**- Introducción.**

En este tutorial vamos a aprender cómo funciona los bucles y temporizadores y para qué sirven. Un bucle es un código que nos permite repetir acciones un número repetido de veces. Un temporizador sería algo parecido a un bucle pero además podemos configurar el tiempo.

## 1.- Creación de un bucle sencillo.

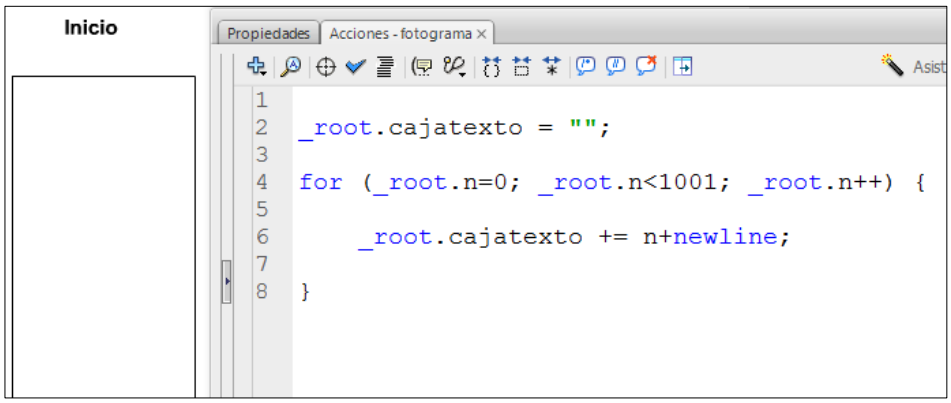


Imagen 1

El bucle anterior permite visualizar en pantalla, en un cuadro de texto, los números del uno al mil. La caja de texto tiene en sus propiedades la variable **_root.cajatexto** para almacenar los números.

Como se puede observar la acción **for** tiene tres parámetros separados por el carácter ";".

El primer parámetro nos sirve para decir qué valor tendrá la variable **_root.n** cuando empiece a ejecutar el bucle. En el segundo parámetro se declara la condición que en nuestro caso significa que se repita el bucle mientras el valor de la variable **_root.n** sea menor que 1001. Y en el tercer parámetro le decimos que aumente el valor de la variable **_root.n** cada repetición en una unidad.

## 2.- Creación de un bucle para duplicar clips de película.

Vamos a crear ahora un programa que nos permita crear clips duplicados de forma automatizada utilizando un bucle.

Para ello vamos a crear tres objetos en el escenario:

- Un botón que podemos llamar duplicar
- Un clip de película que tiene 10 fotogramas con bolas de diferente color.
- Un botón que hará de papelera


Imagen 2

La idea es que al presionar el botón duplicar se pueda crear multitud de bolitas y que se puedan borrar arrastrándolas a la papelera.

Utilizaremos la acción **duplicateMovieClip** que ya hemos estudiado en el tutorial anterior por lo que no voleremos a explicarla aquí. Lo que nos interesa ahora es ver una aplicación más del bucle **for.**

Para empezar necesitaremos definir una variable que llamaremos _root.capa y que la utilizaremos para almacenar el nivel de profundidad (capa) que tendrá cada bolita en el escenario. Por lo tanto pondremos en el fotograma de la película principal **_root.capa=0;** para inicializar la variable.

Pasamos ahora a programar el botón que hace las duplicaciones:

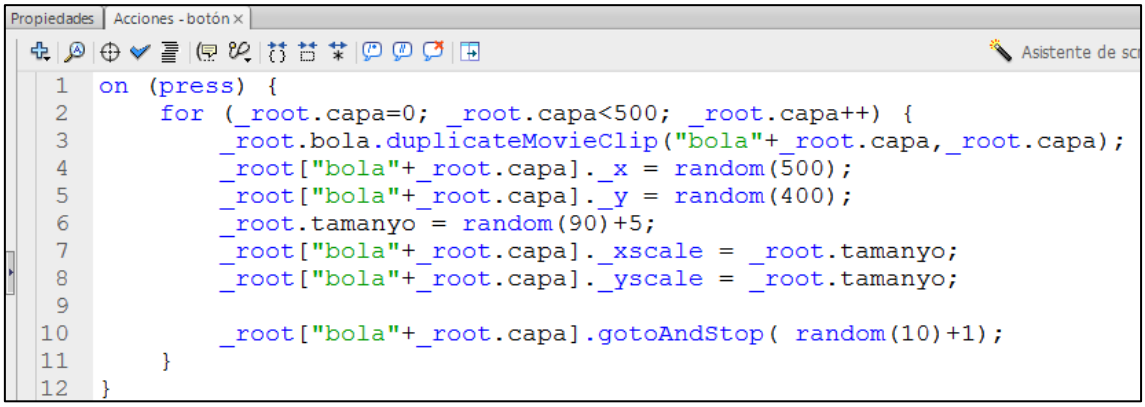


Imagen 3

Como hemos dicho anteriormente los bucles tienen 3 parámetros y el código que está entre llaves es el que se repite.

En el ejemplo anterior además de repetir las duplicaciones, el bucle hace más cosas:

- Coloca aleatoriamente cada bola duplicada en unas coordenadas aleatorias en el escenario.
- Asigna a cada bola un tamaño aleatorio con una variación de 90 en la escala y un mínimo de 5.
- Elige aleatoriamente un fotograma del clip bola. En el clip bola hemos creado 10 fotogramas donde hay bolas de diferente color.
-

Ahora pasamos a escribir el código de clip bola. Recordemos que la idea es que este clip se pueda arrastrar y que al colisionar con la papelera se borre.

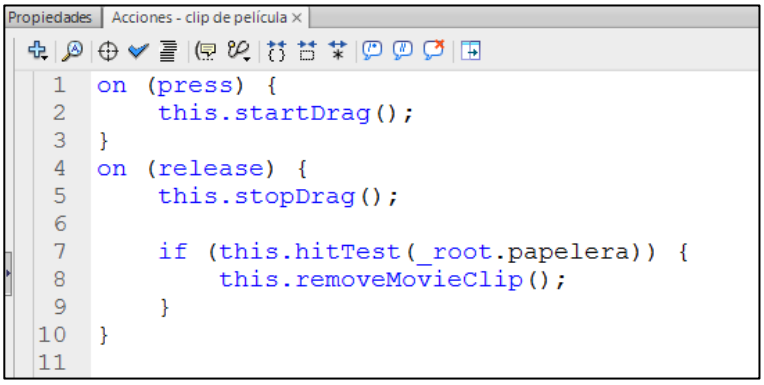


Imagen 4

Con la función **hitTest** lo que comprobamos es si en clip que estoy arrastrando (this=éste) ha colisionado con el clip papelera. Vamos hacer una observación importante y es que la función va precedida por la palabra **this** y significa que nos referimos al clip donde el código está actuando. No sería lo mismo poner **_root.hitTest** que **this.hitTest.** En el primer caso estaríamos comprobando la colisión de la película principal. Lo bueno que tiene utilizar la acción **this** es que cuando se duplica el clip de película el código funciona igual porque no se refiere a un nombre propio de clip si no al mismo que se ha duplicado.

Por último hay que recordar poner al clip papelera el nombre de instancia (papelera).

### 3. Creando un temporizador.

Pero si lo que queremos es que las duplicaciones ocurran cada cierto tiempo entonces será mejor utilizar otro tipo de acciones donde se controle más el tiempo.

Vamos a modificar el ejemplo anterior para hacer que las bolas se dupliquen cada un cierto tiempo para crear el efecto de que se vayan surgiendo poco a poco. Ponemos el código en el fotograma principa.

```
1  _root.capa = 1;
2  var intervalo = setInterval(segundos, 5);
3
4  function segundos() {
5      _root.capa++;
6      _root.bola.duplicateMovieClip("bola"+_root.capa,_root.capa);
7      _root["bola"+_root.capa]._x = random(500);
8      _root["bola"+_root.capa]._y = random(400);
9      _root.tamanyo = random(90)+5;
10     _root["bola"+_root.capa]._xscale = _root.tamanyo;
11     _root["bola"+_root.capa]._yscale = _root.tamanyo;
12
13     _root["bola"+_root.capa].gotoAndStop(random(10)+1);
14
15
16     if (_root.capa>5000) {_root.capa=1}
17
18
19
20 }
```

Imagen 4

Vamos a ver las modificaciones que tenemos que hacer del programa anterior.

En primer lugar hemos utilizado la función **setInterval(segundos,5);** Esta función tiene dos parámetros. El primer parámetro es una función que hemos creado llamada **segundos.** El nombre puede ser el que queramos. Y el segundo parámetro es el tiempo en milisegundos. Es decir en este ejemplo se ejecutaría la función **segundos** cada 5 milisegundos.

En segundo lugar hemos creado una función **segundos**. Como vemos Flash como lenguaje de programación permite crear tus propias funciones. Una función es como un conjunto de acciones a las que damos un nombre. Así cuando queramos ejecutar esas acciones solo hacemos la referencia al nombre de la función. Para crear una función se pone este código:

**function nombre_de_la_funcion() {**

**Aquí se pone el código**

**}**

En el ejemplo de la imagen 4 vemos como la función **setInterval** llama cada 5 milisegundos a la función **segundos().**

Pero entonces tenemos un problema. Si se repite siempre las duplicaciones serán infinitas. Hay dos soluciones. Una es controlar con una sentencia condicional el número de duplicaciones y reiniciar la duplicación como en el ejemplo. La otra solución es desactivar la función intervalo. Para ello insertamos esté código:

**if (_root.capa>100) {clearInterval(intervalo)}**

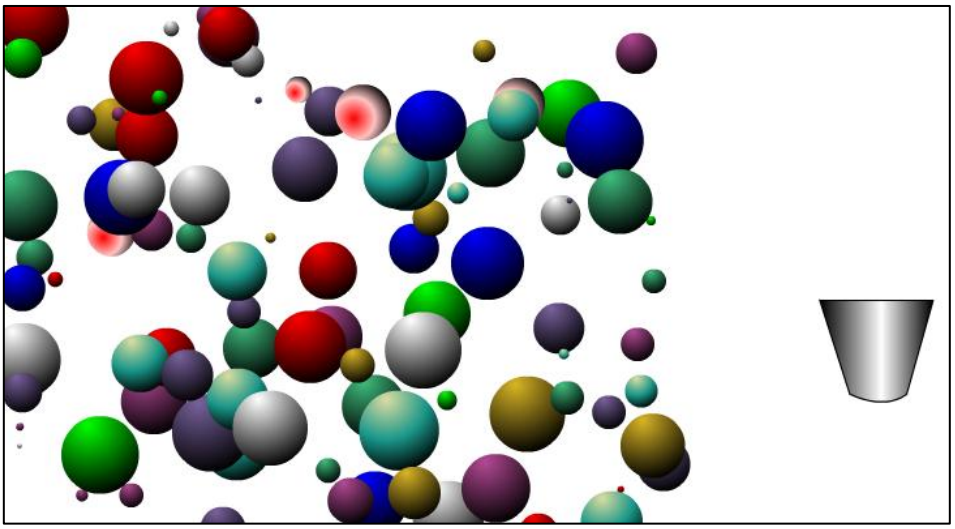

Imagen 5

Los temporizadores tienen muchas aplicaciones. Algunos ejemplos:

- Aplicaciones de cálculo mental donde aparecen números cada cierto tiempo.
- Aplicaciones para trabajar la memoria donde aparecen imágenes o dibujos y después hay que recordarlos.
- Aplicaciones que hacen sonar notas musicales con ritmos basados en tiempos.

**4.- Ejercicio propuesto.**

Propuestas:

1. Un programa donde vayan saliendo números aleatorios cada dos segundos más o menos y se tengan que clasificar bajo algún criterio.
2. Propuesta libre basada en el uso de temporizadores.