

Macromedia Flash MX - Level II

Instructor: Gabino Travassos, Mote Interactive

XML

To import XML files, Flash requires you to create an XML object. That is the first line below. The third line creates a little protocol for Flash to follow when the XML file has loaded. When loaded, it runs a function. In the next lines we create that function (called handleLoad) that will run when the XML has loaded. Even though the XML has loaded, in frame 3 we run a check to see whether the XML file has any content yet.

in frame 1:

```
cdsXML = new XML();
cdsXML.ignoreWhite = true;
cdsXML.onLoad=handleLoad;
```

```
function handleLoad(status){
  if(status){ gotoAndPlay("loaded"); }
}
```

```
cdsXML.load("extList2.xml");
```

in frame 3:

```
countNodes = cdsXML.childNodes.length;
```

```
if(countNodes==null){ gotoAndPlay(2); }
// loop until loaded
```

The last line of frame 1 loads the XML file. Notice the onLoad property on line 3, it calls a function that checks the "status" of the XML file (to check whether it was successfully parsed/imported).

In frame 3 count the nodes of the XML file. If there are no nodes, then the file hasn't loaded yet, so keep looping.

This is an excerpt of the XML file used above.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<cdlist>
  <item jpeg="01.jpg" artist="Amy Campbell"
    title="Architecture" />
  <item jpeg="02.jpg" artist="David Rovics"
    title="Song for Mahmud" />
</cdlist>
```

To get the name of the artists, I would:

```
getcdsNav.childNodes[0].attributes.artist; // Amy Campbell
or
getcdsNav.childNodes[1].attributes['artist']; // David Rovics
```

DUPLICATEMOVIECLIP()

When using duplicateMovieClip(), you're most likely cloning a new movieclip from an existing source. In the example file, the number of CD covers is variable, so we'll create the new movieclips using a loop.

We need the number of nodes for each CD, and we used this line in the previous column:

```
countNodes = getcdsNAV.childNodes.length;
```

```
for(i=0; i<countNodes; i++){
  newName="cd"+i; newDepth=20+i;
  duplicateMovieClip("placeholder",newName,
    newDepth);
  cdImport = getcdsNav.childNodes[i].attributes.jpeg;
  loadMovie(cdImport, newName);
  // move left to right
  if(i%3==0){ moveX=100; }
  if(i%3==1){ moveX=225; }
  if(i%3==2){ moveX=350; }
  eval(newName)._x= moveX;
  // move down
  if(i%3==0){ moveY+=125; }
  eval(newName)._y=moveY;
}
placeholder._visible=0;
```

placeholder is the name of the movieclip we're going to duplicate. When we're finished duplicating, we hide the original: placeholder._visible=0;

On the first line after the for loop we need to create new names and depths for the new movieclips. Every movieclip on the stage needs a unique name so our actionsript knows what to target. Every duplicated movieclip also needs a unique depth. We'll use the loop variable "i" to create unique names and depths.

On the cdImport line we use the xml.attributes properties to get the jpeg for the node "i". We assign that value to the variable cdImport and use the loadMovie function to move that new jpeg into that new movieclip.

All the CDs would be displayed on top of each other if we didn't move them around on the stage during this loop. We use modulus to determine which column the new movieclip should go to. Modulus is the "remainder" of a division equation. For example, 4 divides into 17 four times with a remainder of 1, so $17\%4=1$.

Instead of 'eval' above you could also use
_root[newName]._y

Macromedia Flash MX - Level II

Instructor: Gabino Travassos, Mote Interactive

TEXT FORMATTING IN DYNAMIC TEXT BOXES

To format text dynamically (such as changing type sizes and colours), Flash needs you to create a font object. Then we assign some properties and values to that object. In the example below we're changing the typeface, the colour, and the type size. We could also change the alignment, assign bold or italic, modify the tab stops, create bullets, leading, and other good things. No kerning, unfortunately.

```
myTextFormat = new TextFormat();
myTextFormat.font = "Frutiger 57Cn";
myTextFormat.color = 0x9999ff;
myTextFormat.size = 24;
```

```
// prepare text box first
artistName.setNewTextFormat(myTextFormat);
```

```
// add text to prepared box
artistName.text = "new text content";
```

The color property is in RGB hexadecimal format. It starts with a leading 0x (zero-x).

In the above example, "artistName" is the name of the dynamic text box we're targeting. First we use the setNewTextFormat function to assign our text formatting object to the dynamic text box.

In the last line we target the dynamic text box and change the text property to whatever our new text is.

This is an unfortunately rigid task order: create text object, apply object to dynamic text field, and then update the text.

THE SOUND OBJECT

Similar to the text object, Flash would like you to create a sound object when working with dynamic sounds. When you create a sound object you can refer to it as an object in actionscript and change the properties of that object, including the file being played, but also the volume, and panning.

```
beatz = new Sound();
beatz.loadSound(getSong,true);
```

The only sounds you can import are MP3s, and they can be whatever kbps you want, but they should be one of 11, 22 or 44 mHz. They also shouldn't have a variable bit rate.

The loadSound function has two properties you need to feed it: a string link to the sound file, and a boolean for whether you want the sound to stream or not. "true" is the typical response.

You might not want the song to start playing until a certain amount of it has downloaded to the user's browser. But how to determine that? getBytesLoaded() and getBytesTotal() will help.

```
If you want the entire sound file to preload:
if(beatz.getBytesLoaded()==beatz.getBytesTotal()){
    // do something } else { // loop }
```

Other necessary commands:

```
beatz.start();
beatz.stop();
```

Note that there is no pause(). When you stop() a sound, you have to restart it to get it playing again from the beginning.

If you want to display the length of the song, you have to wait until the song has completely loaded, then:
beatz.duration

If you want to get how many milliseconds of the song have elapsed:
beatz.position

When the song is done you might want to start another one. What you need to do is create a function for the onSoundComplete() function to call.

```
function goNextSong(){
    newSong="http://www.something.something.mp3";
    whichSongID++;
}
beatz.onSoundComplete=goNextSong;
```

Macromedia Flash MX - Level II

Instructor: Gabino Travassos, Mote Interactive

VARIABLES

Flash Actionscript uses several types of variables: boolean, numeric, string, array.

Boolean variables

The simplest variable. It is either 'true' or 'false', either 0 or 1.

Use it to turn the visibility of an object on or off.

```
basketball._visible=0;
```

or test it in a conditional statement

```
if(basketball._visible){  
    basketball._x+=15;  
}
```

In this example, the full statement is

```
if(basketball._visible==1)
```

but the way conditional statements work is

```
if(whatever is true){ do the rest of what's  
in the curly braces }
```

So

```
if(basketball._visible==1)
```

is the same as

```
if(basketball._visible)
```

Numeric variables

Numbers can be added, multiplied, etc. Like most variables, numeric variables have two parts: a name and a value.

```
MyGrossIncome=75000;  
MyTaxDeductions=28000;  
MyNetIncome=MyGrossIncome-MyTaxDeductions;  
// MyNetIncome=47000;
```

String variables

Strings are characters demarcated by quotation marks. Again, you have a variable name on the left and a value on the right.

```
MyDogsName="Smokey";  
MyCatsName="Mystery";  
When strings are added together it is called  
'concatenation';  
MyPetsNames=MyDogsName + " " + MyCatsName;  
// MyPetsnames="Smokey Mystery";
```

You can use single quotes or double quotes, but not interchangeably.

```
MyStreet="Arbour Stone Rise";  
MyQuadrant='NW';  
MyAddress =MyStreet + ' ' + MyQuadrant;
```

Array variables

Arrays are lists of variables. An array is an object in Flash and must be created with the command 'new Array.' The items in an array are separated by commas. Arrays can be strings:

```
MyPets=new Array("Smokey", "Mystery", "Nemo", "Mr.  
Chips");
```

or numeric, or a mix.

```
MyPetsAges=new Array(12,4,0.5,3);
```

Items in an array are accessed with the variable name, plus the position of the item in the list, starting with 0.

```
// MyPets[0]="Smokey";  
// MyPetsAges[2]=0.5;
```

Arrays can also be nested inside each other.

```
MyPets=new Array(new  
Array("Smokey",12,"dog", "Alpo"), new  
Array("Mystery",4,"cat", "Whiskas"), new  
Array("Nemo",0.5,"fish", "Seaweed flakes"),  
new Array("Mr. Chips",3,"finch", "seeds"));  
// MyPets[1][3]="Whiskas";
```

Arrays are useful for storing complex information about an object. Or for storing a list of similar items. In a game of Battleship, for example, I would create an array at the beginning of the game that listed all the places on the board where my opponent's ships were stored. Every time the player took a turn, an Actionscript loop would compare their guess to each value of the array.

```
OpponentShips=new Array("B7", "B8", "B9", "A1",  
"B1", "C1", "D1", "E1", "C7", "D7", "E7", "F7");
```

On click I would loop through the array, but first I need the array length.

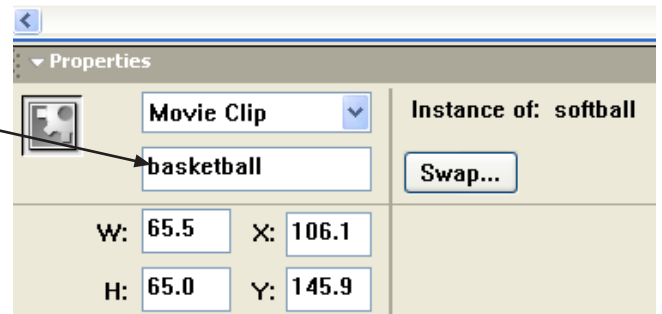
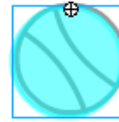
```
NumberShipLocations=OpponentShips.length;  
//12  
for(x=0;x<NumberShipLocations;x++){  
    if(myGuess==OpponentShips[x]){  
        // ship hit  
    }  
}
```

Macromedia Flash MX - Level II

Instructor: Gabino Travassos, Mote Interactive

MODIFYING AN OBJECT'S PROPERTIES

Objects on the stage can be easily affected by Actionscript. The first step is drawing an object on the stage and drag it to the Library. When you get the pop-up palette, choose MovieClip. Only MovieClips on the stage can be affected by Actionscript. Then your object needs a unique name in the Instance Name field in the Properties palette.



Object Instance Naming

Your object instance name cannot include the following characters: commas, spaces, brackets, apostrophes, quotes, and most punctuation. Numbers and letters are safe. You should avoid naming your objects words that are reserved in Flash for programming: root, stop, frame, color. A good naming convention: a descriptive word plus a number, like ball_04, or asteroid901.

You can put Actionscript in three places in Flash. The first place is on a frame. When you do this the frame gets a tiny letter 'a' above the keyframe dot.



If I wanted to change the size of an object called basketball I could put this on a frame:

```
basketball._width=150;
basketball._height=150;
```

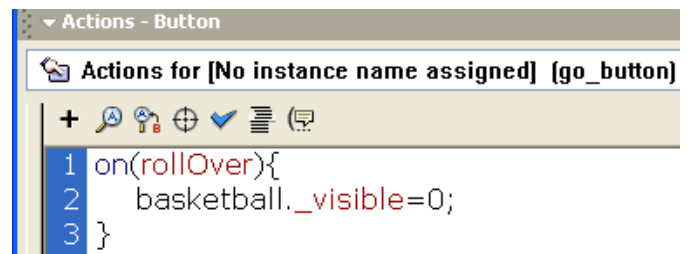
Other attributes I can change:

```
basketball._x = 400; // moves the ball to 400 pixels from the left of the screen
basketball._y = 300; // moves the ball to 300 pixels from the top of the screen
basketball._rotation+=15; // adds 15 degrees of rotation clockwise to the ball
basketball._yscale=150; // scales the width by 150%
basketball._xscale=85; // scales the height by 85%
basketball._alpha=50; // makes the object 50% transparent, min is 0 max is 100
basketball._visible=0; // object is invisible
basketball._visible=1; // object is visible, visible is only either 1 or 0
```

Properties of an object tend to have an underscore in front.

You can also put Actionscript on a button. You select the button on the stage with the black pointer tool, open the Actions palette and your first line has to include the on() function.

```
on(release){
basketball._visible=0;
}
```



You can tell that the button is selected because in the Action palette it should say Actions - Button at the top.

The third place for Actionscript is on the object itself. Select the object on the stage and open the Actions palette. All Actions on a movieClip object must start with onClipEvent.

```
onClipEvent(load){
this._alpha=40;
}
```

You can refer to a movieClip with the word 'this'.

