
2DIY ActionScript Tutorials

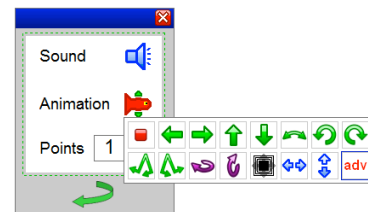
T8 - Add orbiting elements to your game

Outline

In several 2DIY activities, you can place additional elements on the screen that can be used to orbit around other elements. You can make monster, apple or sun elements orbit around other elements, and even make your main character orbit around another element too.

Code Explanation

Place an element on screen that will become the orbitee, and place another element on top of it that will become the orbiter. Right click on the orbiter element and click on the animation option (the second option). Within this option there is an ‘advanced’ setting where you can enter ActionScript code, and here we can set a rule to cause this element to circle around another element.



Tutorial

Create a platform activity. Add an apple element to the screen, and then place a monster on screen above the apple element. Right click on the monster element, and look for the animation settings option (the second option). Choose the advanced option and type in the following code;

```
var angle = Math.atan2(this._y - _root.s13._y, this._x - _root.s13._x) + 0.05;  
    this._x = _root.s13._x + Math.cos(angle) * 100;  
    this._y = _root.s13._y + Math.sin(angle) * 100;
```

var angle this sets up a variable that we are calling ‘angle’

Math.atan2 this is a trigonometrical function (remember your tan / sin / cos from schooldays?) that works out the arctan of the difference between the two elements you are using.

`(this._y - _root.s13._y, this._x - _root.s13._x)` this tells the activity which values to use to find the arctan above

*the value of `_root.s13` will change - see notes below

`+0.05`; this value adds a small amount to the variable 'angle' that results in the orbit effect. We can alter this to create a faster orbit (use a smaller value)

`this._x = _root.s13._x + Math.cos(angle) * 100;`
`this._y = _root.s13._y + Math.sin(angle) * 100;` these two lines together are used to calculate the position of the object that is orbiting. It calculates the position of the orbitee, using the cosine or sine of the angle, and multiplies it by a value to create the orbit radius. The smaller the value, the closer the orbiter appears to the orbitee.

Now press the play button. Your character moves very slowly to begin with, but gradually speeds up as apples are collected, and the value of `dx` rises.

What does this do?

By using trigonometry to calculate the angle between the orbitee and the orbiter, the result is the ability to create an effect of one element orbiting around another object.

Task

Create a platform game. Add a character, and then add some apple elements at various places. Place monster elements on top of the apple elements and add the orbiting code. Now each apple is being 'protected' by a monster. Can you safely avoid them and collect all of the apple?

Notes

You can create a situation whereby the main character becomes the orbiter - and you can steer the orbitee, with your main character orbiting it, around the screen. To achieve this use a sun element and add the code to this (you cannot add actionscript to the main character).

Because you are adding the code in a different location, you do need to change the code slightly, and instead of using

```
var angle = Math.atan2(this._y - _root.s13._y, this._x - _root.s13._x) + 0.05;
```

More information, examples and help can be found at <http://www.2diyarchive.co.uk>

2DIY is ©2Simple Software <http://www.2simple.com/2diy>

you need to use

```
var angle = Math.atan2(_root.player._y - _root.s22._y, _root._x -  
_root.s22._x) + 0.05;
```

Make sure you place the `_root.player` code before the `_root.s22` code, as you want the player to orbit the element, and not the other way around.

You also need to add some code to respond to key presses to move the sun element around the screen. This has been mentioned in other tutorials, but to recap;

```
if(Key.isDown (80)==true) {this._x += 1;} to use the P key to move right  
if(Key.isDown (79)==true) {this._x -= 1;} to use the O key to move left  
if(Key.isDown (81)==true) {this._y -= 1;} to use the Q key to move up  
if(Key.isDown (65)==true) {this._y += 1;} to use the A key to move down
```

To work out the value of the orbitee element use the following;

In a collecting or journey game; `_root.s2`-`_root.11` (monsters), `_root.s12`-`_root.s21` (apples), `_root.s22`-`_root.s31` (suns) Further objects added will be `_root.s32`, `_root.s33` etc

In a platform or snake game; `_root.s3`-`_root.12` (monsters), `_root.s13`-`_root.s22` (apples), `_root.s23`-`_root.s32` (suns) Further objects added will be `_root.s33`, `_root.s34` etc

In collecting, platform and snake games, use `_root.player` for the main character

In journey games use `_root.car` for the main character

You can find out more about this feature on the 2DIY archive:

<http://www.2diyarchive.co.uk/2009/03/make-a-monster-orbit-around-a-collectable-item.html>