Archived:Optimized Collision detection in Flash Lite

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

We do not recommend Flash Lite development on current Nokia devices, and all Flash Lite articles on this wiki have been archived. Flash Lite has been removed from all Nokia Asha and recent Series 40 devices and has limited support on Symbian. Specific information for Nokia Belle is available in Flash Lite on Nokia Browser for Symbian. Specific information for OLD Series 40 and Symbian devices is available in the Flash Lite Developers Library.

Introduction

Collision detection plays an important role in game design. In most games, action is required to be taken upon collision between two objects. Say for example in space shooter kind of games, when the bullet hits the spaceship, there is a explosion. Now to test collisions there are lot of ways. The simplest for a Flash Lite user is however, using the hitTest() function.

Simplest Way

The code below checks for collision between two movieclips – mc and mc2. It is assumed that these are already on stage and posses the same Y coordinate(Just for simplicity). Now paste the following code on an actions layer

```
onEnterFrame = function () {
 mc._x++;
 if (mc._x>=Stage.width) {
  mc._x = -mc._width;
 }
 if(mc.hitTest(mc2)) {
  trace("Collided");
 }
};
```

Here, we move one movieclip (mc) and wrap its motion about the boundaries of the Stage. You will observe that whenever mc and mc2 collide, we get the trace message indicating a Collision.

The hitTest function in Flash involves massive computation. This is because it assumes a generalized algorithm to check collision. In our case, however, by placing two circular objects, this is not needed. We can simple the process and reduce the computation involved by making it simpler.

```
onEnterFrame = function () {
 mc._x++;
 if (mc._x>=Stage.width) {
  mc._x = -mc._width;
 }
 if(Math.ceil(mc2._x)==Math.ceil((mc._x + mc._width)))
 { trace("collided"); }
};
```

Here, as both movieclips share the same Y coordinate, we check the collision by just checking the X coordinate values. By executing the above code, you will understand that the first collision is detected by this code snippet. We are forced to use Math.ceil() to convert to X coordinate values to an integer.

Optimized approach

However, this may just be an artificial case as two bodies (whose collision needs to be detected need not lie on the Y coordinate). Hence, we now introduce a more generalized procedure to check collision. The logic underlying this test would be compute the distance between two bodies and then to relate it with their sum of widths.

Now having the same two movieclips on stage and moving the mc2 clip slightly down (incrementing its Y coordinate), we demonstrate the more general case. The center of each object can be deduced to (mc._x + mc._width/2, mc._y + mc._height/2). This holds good when both their registration points are Top-Left corner. Now if the distance between their centers is less than the sum of their widths, a collision said to have occurred.We can recollect that the distance between two points can be calculated from

Dist = Sqrt(square(x1-x2) + square(y1-y2))

To check this paste the following code snippet and try it out yourself.

```
var widthsum = (mc._width+mc2._width)/2;
Y_coord = mc._y;
Mc2_x = mc2._x+mc2._width/2;
Mc2_y = mc2._y;
onEnterFrame = function () {
 mc._x++;
 if (mc._x>=Stage.width) {
  mc._x = -mc._width;
 }
 distanceBetween = dist(mc._x+mc._width/2, Y_coord, Mc2_x, Mc2_y);
 if (distanceBetween<widthsum) {
  trace("collided");
 }
};
function dist(x1:Number, y1:Number, x2:Number, y2:Number):Number {
 sample = Math.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
 return sample;
}
```

You may note that since mc2 is static I compute its center outside the onEnterFrame and this applies to the static Y co-ordinate and the sum of their widths. Its lucrative to reduce the computation inside the looping constructs or onEnterFrame functions. This type of checking is the most fundamental way to check collisions between circular objects.

Download Code

A sample Code supporting this article is available Media:Collision.zip.

--manikantan 08:46, 25 April 2009 (EEST)