Notes 5: Collisions the dumb way

So far all we have done is bounce balls around. But what happens when balls hit each other? Should they just go through each other? Should they disappear? We are going to see how two balls can detect when they have hit each other and when they do so stop. We will also see how to detect if balls are hitting stationary objects and have them “stick” to the object when they hit it.

Before going further, in addition to google, use the HELP menu to look stuff up. In there you will find excellent information including class descriptions.

We need a starting point. Lets start with two balls bouncing around inside of a box. The box was created by drawing four lines around the Stage perimeter. Check out a5_1.fla

a5_1.fla :

```actionscript
var tempBall1:MovieClip;
var tempBall2:MovieClip;

// set up velocities for balls 1 and 2, xv1 means x velocity of ball 1
var xv1 = 6;
var yv1 = -5;
var xv2 = -8;
var yv2 = 4;

tempBall1 = attachMovie("ball", "b1", 1);
tempBall2 = attachMovie("ball", "b2", 2);

// This allows you to scale the size of the ball to your liking
var scaleFactor:Number = 30;
tempBall1._xscale = scaleFactor;
tempBall1._yscale = scaleFactor;
tempBall2._xscale = scaleFactor;
tempBall2._yscale = scaleFactor;

// put in initial locations
tempBall1._x = Stage.width / 2; tempBall1._y = Stage.height / 2;
tempBall2._x = Stage.width / 2; tempBall2._y = Stage.height / 2;

onEnterFrame = function () {
    // update new locations of balls 1 and 2
    var rv:Number;
    // rv = Math.random() * 10 - 5.0;
    rv = 0;
    tempBall1._x += (xv1 + rv);
    tempBall1._y += (yv1 + rv);
    tempBall2._x += (xv2 + rv);
    tempBall2._y += (yv2 + rv);
    // trace(tempBall1._x + " " + tempBall1._y);

    // check boundary conditions, if off screen wrap around from the other side
    if (tempBall1._x < 0) xv1 *= -1;
    if (tempBall1._y < 0) yv1 *= -1;
    if (tempBall2._x < 0) xv2 *= -1;
    if (tempBall2._y < 0) yv2 *= -1;
}
```

There really is nothing exciting here, just a cleaner implementation of what you have seen before. Now check out the code for making the two balls stop when they hit each other.

**a5_2.fla:**

```actionscript
var DEBUG:Number = 15; // the debugging level, smaller means more statement printed

var tempBall1:MovieClip;
var tempBall2:MovieClip;
var ballSpeed:Number = 2;

// set up velocities for balls 1 and 2, xv1 means x velocity of ball 1
var xv1 = 3 * ballSpeed;
var yv1 = -2 * ballSpeed;
var xv2 = -4 * ballSpeed;
var yv2 = 2 * ballSpeed;

tempBall1 = attachMovie("ball", "b1", 1);
tempBall2 = attachMovie("ball", "b2", 2);
var scaleFactor:Number = 190;
tempBall1._xscale = scaleFactor;
tempBall1._yscale = scaleFactor;
tempBall2._xscale = scaleFactor;
tempBall2._yscale = scaleFactor;

// put in initial locations, opposite corners
tempBall1._x = tempBall1._width + 10;
tempBall1._y = tempBall1._height + 10;
tempBall2._x = Stage.width - tempBall2._width - 10;
tempBall2._y = Stage.height - tempBall2._height - 10;

intersect = function(mc1:MovieClip, mc2:MovieClip):Boolean {
    var lx1:Number = mc1._x;
    var lx2:Number = mc2._x;
    var hx1:Number = lx1 + mc1._width;
    var hx2:Number = lx2 + mc2._width;

    var ly1:Number = mc1._y;
    var ly2:Number = mc2._y;
```
var hy1:Number = ly1 + mc1._height;
var hy2:Number = ly2 + mc2._height;

if (DEBUG < 10)
    trace("Inside intersect, lx1 hx1 ly1 hy1 lx2 hx2 ly2 hy2 equal: "+"n" +
        lx1 +"n" + hx1 +"n" + ly1 +"n" + hy1 +"n" +
        lx2 +"n" + hx2 +"n" + ly2 +"n" + hy2);

// if the X intervals do not intersect set xNotIntersect to true, else false
var xNotIntersect:Boolean = ( (hx1 < lx2) || (lx1 > hx2) ) ;

// if the Y intervals do not intersect set yNotIntersect to true, else false
var yNotIntersect:Boolean = ( (hy1 < ly2) || (ly1 > hy2) ) ;

if (DEBUG < 10)
    trace("xNot, yNot = " + xNotIntersect + " + yNotIntersect) ;

if (xNotIntersect || yNotIntersect) {
    // trace("returning false") ;
    return(false) ;
}
else {
    // trace("returning true") ;
    return(true) ;
}

onEnterFrame = function () {

    // uncomment below if you want random motion
    var rv:Number ;
    // rv = Math.random() * 10 - 5.0 ;
    rv = 0 ;

    // update new locations of balls 1 and 2
    tempBall1._x += (xv1 + rv) ;
    tempBall1._y += (yv1 + rv) ;
    tempBall2._x += (xv2 + rv) ;
    tempBall2._y += (yv2 + rv) ;

    // check if balls hit each other, if so stop the balls
    if (intersect(tempBall1,tempBall2) ) {
        xv1 = 0 ;
        xv2 = 0 ;
        yv1 = 0 ;
        yv2 = 0 ;
    }

    // check boundary conditions, if hit boundary reflect by negating velocity
    if (tempBall1._x < 0) xv1 *= -1 ;
    if (tempBall1._x > (Stage.width - tempBall1._width) ) xv1 *= -1 ;
    if (tempBall1._y < 0) yv1 *= -1 ;
    if (tempBall1._y > (Stage.height - tempBall1._height) ) yv1 *= -1 ;
    if (tempBall2._x < 0) xv2 *= -1 ;
    if (tempBall2._x > (Stage.width - tempBall2._width) ) xv2 *= -1 ;
if (tempBall2._y < 0)  yv2 *= -1 ;
if (tempBall2._y > (Stage.height - tempBall2._height) )  yv2 *= -1 ;
}

There is a bunch of stuff worthy of note here. First, a minor point, I made the balls bigger so that they would collide sooner.

The real concept of interest here is “collision”. If two object touch/overlap one another they are said to collide. In the onEnterFrame function a check is done to see if the two balls intersect, if they do their velocities are set to zero, thus stopping the balls. In addition the usual boundary check is done.

The function intersect( ) should be looked at carefully. The arguments are two MovieClips. The bounding boxes of the MovieClips are the rectangles that fully enclose the object. Each box has a lower-x coordinate (lx), and high-x coordinate (hx), a lower-y (ly), and an upper y (hy). Because there are two boxes being compared the variables are suffixed with 1 or 2: lx1, hx1, lx2, hx2, ly1, hy1, ly2, hy2. First these variables are initialized from the MovieClip objects. Now we test to see if the boxes intersect. In order for the boxes to intersect their x-intervals must overlap AND their y-intervals must overlap. If either does not overlap then the boxes do not intersect. The code makes these checks, and returns true if the boxes intersect, false otherwise.

Note, the bounding boxes of the two circles intersect, but the actual circles do not touch. Doing this correctly (treating them as circles) is a bit harder and will be talked about if time allows.

The above code works assuming that the objects intersect at the time the intersection check, but, it may be the case that they pass through each other and the collision is missed. Consider the example below:

Frame 1:

![Frame 1](image1)

Frame 2:

![Frame 2](image2)

In this case it will look like the balls moved right through each other and no collision ever occurred, yet a collision should have occurred. We will talk about how to solve this later. An example of this can be found in a5_3.fla. Run the code, you can see the balls pass right through each other.

a5_3.fla
This is the same code as a3_2.fla, only the Stage, speed, and size parameters have been changed.

The above only considers the intersection of two MovieClip objects. It could be the case you want to test a MovieClip object against several other objects or regions. Perhaps the objects are stationary and you know their coordinates (in Flash you can get a screen object’s coordinates by clicking on the object and looking at the Properties Panel.) The following code shows a modification of the intersect() function to take arguments of a MovieClip and a fixed bounding box.

a5_4.fla:

```actionscript
var DEBUG:Number = 15 ; // the debugging level, smaller means more statement printed

var tempBall1:MovieClip ;
var tempBall2:MovieClip ;
var ballSpeed:Number = 2 ;

// set up velocities for balls 1 and 2, xv1 means x velocity of ball 1
var xv1 = 3 * ballSpeed ;
var yv1 = -2 * ballSpeed ;
var xv2 = -4 * ballSpeed ;
var yv2 = 2 * ballSpeed ;

tempBall1 = attachMovie("ball", "b1", 1);
tempBall2 = attachMovie("ball", "b2", 2);
var scaleFactor:Number = 40 ;
tempBall1._xscale = scaleFactor ;
tempBall1._yscale = scaleFactor ;
tempBall2._xscale = scaleFactor ;
tempBall2._yscale = scaleFactor ;

// put in initial locations
tempBall1._x = tempBall1._width + 10 ;
tempBall1._y = tempBall1._height + 10 ;
tempBall2._x = Stage.width - tempBall2._width - 10 ;
tempBall2._y = Stage.height - tempBall2._height - 10 ;

// tempBall1._x = Stage.width / 2;
// tempBall1._y = Stage.height / 2 ;
// tempBall2._x = Stage.width / 2 ;
// tempBall2._y = Stage.height / 2 ;

intersect = function(mc1:MovieClip, mc2:MovieClip):Boolean {
    var lx1:Number = mc1._x ;
    var lx2:Number = mc2._x ;
    var hx1:Number = lx1 + mc1._width ;
    var hx2:Number = lx2 + mc2._width ;

    var ly1:Number = mc1._y ;
    var ly2:Number = mc2._y ;
```
var hy1:Number = ly1 + mc1._height;
var hy2:Number = ly2 + mc2._height;

    // if the X intervals do not intersect set xNotIntersect to true, else false
    var xNotIntersect:Boolean = (hx1 < lx2) || (lx1 > hx2);

    // if the Y intervals do not intersect set yNotIntersect to true, else false
    var yNotIntersect:Boolean = (hy1 < ly2) || (ly1 > hy2);

if (DEBUG < 10)
    trace("xNot, yNot = " + xNotIntersect + " " + yNotIntersect);

// if either interval does not intersect, hit is false, else both do and hence true
if (xNotIntersect || yNotIntersect)
    return(false);
else
    return(true);
}

intersectRectangle = function(mc1:MovieClip, lx2:Number, hx2:Number,
    ly2:Number, hy2:Number):Boolean {
    var lx1:Number = mc1._x;
    var hx1:Number = lx1 + mc1._width;
    var ly1:Number = mc1._y;
    var hy1:Number = ly1 + mc1._height;

    // if the X intervals do not intersect set xNotIntersect to true, else false
    var xNotIntersect:Boolean = (hx1 < lx2) || (lx1 > hx2);

    // if the Y intervals do not intersect set yNotIntersect to true, else false
    var yNotIntersect:Boolean = (hy1 < ly2) || (ly1 > hy2);

    // if either interval does not intersect, hit is false, else both do and hence true
    if (xNotIntersect || yNotIntersect)
        return(false);
    else
        return(true);
}

onEnterFrame = function () {

    // uncomment below if you want random motion
    var rv:Number;
    // rv = Math.random() * 10 - 5.0;
    rv = 0;

    // update new locations of balls 1 and 2
    tempBall1._x += (xv1 + rv);

tempBall1._y += (yv1 + rv) ;
tempBall2._x += (xv2 + rv) ;
tempBall2._y += (yv2 + rv) ;

// Check if tempBall1 hit any of the three obstacles (rectangles). NOTE, I
// obtained the boundaries of the obstacles directly from the property
// panel of flash and hard-coded them
if (intersectRectangle(tempBall1,100,120,90,140) ) { xv1 = 0 ; yv1 = 0 ;}
if (intersectRectangle(tempBall1,240,260,170,210)) { xv1 = 0 ; yv1 = 0 ;}
if (intersectRectangle(tempBall1,90,120,260,310) ) { xv1 = 0 ; yv1 = 0 ;}
if (intersectRectangle(tempBall2,100,120,90,140) ) { xv2 = 0 ; yv2 = 0 ;}
if (intersectRectangle(tempBall2,240,260,170,210)) { xv2 = 0 ; yv2 = 0 ;}
if (intersectRectangle(tempBall2,90,120,260,310) ) { xv2 = 0 ; yv2 = 0 ;}

// check boundary conditions, if off screen wrap around from the other side
if (tempBall1._x < 0) xv1 *= -1 ;
if  (tempBall1._x > (Stage.width - tempBall1._width) ) xv1 *= -1 ;
if (tempBall1._y < 0)  yv1 *= -1 ;
if (tempBall1._y > (Stage.height - tempBall1._height) ) yv1 *= -1 ;
if (tempBall2._x < 0) xv2 *= -1 ;
if (tempBall2._x > (Stage.width - tempBall2._width) ) xv2 *= -1 ;
if (tempBall2._y < 0)  yv2 *= -1 ;
if (tempBall2._y > (Stage.height - tempBall2._height) ) yv2 *= -1 ;

The above code uses the function intersectRectangle( ) to compare a movie clip with a
fully specified rectangle, other than that it is the same.

EXERCISE 5.1

Write code that tests for intersection of an array of balls against a fixed set of
three objects.
Stuff to come:

The Library

GoToAndPlay, GotoAndStop

Functions and Scope

Input and Output with dynamic text boxes and variables

Arrays

Objects/Classes

Arrays of Objects

Mouse Handlers (in each frame)

Keyboard Handlers (in each frame)

Listeners