as follows

Original script below

```
on (press) {  
    startDrag(this, false, 146, getProperty("",_y), 246, getProperty("",_y));  
}  
on (release, dragOut) {  
    stopDrag();  
}
```

modified script

```
on (press) {  
    startDrag(this, false, 504, getProperty("",_y), 604, getProperty("",_y));  
}  
on (release, dragOut) {  
    stopDrag();  
}
```