

Flash Action Script Lecture 6 Working with Sound Objects R. Berdan

Lesson 16 Scripting for Sound page 524-555

Objectives:

- 1) Create and use a sound object in a multimedia game of basket ball
- 2) Drag and drop object (movie clip) within defined boundary
- 3) Control the sound of an object (bouncing ball) based on its y coordinates
- 4) Control the panning of an object depending on its x coordinate within a quadrant
- 5) Start, Stop and loop sounds dynamically

Chapter 16 text book – we will do the exercises in order to make a basket ball draggable and vary its size and volume when it moves up and down the court and we will vary the pan – sound from the left and right speakers when the ball moves right or left.

In Flash Sound has three properties:

- 1) Volume – how loud it is
- 2) length - duration of clip
- 3) Panning – volume varies from right to left speakers

One of the ways to add sound to a Flash movie is to create a sound object and then control its various properties. To control sound on the root timeline, you need to create the sound object and associate it with the root timeline. Sound objects can also be used to control sounds in movie clips loaded into levels.

```
SoundobjectName = new Sound("TargetMovieclip")
```

e.g.

```
mySound = new Sound("_root.myMovieClip")
```

You can then add properties to the sound object e.g.

```
MySound.setVolume(50)        this sets the volume of the sound to 50%
```

Exercise #1 Open basketball.fla/Lesson 16/Assets

The main time line includes a basketball movie clip with a ball that bounces up and down and it includes a bounce sound.

Select the basketball movie clip – double click on it to view the bounce and animation then return to the main movie, select the movie clip and add the following script.

```
OnClipEvent(load)  
{  
bounce = new Sound(this) // create a sound object called bounce and target the movie clip  
}
```

```
this = _root.basketball
```

The script on the previous page creates a new Sound object named bounce that is associate with the basketball movie clip timeline. Since the “bouncing sound” is part of this timeline by controlling the bounce object we will be able to modify the volume and pan properties of the basketball movieclip.

First we want to vary the volume of the basket ball depending on the position of the basketball movieclip on the main time line (basket ball court). Therefore we are going to make the basketball movie clip draggable within a defined boundary. There are several ways you can define the area in which a movie clip is draggable.

Draggable movie clips within defined areas .

The easiest method is add the following code to the movie clip you want to drag in “normal mode” not expert mode.

```
on (press)
{
startDrag(movieclipname, true, 100, 100, 300, 300) // Left, Top, Right, Bottom coordinates
}
on (release)
{
stopDrag()
}
```

E.g. Create a new movie, drag a circle, convert it to a movie clip and name the instance “ball”. Select the ball and add the script below to make it draggable within a defined boundary.

```
on (press) {
    startDrag(ball, true, 100, 100, 300, 300);
}
on (release) {
    stopDrag();
}
```

myMovieClip.startDrag([lock, [left, top, right, bottom]])

Parameters

lock A Boolean value specifying whether the draggable movie clip is locked to the center of the mouse position (**true**), or locked to the point where the user first clicked on the movie clip (**false**). This parameter is optional.

left, top, right, bottom Values relative to the coordinates of the movie clip's parent that specify a constraint rectangle for the movie clip. These parameters are optional.

In the basket ball movie clip you will define the boundaries of the draggable area using variables – this is so the script is more dynamic and can be modified.

Add the following script to the first frame of the main time line.

```
onClipEvent(load)  
{  
bounce = new Sound();  
leftBoundary = 60;  
rightBoundary = 490;  
topBoundary = 220;  
bottomBoundary = 360;  
}
```

then add the following script

```
onClipEvent(mousemove)  
{  
    if (_root._xmouse > leftBoundary && root._xmouse < rightBoundary &&  
    root._ymouse > topBoundary && root.ymouse < bottomBoundary)  
    {  
        startDrag(this, true)  
    }  
    else  
    {  
        stopDrag();  
    }  
}
```

By using the mousemove event handler the if statement is tested everytime the mouse is moved.

Save the file as basketball3 fla

Controlling Volume of the basket ball movie clip depending on its position within the draggable area.

To modify the volume of a movie clip

`bounce.setVolume(70)` sets the bounce movie clip to a volume of 70%

But we want to vary the volume depending on its y position (vertical position) in the defined draggable area. Therefore

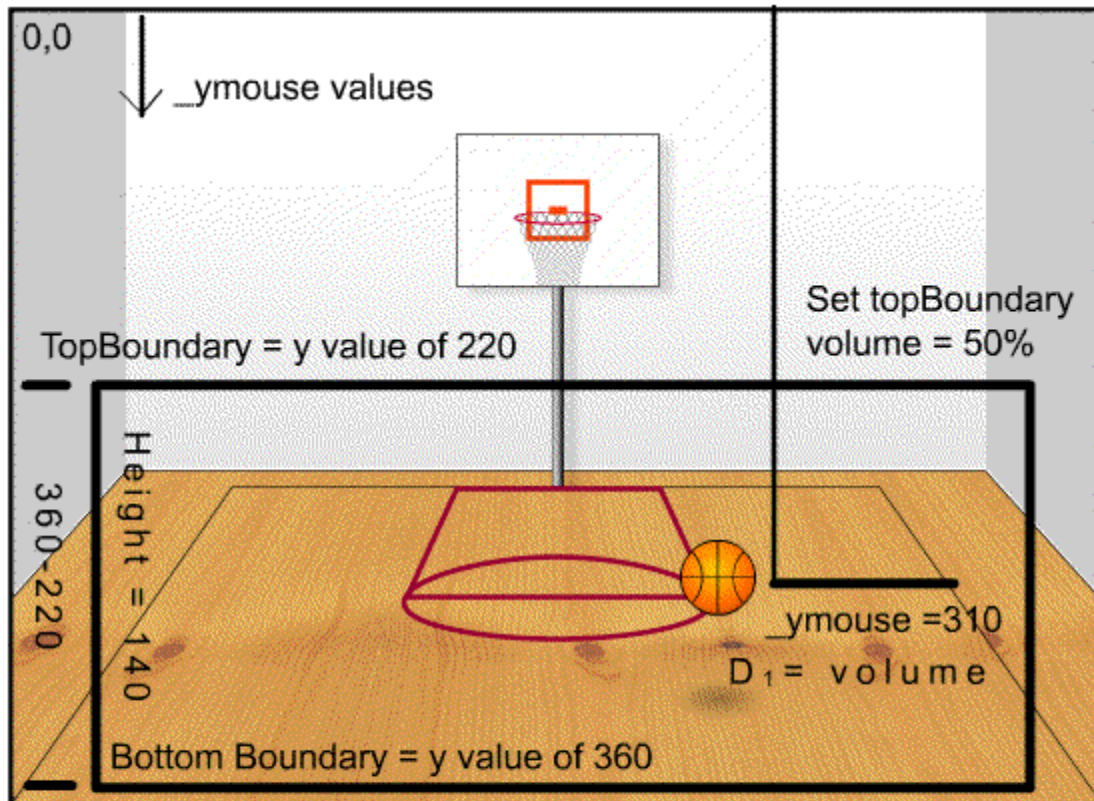
`bounce.setVolume(myvariable)`

we need to create a variable that reflects the y position in the defined draggable area. To do this we need to create a variable and constantly update its value between 50 and 100 percent.

Basically we will set the top boundary as equal to `setVolume(50)` and the bottom boundary to `SetVolume(100)`

The topboundary is 220 (50%) and the bottom boundary is 360 (100%) the difference between them is $360 - 220 = 140$ our vertical size. (see page 537) .

In order to set the volume of the bounce movie clip to a value between 50 and 100% we need to know the y position of the mouse relative to the top and bottom boundaries and constantly update the variable.



$D_1 = (310 - 220) / 140 \times 100 = \text{percent value from Top of main movie}$
 To convert so TopBoundary volume = 50% and Bottom 100% need to $D_1 / 2 + 50$

Eg. $D = 310$ then $90 / 140 \times 100 = 64.3\% / 2 = 32.15\%$
 However since we want the TopBoundary to be 50% we need to Add +50 to the `_ymouse` value $32.15 + 50 = 82.15\%$

`bounce.setVolume(D1)` we will create a variable called `topToBottomPercent`

`bounce.setVolume(topToBottomPercent)`

`topToBottomPercent = (((_root._ymouse - topBoundary) / BoundaryHeight) * 100) / 2 + 50)`

add this script to the onClipEvent(mousemove) after the if statement.

```
onClipEvent(load)
{
    bounce = new Sound()
    leftBoundary = 60;
    rightBoundary = 490;
    topBoundary= 220;
    bottomBoundary= 360;
    boundaryHeight = bottomBoundary – topBoundary
}

onClipEvent(mousemove)
{
    if (_root._xmouse>leftBoundary && root._ymouse < topBoundary && root._mouse <
leftBoundary && root._ymouse > bottomBoundary)
    {
        startDrag(this, true);
        topToBottomPercent = (((_root._ymouse – topBoundary)/BoundaryHeight)*100)/2 + 50)
        bounce.setVolume(topToBottomPercent)

    }
    else
    {
        stopDrag()
    }
}
```

To make the basket ball effect more realistic you will change the dimension of the movie clip when it is dragged within the boundary area by adding the following script below the bounce.setVolume()

```
this._xscale = topToBottompercent;
this._yscale= toptoBottompercent
```

this refers to the current object i.e. _root.basketball movie clip

CONTROLLING PANNING

Panning refers to the relative volume of the sound in the left and right speakers – to set Pan value:

```
bounce.setPan(myVariable)
```

The trick is make the variable **myVariable** reflect the distance of the bounce movie clip from the center of the draggable boundary area. Essentially you will need to know the width of the Boundary then divide it by 2 to set the center point and then monitor the _xmouse position relative to this center point.

To make the bounce.setPan(myVariable) work you will need to determine 3 things

- 1) Determine the horizontal size of the draggable boundary
- 2) Establish the position of the horizontal center
- 3) Determine the mouse `_xmouse` (horizontal) position in the quadrant relative to the center point.

1) $\text{boundaryWidth} = \text{rightBoundary} (490) - \text{leftBoundary}(60)$
 $= 430$

2) $\text{quadrant size} = \text{boundaryWidth}/2$
 $= 430/2 = 215$

3) $\text{centerPoint} = \text{rightBoundary} (490) - \text{quadrantSize}(215)$
 $= 275$

Create your variable (myVariable) and lets call it **panAmount** so we can vary the Pan as follows:

`panAmount = ((_root._xmouse - centerPoint)/quadrantsize) * 100`
`bounce.setPan(panAmount)`

If the mouse is in the right quadrant it will have a positive value, in the left quadrant it will have a negative value and both values will be converted to a percentage of the quadrant size so Pan will vary from 0 to 100%.

```
onClipEvent(load)
{
    bounce = new Sound(this)
    dynaSounds = new Sound();
    leftBoundary = 60;
    rightBoundary = 490;
    topBoundary = 220;
    bottomBoundary = 360;

    boundaryHeight = bottomBoundary - topBoundary; // value of 140
    boundaryWidth = rightBoundary - leftBoundary; // 430
    quadrantsize= boundaryWidth/2;
    centerpoint = rightBoundary - quadrantsize;
}

onClipEvent(mousemove)
{
    if(_root._xmouse > leftBoundary && _root._xmouse < rightBoundary && _root._ymouse >
topBoundary && _root._ymouse < bottomBoundary)
    {
        startDrag(this, true);
    }
}
```

```

    toptoBottomPercent = (((_root._ymouse - topBoundary)/boundaryHeight)
*100)/2) + 50;
    bounce.setVolume(topToBottomPercent);
    this._xscale = toptoBottomPercent;
    this._yscale = toptoBottomPercent;
    panAmount = ((_root._xmouse - centerpoint)/ quadrantsize)*100;
    bounce.setPan(panAmount);
    //trace(panAmount);

}
else
{
    stopDrag();
}
}

```

trace function can be used to see the panAmount() values and is used for debugging the script it can be removed once the script is working properly.

ATTACHING SOUNDS Dynamically and controlling playback

In non dynamic projects you can simply drag sound to the time line as either an event or streaming sound file.

Another way is to call sounds from the library only when they are needed. To this you will need to attach a sound file from the library. To do this you must first assign identifier names to all the sounds in the library – then you can attach them to Sound objects and control their playback (i.e. when they start and how many times they loop).

E.g. You have a sound file in the library – you give it the identifier name “rockMusic” you can now control its playback with the following code.

```

on (release)
{
    music = new Sound();
    music.attachSound(“rockMusic”); // attaches object to the library sound file
    music.start(0,5); // start 0 seconds into clip and loop 5 times
}

```

One of the advantages of controlling music this way is that you can set all the value using variables or expressions

In the next exercise you will attach a random sound from the library and trigger its playback when the mouse button is pressed and you will stop the sound when a key is pressed.

Open basketball5.fla in the Lesson 16/Assets folder.

First we must open the library and select the imported sound files – you will see three Sound 1, Sound 2 and Sound 3. These sounds only exist within the library and have not been placed in the main timeline.

- A) Select the first sound file, place your mouse over top and right click – select Linkage. (You can also do this by selecting the drop menu from the top right corner of the library panel).
- B) Set the Identifier name to Sound0 - **do not leave spaces** , select the checkboxes - script for ActionScript and Export in first frame.
- C) Repeat for Sound2 and Sound3

Note – although library-item names can contain spacers, identifier names can not – follow the same rules for naming variables.

Select the basketball movieclip and after the line of script that creates the bounce sound object (se below) add dynasounds = new Sound();

```
onClipEvent(load)
{
    bounce = new Sound(this)
    dynaSounds = new Sound();
}
```

This creates a new sound object called dynaSounds and loads it when the basketball movie clip is loaded into the scene.

Add the following script to the end of the current script

```
onClipEvent(mouseDown)
{
    randomSound = random(3); // creates 0, 1 and 2 i.e. Sound0, Sound1, Sound3
    randomLoop = random(2) + 1; // add 1 since 0 or no loop not useful
    dynaSounds.attachSound("Sound" + randomSound)
    //trace("Sounds" + randomSound);
    dynaSounds.start(0, randomLoop);
}
```

Then add the script below to stop the sounds when a key is pressed.

```
onClipEvent(keyDown)
{
    dynaSounds.stop();
}
```

