ADOBE FLASH • **ActionScript**

Contenido

1. ¿Qué es ActionScript?

a. P.O.O. (Programación Orientada a Objetos):
La Programación Orientada a Objetos (POO) es un estilo de programación en el que el programador crea objetos del mundo real, es decir, está basada en el modelo de los objetos del mundo real, que poseen una serie de características (que llamamos *propiedades*) y realizan una serie de acciones (que denominamos *métodos* o *funciones*).
La intención de la POO es que el programador pueda crear una serie de objetos completamente funcionales para luego hacerlos trabajar juntos en un proyecto (programa).

b. ActionScript:
Es el nombre del lenguaje de Programación utilizado por Adobe Flash, que en la versión CS4 llega a la versión 3.0.
Como en cualquier idioma, todo lenguaje de programación consiste en una serie de palabras y reglas que permiten dar sentido a una oración (lo que en programación llamamos *sentencia*). Por ejemplo, en el castellano se utiliza un punto para terminar una oración, en ActionScript se utiliza el punto y coma (;) para terminar una sentencia. También como en los idiomas, una sentencia puede estar conformada por una o más palabras o frases. A un conjunto de sentencias se le suele llamar *bloque de código*.

c. Objetos Programables:
Podemos escribir código de ActionScript en: cualquier fotograma clave de la línea de tiempo (incluyendo la línea de tiempo interna de un MovieClip), cualquier instancia de MovieClip o Botón, o cualquier instancia de un Componente.

2. Conceptos Básicos

a. Variables:
Son porciones de memoria RAM que permiten almacenar datos para ser utilizados luego por la aplicación. Existen tres tipos de variables: *local*, *global* y *de Línea de Tiempo*.

b. Operadores:
Son símbolos matemáticos que realizan ciertas operaciones, sean aritméticas (sumar [+], restar [-]), de comparación (mayor que [>], menor que [<]), etc. Estos son utilizados dentro de las sentencias de diversos modos, según sean necesarios.

c. Instancia:
La comparación más sencilla sería la familiar padre-hijo. Cuando creamos un símbolo en Flash, o importamos un objeto, éste va directamente a la Biblioteca, sería el objeto Padre; cuando colocamos este objeto en el Escenario, realmente estamos colocando una *instancia* del objeto, es decir, colocamos un hijo concreto del objeto padre que está en la biblioteca. Este hijo hereda todas las propiedades y animaciones (si las hubiere) del objeto padre, pero además tiene identidad propia, lo que llamamos *nombre de Instancia*.

d. Propiedad:

Llamamos *propiedad* a cualquier característica propia de un objeto, como en los objetos del mundo real, un objeto en Flash puede tener ciertas características como el color, la altura, el ancho, etc.

e. Método:

Al igual que como ocurre en el mundo real, en el que los objetos realizan ciertas funciones (como el encendido/apagado de un aparato eléctrico o la forma como una ventana se abre/cierra), los objetos en la programación tienen ciertas formas para realizar acciones (cargar un Movie Clip externo, reproducir un sonido, etc.). Estas formas están determinadas por bloques de código que son transparentes al usuario (es decir, que el usuario no las ve) y permiten que los objetos realicen ciertas acciones que les son propias.

f. Eventos:

Acciones que tienen lugar durante la reproducción de un archivo SWF. Por ejemplo, cuando se carga un clip de película se generan diferentes eventos: la cabeza lectora accede a un fotograma, el usuario hace clic en un botón o clip de película o el usuario introduce información mediante el teclado.

g. Tipo de Datos:

Indica el tipo de información que puede contener una variable, p.e.: el tipo de dato *Number* sólo permite números; el tipo de dato *String*, texto, etc. Si se asigna un tipo de datos a una variable se evita que en el código se le asigne un tipo de información distinta a la que se requiere.

h. Niveles:

Flash Player tiene un orden de apilamiento con varios niveles. Todas los documentos de Flash tienen una línea de tiempo principal situada en el nivel 0 de Flash Placer. Si carga documentos en niveles superiores al nivel 0, los documentos se colocarán uno encima de otro, como si se tratara de dibujos en papel transparente; cuando no haya contenido en el escenario, podrá ver el contenido de los niveles inferiores. Si carga un documento en el nivel 0, sustituye la línea de tiempo principal. Cada documento cargado en un nivel de Flash Player tiene su propia línea de tiempo.

3. Eventos “on”, “onClipEvent”, “Listeners”

a. Evento “on()” - (AS2)

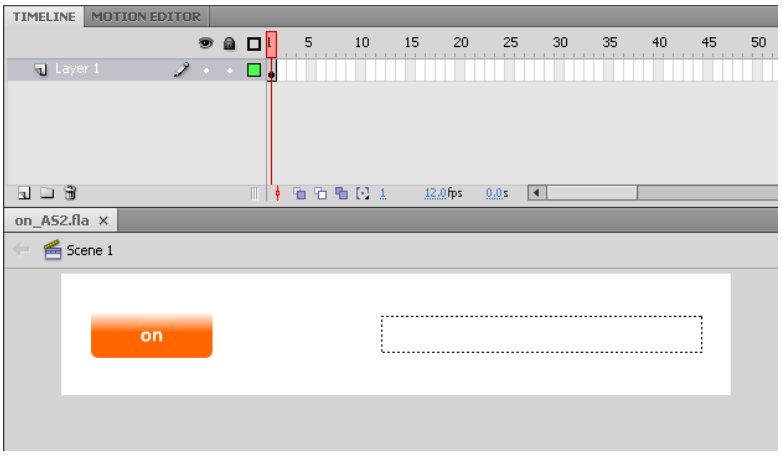
Sintaxis:

```
on(eventoDeRatónOTecla){
  sentencia(s);
}
```

Se utiliza para programar un reaccionar ante una acción del usuario, ya sea por medio del ratón o del teclado, esta acción se denomina *Evento*. Cuando el usuario desencadena el evento especificado se ejecutan las sentencias que están entre las llaves.

Ejercicio on_AS2.fla:

Para este ejercicio haremos un clip de película que indique las acciones realizadas sobre un botón. Colocamos en el escenario un botón y un campo de texto dinámico como se muestra en la siguiente figura y les damos nombre de instancia “boton_btn” y “evento_txt” respectivamente.



Existen dos opciones para asignar el código, asociado al botón directamente o al frame.

En el primer caso el botón no necesita nombre de instancia y la sintaxis del código que usaremos es:

```
on(rollOver){
    evento_txt.text = "Roll Over";
}

on(rollOut){
    evento_txt.text = "Roll Out";
}

on(press){
    evento_txt.text = "Press";
}

on(release){
    evento_txt.text = "Release";
}

on(releaseOutside){
    evento_txt.text = "Release Outside";
}
```

En el segundo caso la sintaxis es:

```
boton_btn.onRollOver = function() {
	evento_txt.text = "Roll Over";
}

boton_btn.onRollOut = function() {
	evento_txt.text = "Roll Out";
}

boton_btn.onPress = function() {
	evento_txt.text = "Press";
}

boton_btn.onRelease = function() {
	evento_txt.text = "Release";
}

boton_btn.onReleaseOutside = function() {
	evento_txt.text = "Release Outside";
}
```

Ejercicio on_AS3.fla:

Con el mismo escenario del ejercicio anterior usamos el siguiente código actionscript, el cual sólo puede ser asociado a los frames:

```
boton_btn.addEventListener(MouseEvent.MOUSE_OVER, loadOver);
boton_btn.addEventListener(MouseEvent.MOUSE_OUT, loadOut);
boton_btn.addEventListener(MouseEvent.MOUSE_DOWN, loadDown);
boton_btn.addEventListener(MouseEvent.MOUSE_UP, loadUp);

function loadOver(event:MouseEvent):void {
	evento_txt.text = "Roll Over";
}

function loadOut(event:MouseEvent):void {
	evento_txt.text = "Roll Out";
}

function loadDown(event:MouseEvent):void {
	evento_txt.text = "Press";
}

function loadUp(event:MouseEvent):void {
	evento_txt.text = "Release";
```

}

En AS3 no disponemos del evento ReleaseOutside.

b. Evento "onClipEvent()"- (AS2)

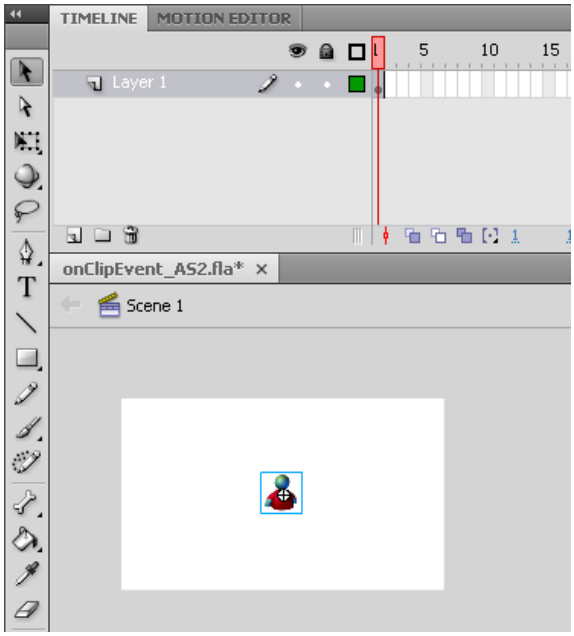
Sintaxis

onClipEvent(*movieEvent*){
sentencia(s);
}

Al igual que *on()*, se utiliza para responder a un evento, ya sea desencadenado por el usuario (*mouseMove*, *mouseDown*, etc.) o por el propio clip de Película (*load*, *enterFrame*, etc.). La principal diferencia entre *on* y *onClipEvent* es que el primero se utiliza en instancias de Botón mientras el segundo se utiliza para instancias de Clip de Película; también cambian los eventos ante los que reaccionan uno y otro.

Ejercicio onClipEvent_AS2.fla:

Para este ejercicio haremos un clip de película que permita reposicionar un objeto con el mouse. Colocamos en el escenario un clip de película como muestra la siguiente figura y le damos nombre de instancia "logo_mc".



onClipEvent es una función que sólo puede ser asociada a instancias de clipsd de película, no puede ser asociada a fotogramas, en nuestro ejercicio usamos el siguiente código:

```
onClipEvent(mouseDown){
        Mouse.hide();
        startDrag(this, true);
}

onClipEvent(mouseUp){
        Mouse.show();
        stopDrag();
}
```

Ejercicio onClipEvent_AS3.fla:

Con el mismo escenario del ejercicio anterior, colocamos una nueva capa y colocamos un clip de película con alfa=0, que contenga un recuadro que cubra todo el escenario, le damos nombre de instancia "box" y en el fotograma 1 asociamos el siguiente código:

```
box.addEventListener(MouseEvent.MOUSE_DOWN, loadStartDrag);
box.addEventListener(MouseEvent.MOUSE_UP, loadStopDrag);

function loadStartDrag(event:MouseEvent):void {
        Mouse.hide();
        logo_mc.startDrag(true);
}

function loadStopDrag(event:MouseEvent):void {
        Mouse.show();
        logo_mc.stopDrag();
}
```

4. *Variables*

Una variable es un espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato soportado por el lenguaje. Una variable es representada y usada a través de una etiqueta (un nombre) que se le asigna o que ya viene predefinida.

a. Tipos de Datos Simples.

var x = 3;

Dado que x no se declaró con un tipo de datos estricto, el compilador no podrá determinar el tipo; para el compilador, la variable x puede tener un valor de cualquier tipo.

ActionScript 2.0 convierte siempre automáticamente los tipos de datos simples (como Boolean, Number, String, null o undefined) cuando una expresión requiere la conversión y las variables no se han introducido con tipos de datos estrictos.

b. (Tipo Estricto de Datos (Strict Data Typing).

Los tipos de datos estrictos ofrecen diversas ventajas a la hora de compilar. La declaración de tipos de datos (tipos de datos estrictos) puede ayudar a evitar o diagnosticar errores existentes en el código al compilar. Para declarar una variable con un tipo de datos estricto, utilice el siguiente formato:
var variableName:datatype = valor;
Los tipos de datos estrictos garantizan que no se asigne accidentalmente un tipo de valor incorrecto a una variable.
En Actionscript 3.0 las variables deben ser declaradas de forma estricta.

c. Ámbito de una variable.

El ámbito de una variable se refiere al área en la que se conoce la variable (área en la que está definida) y se puede hacer referencia a ella. El área en la que se conoce la variable puede estar dentro de una determinada línea de tiempo o dentro de una función, o podría conocerse globalmente en toda la aplicación. Para más información sobre el ámbito, consulte Ámbito y referencias.

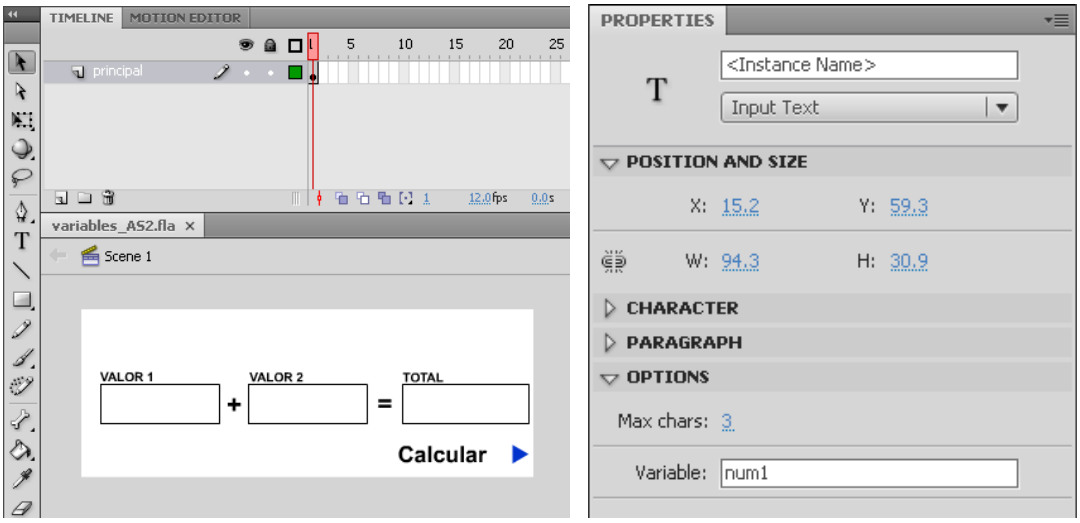
Comprender en qué consiste el ámbito de las variables es importante a la hora de desarrollar aplicaciones Flash con ActionScript. El ámbito no sólo indica cuándo y dónde puede hacer referencia a las variables, sino también el tiempo durante el cual una determinada variable existe en una aplicación. Al definir variables en el cuerpo de una función, éstas dejan de existir al finalizar la función especificada. Si intenta hacer referencia a objetos de un ámbito erróneo o a variables que han caducado, aparecerán errores en los documentos de Flash que provocarán un comportamiento inesperado o que no esté disponible una determinada funcionalidad.

ActionScript contiene tres tipos de ámbitos de variable:

- Las funciones y las **Variables globales** son visibles para todas las líneas de tiempo y todos los ámbitos del documento. Por consiguiente, una variable global se define en todas las áreas del código.
- Las **Variables de línea de tiempo** están disponibles para cualquier script de dicha línea de tiempo.
- Las **Variables locales** están disponibles en el cuerpo de la función en la que se han declarado (incluidas entre llaves). Por consiguiente, las variables locales sólo se definen en una parte del código.

Ejercicio variables_AS2.fla:

En este ejercicio calcularemos la suma de dos números. Para esto colocamos en el escenario dos campos de entrada de texto y un campo de texto dinámico. Posteriormente editamos sus propiedades y les asociamos las variables “num1”, “num2” y “total” respectivamente.



Adicionalmente colocamos un botón y le asignamos directamente el código:

```
on(release) {
n1 = Number(num1);
n2 = Number(num2);
total = n1 + n2;
}
```

Ejercicio variables_AS3.fla:

En el escenario del ejercicio anterior eliminamos las variables asociadas a los campos y les damos los nombres de instancias “num1”, “num2” y “total”, respectivamente y el nombre “calcular” al botón. Finalmente usamos el siguiente código en el primer fotograma:

```
calcular.addEventListener(MouseEvent.MOUSE_UP, fcalcular);

function fcalcular(Event:MouseEvent):void {
var n1 = Number(num1.text);
var n2 = Number(num2.text);
total.text = n1 + n2;
}
```

5. *Sentencias de Decisión (Condicionales)*

a. Sentencia “if”

Calcula el resultado de una condición para determinar la siguiente acción en un archivo SWF. Si la condición es true, Flash ejecuta las sentencias entre llaves ({}) que van a continuación de la condición. Si la condición es false, Flash omite las sentencias

entre llaves y ejecuta las sentencias que siguen a dichas llaves. Utilice la acción **if** para crear lógica de ramificación en sus scripts.

b. Sentencia "else"

Especifica las sentencias que deben ejecutarse si la condición de la sentencia if devuelve el valor false.

```
if (condición){
  sentencia(s)…;
} else (condición){
  sentencia(s)…;
}
```

c. Sentencia "else if"

Calcula el valor de una condición y especifica las sentencias que deben ejecutarse si la condición de la sentencia if inicial es false. Si la condición **else if** es true, el intérprete de Flash ejecuta las sentencias que van después de la condición entre llaves ({}). Si la condición **else if** es false, Flash pasa por alto las sentencias entre llaves y ejecuta las sentencias que van después de las llaves. Utilice la acción **else if** para definir lógica de ramificación en los scripts.

Sintaxis:

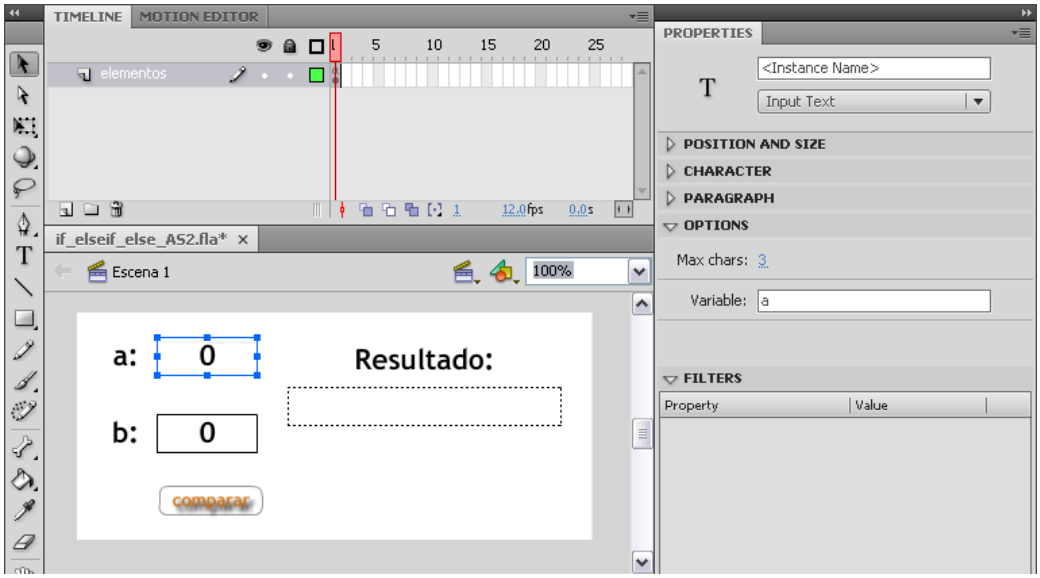
```
if (condición){
  sentencia(s)…;
} else if(condición){
  sentencia(s)…;
}else{
   sentencias(s)…;
}
```

Ejemplo: La condición entre paréntesis comprueba si la variable name contiene el valor literal "Erica". Si es así, se ejecuta la acción play() entre llaves.

```
if(name = = "Erica"){
  play();
}
```

Ejercicio if_elseif_else_AS2.fla:

Para este ejercicio haremos un clip de película que permita realizar la comparación de dos números cualesquiera. Colocamos en el escenario dos campos de entrada texto y un campo de texto dinámico, seleccionando cada campo les asociamos una variable a cada campo, la variable "a", "b" y "resultado". Posteriormente colocamos un botón con el nombre de instancia "comparar".



Finalmente colocamos el siguiente código en el primer fotograma:

```
comparar.onRelease = function() {
	a = Number(a);
	b = Number(b);
	if(a > b){
		resultado = "\"a\" es MAYOR que \"b\"";
	}else if(a < b){
		resultado = "\"a\" es MENOR que \"b\"";
	}else{
		resultado = "\"a\" es IGUAL a \"b\"";
	}
}
```

Ejercicio if_elseif_else_AS3.fla:

Con el mismo escenario anterior y eliminando las variables que se asociaron a cada campo, el código del primer fotograma es:

```
comparar.addEventListener(MouseEvent.MOUSE_UP, comparacion);

function comparacion(event:MouseEvent):void {
	var numa:Number=Number(a.text);
	var numb:Number=Number(b.text);
	if (numa > numb) {
		resultado.text = "\"a\" es MAYOR que \"b\"";
	}
	else if (numa < numb) {
		resultado.text = "\"a\" es MENOR que \"b\"";
	}
```

```
        else {
                resultado.text = "\"a\" es IGUAL a \"b\"";
        }
}
```

d. Sentencia "switch"

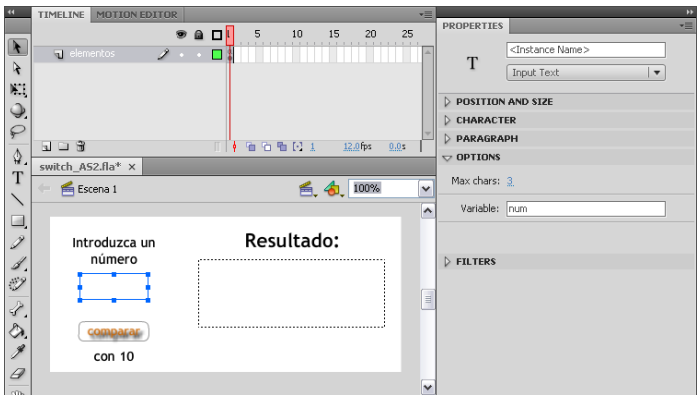
Crea una estructura ramificada para sentencias de ActionScript. Al igual que la acción if, la acción **switch** prueba una condición que ejecuta sentencias si la condición devuelve el valor true.

Ejemplo: Si el parámetro *number* da como resultado 1, se ejecuta la acción trace() que sigue a case 1; si el parámetro *number* da como resultado 2, se ejecuta la acción trace() que sigue a case 2, y así sucesivamente. Si ninguna expresión case coincide con el parámetro *number*, se ejecuta la acción trace() que sigue a la palabra clave *default*:

```
switch (number) {
 case 1:
  trace ("case 1 tested true");
  break;
 case 2:
  trace ("case 2 tested true");
  break;
 default:
  trace ("no case tested true")
}
```

Ejercicio switch_AS2.fla:

Para este ejercicio realizaremos un clip de película que compare un número cualquiera contra el número 10. Colocamos un campo de entrada de texto y un campo de texto dinámico y les asociamos las variables "num" y "resultado". Posteriormente colocamos un botón con nombre de instancia "comparar".



En el primer fotograma colocamos el siguiente código asociado:

```
comparar.onRelease = function() {
   var compara:Boolean = Number(num) == 10;
   switch(compara) {
      case true:
         resultado = "El número es IGUAL a 10";
         break;
      case false:
         if (num > 10) resultado = "El número es MAYOR a 10";
         else if(num < 10) resultado = "El número es MENOR a 10";
         else resultado = "Introduzca un número";
         break;
       }
}
```

Ejercicio switch_AS3.fla:

Con el mismo escenario anterior y eliminando las variables que se asociaron a cada campo, el código del primer fotograma es:

```
comparar.addEventListener(MouseEvent.MOUSE_UP, comparacion);

function comparacion(event:MouseEvent):void {
   var numero:String = num.text;
   if (numero == "") numero = "nada";
   var compara:Boolean = Number(numero) == 10;
   switch(compara) {
      case true:
         resultado.text = "El número es IGUAL a 10";
         break;
      case false:
         if (Number(numero)>10) resultado.text = "El número es MAYOR que 10";
         else if (Number(numero)<10) resultado.text="El número es MENOR que 10";
         else resultado.text = "Introduzca un número";
         break;
}
}
```

6. Funciones

Una función es un conjunto de sentencias agrupadas y ordenadas para realizar una determinada tarea. La acción *return* en las sentencias de una función se utiliza para hacer que la función devuelva o genere un valor.

a. Funciones anónimas

Como su nombre lo indica no son funciones definidas o "declaradas" y pueden utilizarse en expresiones o para escribir métodos en objetos.

Ejercicio función_anonima_AS2.fla:

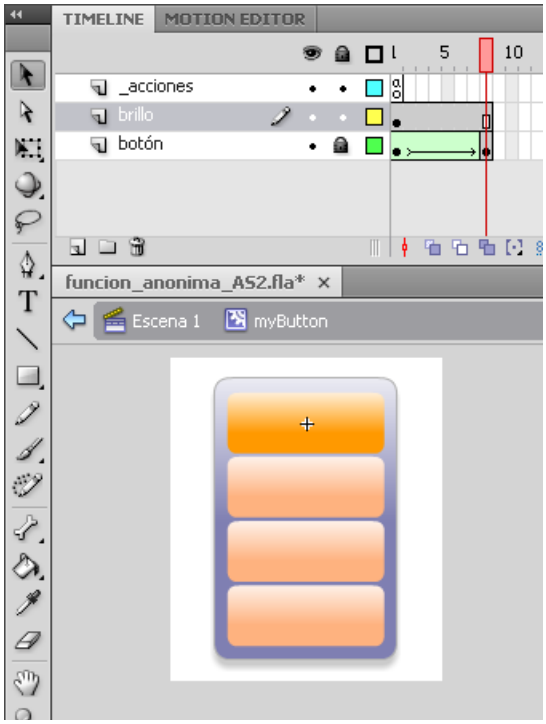
En este ejercicio comenzaremos la creación de un menú de cuatro botones. Para ésto creamos un clip de película que reusaremos con varias instancias que conformen el menú. Lo iniciamos creando un recuadro de color rojo y lo convertimos en el clip de película myButton y recrearemos el comportamiento de los estados editando el clip y generando una interpolación de forma sobre el recuadro que le cambie el color en el fotograma 8. En una capa superior le colocamos un brillo desde el borde superior. Posteriormente en el escenario principal colocaremos sobre un fondo cuatro instancias de este clip de película para crear un menú de cuatro botones.

En este punto colocamos en el primer fotograma de la línea de tiempo del clip de película el siguiente código que contiene dos funciones anónimas para sobrescribir los eventos del botón:

```
stop();

this.onRollOver = function(){
this.onEnterFrame = function(){
        if(_currentframe < _totalframes){
                nextFrame();
        }else{
                delete(this.onEnterFrame);
        }
}
}

this.onRollOut = function(){
this.onEnterFrame = function(){
        if(_currentframe > 1){
                prevFrame();
        }else{
                delete(this.onEnterFrame);
        }
}
}
```



Ejercicio función_anonima_AS3.fla:

Con el mismo escenario del ejercicio anterior las acciones dentro del clip de película son:

```
stop();

this.addEventListener(MouseEvent.MOUSE_OVER, buttonOver);
this.addEventListener(MouseEvent.MOUSE_OUT, buttonOut);

function buttonOver(event:MouseEvent):void {
this.addEventListener(Event.ENTER_FRAME, moveOver);
}

function buttonOut(event:MouseEvent):void {
this.addEventListener(Event.ENTER_FRAME, moveOut);
}

function moveOver(event:Event):void {
if (currentFrame < totalFrames) {
        nextFrame();
}
else {
        removeEventListener(Event.ENTER_FRAME, moveOver);
}
```

```
}

function moveOut(event:Event):void {
removeEventListener(Event.ENTER_FRAME, moveOver);
if (currentFrame > 1) {
        prevFrame();
}
else {
        removeEventListener(Event.ENTER_FRAME, moveOut);
}
}
```

b. Funciones Definidas por el Usuario

Sintaxis:

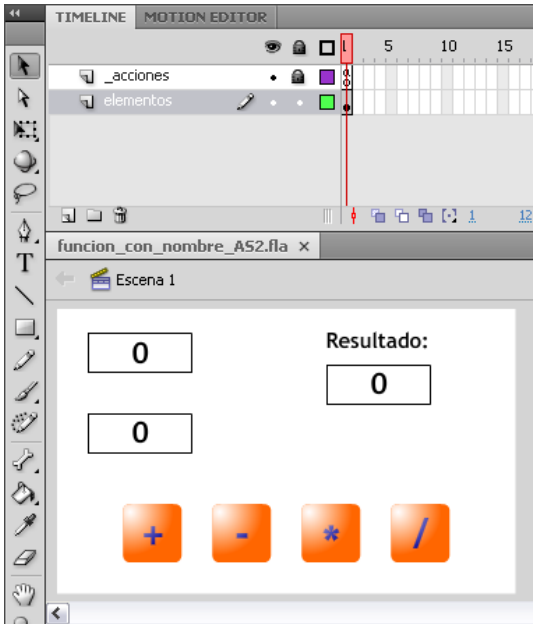
function *nombreDeFunción* ([*parámetro0, parámetro1,... parámetroN*]){
sentencia(s)
}

function ([*parámetro0, parámetro1,... parámetroN*]){
sentencia(s)
}

Son funciones *declaradas* o definidas con un nombre específico en una única ubicación lo cual les permite ser llamadas o invocadas desde diferentes scripts de un archivo SWF. Cuando se define una función, también se le puede especificar sus parámetros. Los parámetros son marcadores de posición para valores en los que opera la función. Puede pasar diferentes parámetros a una función cada vez que la llama. Esto permite volver a utilizar una función en muchas situaciones diferentes.

Ejercicio funcion_con_nombre_AS2.fla:

En este ejercicio simularemos una calculadora básica. Colocamos en el escenario dos campos de entrada de texto y un campo de texto dinámico y les damos los nombres de instancia “num1_txt”, “num2_txt” y “total_txt” respectivamente. También colocamos cuatro botones con los símbolos de suma, resta, multiplicación y división y les asignamos los nombres de instancia “sumar_btn”, “restar_btn”, “multiplicar_btn” y “dividir_btn”.



Finalmente en el primer fotograma colocamos el código que contiene la función “operar”:

```
num1_txt.restrict = "0-9";
num2_txt.restrict = "0-9"

function operar(num1:Number, num2:Number, operacion:String):Number{
 switch (operacion) {
            case "sumar": {
                        return(Number(num1) + Number(num2));
            }
            case "restar": {
                        return(num1 - num2);
            }
            case "multiplicar": {
                        return(num1 * num2);
            }
            case "dividir": {
                        return(num1 / num2);
            }
            default: {
                        return 0;
            }
}
}

sumar_btn.onRelease = function(){
 total_txt.text = operar(num1_txt.text, num2_txt.text, "sumar");
}
```

```
restar_btn.onRelease = function(){
 total_txt.text = operar(num1_txt.text, num2_txt.text, "restar");
}

multiplicar_btn.onRelease = function(){
 total_txt.text = operar(num1_txt.text, num2_txt.text, "multiplicar");
}

dividir_btn.onRelease = function(){
 total_txt.text = operar(num1_txt.text, num2_txt.text, "dividir");
}
```

Ejercicio funcion_con_nombre_AS3.fla:

Con el mismo escenario del ejercicio anterior, en el primer fotograma colocamos el código que contiene la función “operar”:

```
num1_txt.restrict = "0-9";
num2_txt.restrict = "0-9";

function operar(num1:String, num2:String, tip:String):String {
	// funcion para realizar operacion de dos valores
	switch(tip) {
		case "sumar":
			return String(Number(num1) + Number(num2));
			break;
		case "restar" :
			return String(Number(num1) - Number(num2));
			break;
		case "multiplicar" :
			return String(Number(num1) * Number(num2));
			break;
		case "dividir" :
			return String(Number(num1) / Number(num2));
			break;
		default:
			return "0";
	}
}

sumar_btn.addEventListener(MouseEvent.MOUSE_UP, sumar);
restar_btn.addEventListener(MouseEvent.MOUSE_UP, restar);
multiplicar_btn.addEventListener(MouseEvent.MOUSE_UP, multiplicar);
dividir_btn.addEventListener(MouseEvent.MOUSE_UP, dividir);

function sumar(event:MouseEvent):void {
	total_txt.text = operar(num1_txt.text, num2_txt.text, "sumar");
```

```
}

function restar(event:MouseEvent):void {
	total_txt.text = operar(num1_txt.text, num2_txt.text, "restar");
}

function multiplicar(event:MouseEvent):void {
	total_txt.text = operar(num1_txt.text, num2_txt.text, "multiplicar");
}

function dividir(event:MouseEvent):void {
	total_txt.text = operar(num1_txt.text, num2_txt.text, "dividir");
}
```

7. Bucles (ciclos)

El Bucle es un tipo de sentencia que ejecuta un bloque de código repetidas veces (de ahí su nombre) mientras se cumpla la condición propuesta.

a. Bucle “for”

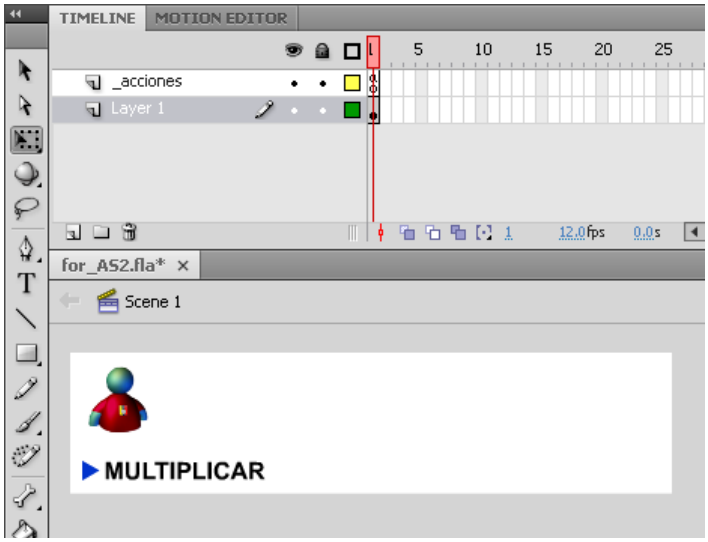
Sintaxis:

```
for(variable; condición; siguiente) {
  sentencia(s);
}
```

Permite realizar un bucle cierta cantidad de veces, determinada por los valores asignados para la variable (entre paréntesis en la sintaxis). Suele utilizarse sobre todo para trabajar con Arrays o con objetos numerados.

Ejercicio for_AS2.fla:

En este ejercicio crearemos copias de un clip de película al oprimir un botón. Colocamos en el escenario un clip de película y un botón y les damos nombres de instancia “logo_mc” y “multiplicar_btn”, respectivamente, como se muestran en la siguiente figura.



Para multiplicar el clip colocamos el siguiente código en el primer fotograma:

```
multiplicar_btn.onRelease = function(){
	for(i=2; i <= 10; i++){
		duplicateMovieClip("logo_mc", "logo" + i + "_mc", i);
		var newLogo_mc:MovieClip = eval("logo" + i + "_mc");
		with(newLogo_mc){
			_x = (logo_mc._x * i);
			_rotation = i * 36;
			_alpha = (100 / i);
		}
	}
}
```

b. Bucle “while”

Sintaxis:

```
while(condición) {
 sentencia(s);
}
```

Comprueba el valor de una expresión y ejecuta una sentencia o una serie de sentencias indefinidamente mientras el valor de la expresión sea true.
Antes de que se ejecute el bloque de sentencia, se comprueba *condición*; si se obtiene como resultado el valor true, se ejecuta el bloque de sentencia. Si la condición es false, el bloque de sentencia se omite y se ejecuta la primera sentencia después de ejecutar el bloque de sentencia de la acción while.

c. Bucle “do while”

Sintaxis:

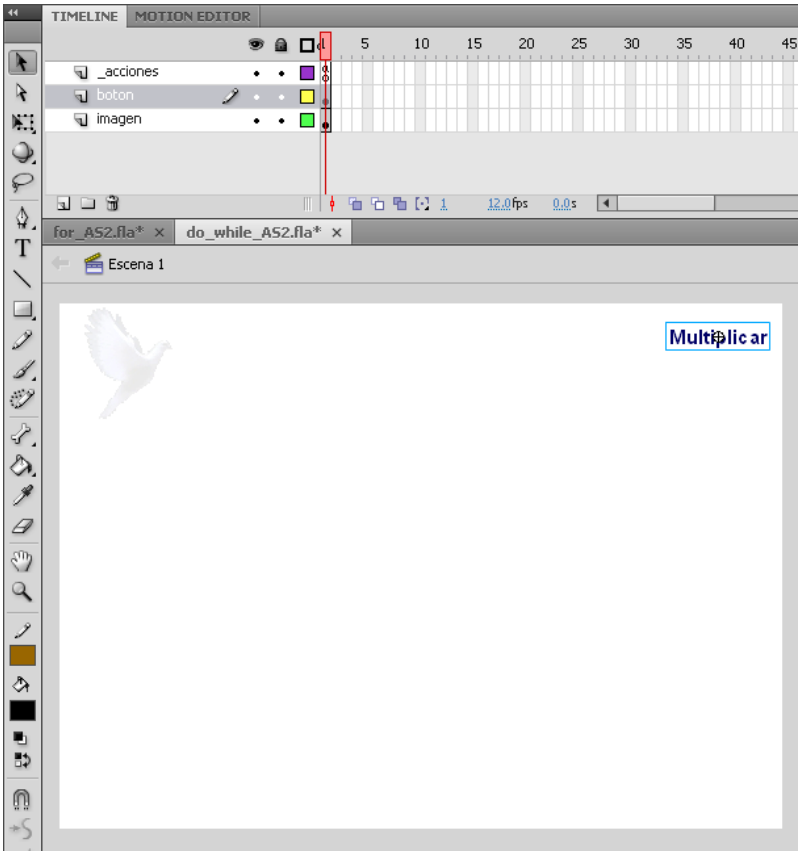
do {
sentencia(s);
} while (*condición*)

Ejecuta las sentencias y después comprueba la condición de una reproducción indefinida mientras la condición sea true. También es posible colocar la condición al principio, lo que hará que primero se compruebe la condición y, en caso de ser true, se ejecuta(n) la(s) sentencia(s) dentro del bloque "do". Para tal caso la sintaxis sería como sigue.

do while(*condición*){
sentencia(s);
}

Ejercicio do_while_AS2.fla:

En este ejercicio crearemos duplicados de un clip con la sentencia DO WHILE. Colocamos en el escenario un clip de película y un botón como muestra la figura y les damos nombres de instancia "image_mc" y "multiplicar_btn" respectivamente.



Para crear los duplicados usamos el siguiente código en el primer fotograma:

```
function duplicarImagen(images:Number):Void{
 var i:Number = 1;
```

```
do{
        duplicateMovieClip("image_mc", "image" + i + "_mc", i);
        var newImage_mc:MovieClip = eval("image" + i + "_mc")
        with(newImage_mc){
                _y = image_mc._y * (i+1);
                _x = image_mc._x * (i+1);
                _alpha = image_mc._alpha * i;
        }

        i++;
}while(i < images);
}

multiplicar_btn.onRelease = function(){
duplicarImagen(8);
}
```

8. Arrays

a. ¿Qué es un Array?

La clase *Array* permite acceder a matrices y manipularlas. Una matriz es un objeto cuyas propiedades se identifican con un número que representa su posición en la matriz. Este número se denomina *índice*. Todas las matrices tienen base cero, lo que quiere decir que el primer elemento de la matriz es [0], el segundo elemento es [1] y así sucesivamente.

b. Creación de un Array

Para crear un objeto *Array*, utilice el constructor *new Array()* o el operador de acceso a matriz ([ ]).

Ejemplo 1:

En el ejemplo siguiente, se crea un *Array* llamado “mi_Array” y se llena con los meses del año:

```
var mi_Array:Array = new Array(“Enero”, “Febrero”, “Marzo”, “Abril”, “Mayo”, “Junio”, “Julio”, “Agosto”, “Septiembre”, “Octubre”, “Noviembre”, “Diciembre”);
```

También es posible crear el array y dejarlo vació para llenarlo luego, p.e.:

```
var mi_Array:Array = new Array(númeroDeElementos)
```

Si se coloca un número en *númeroDeElementos* se define de una vez cuántos elementos tendrá en Array, en caso de no querer determinarlo puede dejarse vacío, quedando: var mi_Array:Array = new Array().

Ejemplo 2:

En el ejemplo siguiente, my_Array contiene los meses del año.

```
mi_Array[0] = "Enero"
mi_Array[1] = "Febrero"
mi_Array[2] = "Marzo"
```

```
mi_Array[3] = "Abril"
```

Nótese que al llenar los datos, el *Array* se crea automáticamente.

c. Acceder a los Valores de un Array

Para acceder a los valores de un *Array*, utilice el operador de acceso a matriz ([ ]).

Ejemplo 1:

Utilizando el array del ejemplo anterior, colocamos en el panel de salida el valor del elemento 3 de mi_Array:

```
trace(mi_Array[3]);
```

El panel de salida mostrará: Abril

También podemos cambiar el valor del elemento:

```
mi_Array[3] = nuevoValor;
```

Ejemplo 2:

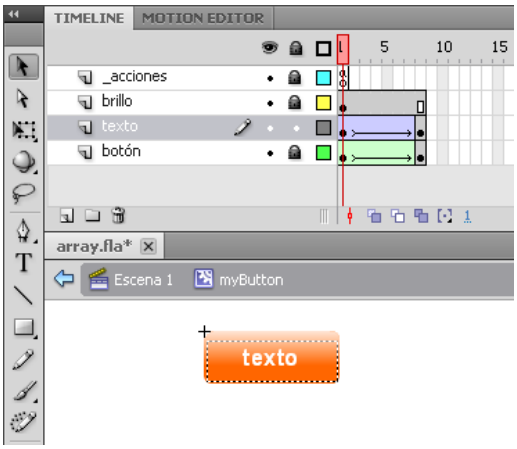
Uno de los usos más comunes de los Arrays es en los bucles. En este ejemplo utilizaremos un bucle *for* para mostrar en el panel de salida los valores del Array. Utilizaremos también la propiedad *length* para determinar la cantidad de elementos del Array:

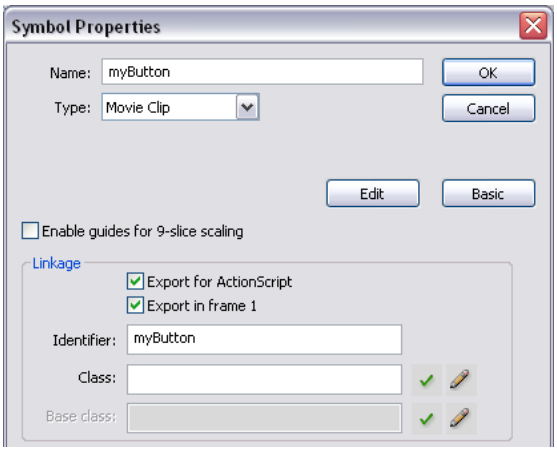
```
for(i = 0; i < mi_Array.length; i++){
trace(mi_Array[i]);
}
```

El panel de salida mostrará: Enero Febrero Marzo ...

Ejercicio array_AS2.fla:

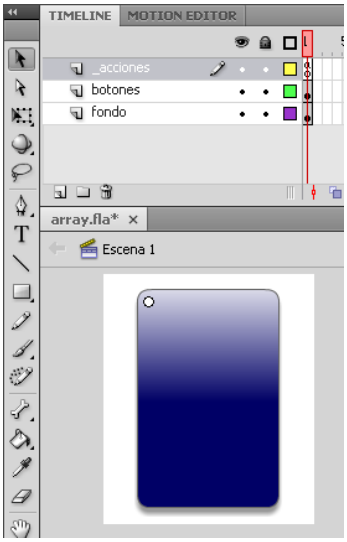
Para este ejercicio retomaremos el del menú de funciones anónimas para asignarle título a cada botón. Para ésto le agregaremos al clip “myButton” una capa para crear un clip animado que llamaremos “titulo” el cual contendrá un campo de texto dinámico que llamaremos “texto”. Le damos una interpolación de movimiento hasta el fotograma 8 para darle un efecto de color. Posteriormente seleccionamos a myButton en la biblioteca y abrimos sus propiedades y marcamos la opción “EXPORT FOR ACTIONSCRIPT”.





Como sabemos al momento de publicar un flash sólo se compilan los objetos localizados en el escenario dentro de todos los fotogramas. Para este ejercicio NO colocaremos el botón en el escenario pero configurándolo como exportable para actionscript tendremos acceso dinámico al botón.

Nuestro escenario inicial tendrá un fondo para el menú y sobre él un clip de película vacío con el nombre de instancia “contenedor”.



Finalmente en el primer fotograma colocamos el siguiente código:

```
var botones:Array = new Array(4);
var titulos:Array = new Array("Home", "Info", "Servicio", "Contacto");

for(var i:Number = 0; i < botones.length; i++){
    botones[i] = attachMovie("myButton", "b" + i + "_btn", getNextHighestDepth());
    if(i == 0){
        botones[i]._x = contenedor._x;
        botones[i]._y = contenedor._y;
    }
    else {
        botones[i]._x = b0_btn._x;
        botones[i]._y = botones[i-1]._y + botones[i]._height + 1;
    }
    botones[i].titulo.texto.text = titulos[i];
}
```

Ejercicio array_AS3.fla:

En el mismo escenario del ejercicio anterior, editamos el botón myButton y convertimos el brillo en clip de película para evitar un error en el clip del botón.

Realizado esto colocamos el siguiente código en el primer fotograma:

```
var botones:Array=new Array(4);
var titulos:Array=new Array("Home","Info","Servicio","Contacto");

for (var i:Number = 0; i < botones.length; i++) {
 var boton:myButton = new myButton();
 boton.name="boton"+i;
 if (i==0) {
          boton.x=contenedor.x;
          boton.y=contenedor.y;
 } else {
          boton.x=contenedor.x;
          boton.y=botones[i-1].y+boton.height+2;
 }
 boton.titulo.texto.text = titulos[i];
 botones[i]=addChild(boton);
 }
```

9. Carga de Archivos Externos

a. Cargar Variables Externas:

Flash permite la carga de variables guardadas en archivos externos a la película en tiempo de ejecución; lo que permite realizar ciertos cambios en la película. Esto puede utilizarse, por ejemplo, para cargar un cuadro de noticias que vaya actualizándose periódicamente o un conjunto de datos cambiantes, sin tener que volver a editar el archivo fuente y volver a publicarlo.

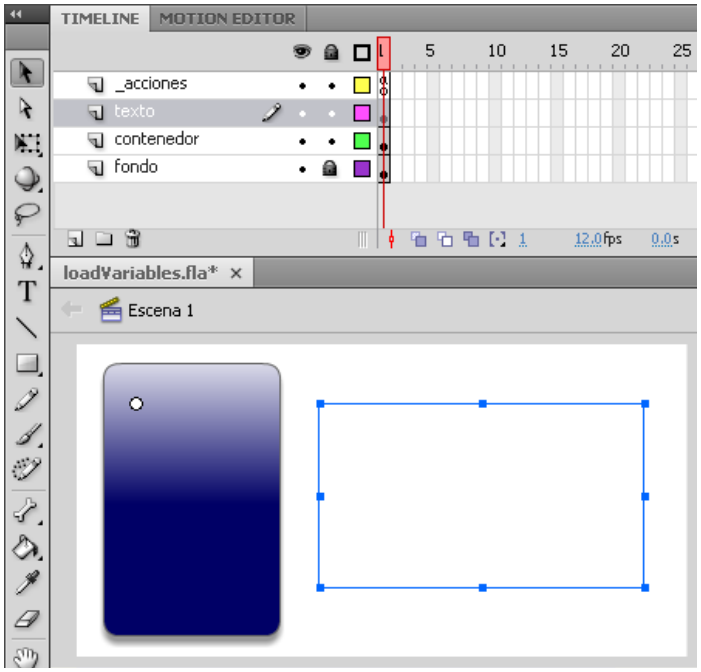
Para cargar variables externas puede utilizar la función *loadVariables()*, para cargar las variables en un clip de película de destino, o la función *loadVariablesNum()*, si desea cargarlas en un nivel específico.

Si desea tener un mayor control del proceso de carga, o enviar datos además de recibirlos, puede utilizar la clase *loadVars().*

Ejercicio loadVariables_AS2.fla:

Para este ejercicio usaremos el menú que hemos creado en el ejercicio de los arreglos con el fin de cargar el contenido de las secciones de dicho menú. Este contenido será texto en código HTML que se encuentra en cuatro variables declaradas dentro del archivo externo “variables.txt”. Mediante actionscript extraeremos sus valores indicando la ruta del archivo y los nombres de las variables.

Comenzamos colocando en el escenario el menú creado en el ejercicio de arreglos y adicionamos un campo de texto dinámico al cual le damos nombre de instancia “texto_txt”.



Ahora colocamos el código que sigue en el primer fotograma, este es el mismo del ejercicio de arreglos adicionando las líneas marcadas en negrita:

```
var botones:Array = new Array(4);
var titulos:Array = new Array("Home", "Info", "Servicio", "Contacto");
loadVariables("externos/variables.txt", _root);

for(var i:Number = 0; i < botones.length; i++){
        botones[i] = attachMovie("myButton", "b" + i + "_btn", getNextHighestDepth());
if(i == 0){
            botones[i]._x = contenedor._x;
            botones[i]._y = contenedor._y;
}
           else{
            botones[i]._x = b0_btn._x;
            botones[i]._y = botones[i-1]._y + botones[i]._height + 1;
}
botones[i].indice = i;
botones[i].onRelease = function(){
            texto_txt.htmlText = _root["text" + this.indice];
}
botones[i].caption_txt.text = titulos[i];
}
```

Ejercicio loadVariables_AS3.fla:

En el mismo escenario del ejercicio anterior colocamos el código que sigue en el primer fotograma, este es el mismo del ejercicio de arreglos con AS3 adicionando las líneas marcadas en negrita:

```
var botones:Array=new Array(4);
var titulos:Array=new Array("Home","Info","Servicio","Contacto");
var datos:URLVariables;
var myTextLoader:URLLoader = new URLLoader();
var num:Number;

myTextLoader.dataFormat = URLLoaderDataFormat.VARIABLES;

myTextLoader.addEventListener(Event.COMPLETE, onLoaded);

function onLoaded(e:Event):void {
 datos = e.target.data;
}

myTextLoader.load(new URLRequest("externos/variables.txt"));

for (var i:Number = 0; i < botones.length; i++) {
 var boton:myButton = new myButton();
 boton.name="boton"+i;
 if (i==0) {
          boton.x=contenedor.x;
          boton.y=contenedor.y;
 } else {
          boton.x=contenedor.x;
          boton.y=botones[i-1].y+boton.height+2;
 }
 boton.titulo.texto.text = titulos[i];
 botones[i]=addChild(boton);
 boton.addEventListener(MouseEvent.MOUSE_UP,root["load"+i]);
 }

function load0(ev:Event):void {
 texto_txt.htmlText = datos.text0;
}

function load1(ev:Event):void {
 texto_txt.htmlText = datos.text1;
}

function load2(ev:Event):void {
 texto_txt.htmlText = datos.text2;
}
```

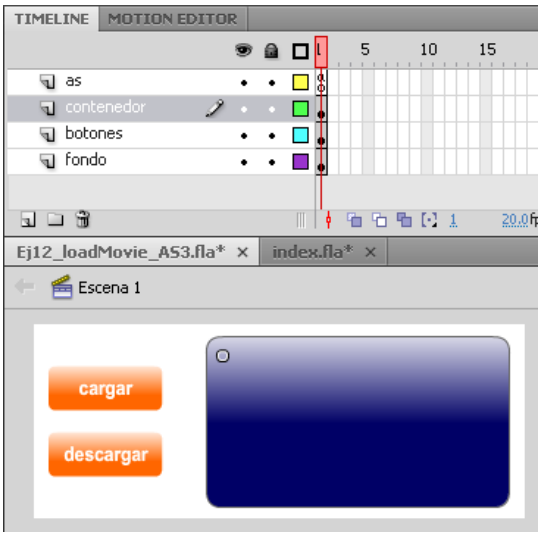
```
function load3(ev:Event):void {
texto_txt.htmlText = datos.text3;
}
```

b. Cargar Archivos Multimedia Externos:

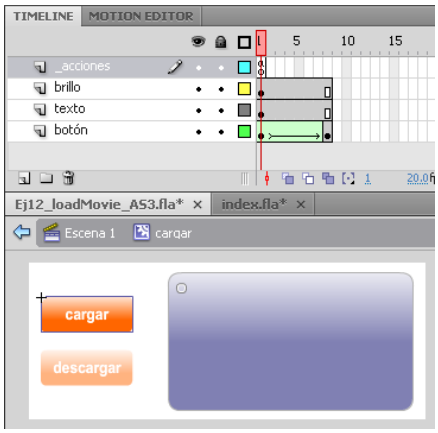
Existen cuatro tipos de archivos multimedia que pueden cargarse en una aplicación Flash en tiempo de ejecución: archivos SWF, MP3, JPEG y FLV. Flash Player puede cargar archivos multimedia externos desde cualquier dirección HTTP o FTP, desde un disco local utilizando una ruta relativa o mediante el protocolo file://.
Para cargar archivos SWF y JPEG externos, puede utilizar las funciones *loadMovie()* o *loadMovieNum()* o el método *MovieClip.loadMovie()*. Al cargar un archivo SWF o JPEG, debe especificarse un clip de película o nivel de película como destino de dichos archivos.
Para reproducir un archivo MP3 externo (MPEG capa 3), utilice el método *loadSound()* de la clase Sound. Este método permite especificar si el archivo MP3 se debe reproducir en flujo o descargar completamente antes de empezar a reproducirse.
Flash Video (FLV) es el formato de video nativo que utiliza Flash Player. Los archivos FLV se pueden reproducir a través de HTTP o desde el sistema de archivos local. La reproducción de archivos FLV externos ofrece varias ventajas frente a la incorporación de video en un documento de Flash, como por ejemplo mejor rendimiento y administración de la memoria y velocidades de fotogramas de vídeo y Flash independientes.

Ejercicio loadMovie.fla_AS2:

En este ejercicio haremos la carga y descarga de un clip de película externo. Para esto colocamos en el escenario dos clips de película que llamaremos “cargar_bt” y “descargar_bt” respectivamente. Colocamos un clip de fondo y sobre él un clip vacío al cual llamamos “contenedor”.



Podemos colocar una animación en la línea de tiempo de los movieclips que usamos como botones, agregándole el siguiente código actionsript en el primer frame como indica la figura.




```
stop();

this.onRollOver = function(){
	this.onEnterFrame = function(){
		if(_currentframe < _totalframes){
			nextFrame();
		}else{
			delete(this.onEnterFrame);
		}
	}
}

this.onRollOut = function(){
	this.onEnterFrame = function(){
		if(_currentframe > 1){
			prevFrame();
		}else{
			delete(this.onEnterFrame);
		}
	}
}
```

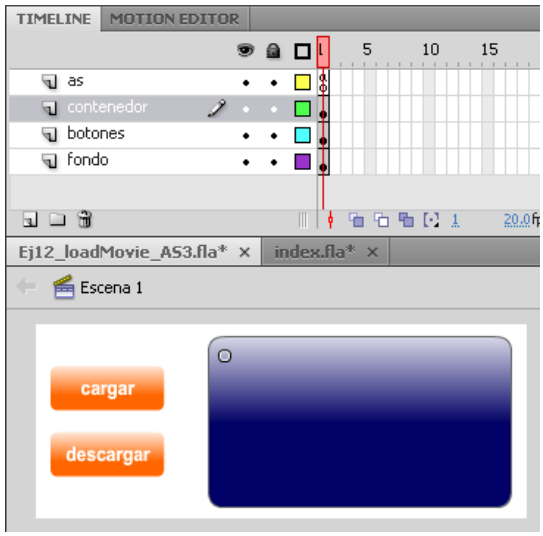
Finalmente colocamos el siguiente código en el primer fotograma de la escena 1:

```
cargar_bt.onRelease = function(){
    loadMovie("externos/pelicula.swf", contenedor);
}
```

```
descargar_bt.onRelease = function(){
        unloadMovie(contenedor);
}
```

Ejercicio loadMovie.fla_AS3:

Usando los mismos componentes del ejercicio en AS2, sólo debemos sustituir los códigos de los botones y de la escena 1 y colocar los que se indican a continuación de la figura.



Código actionscript para el primer frame de los movieclips usados como botones (cargar y descargar):

```
stop();

this.mouseEnabled = true;
this.buttonMode = true;

this.addEventListener(MouseEvent.MOUSE_OVER,turnOn);
this.addEventListener(MouseEvent.MOUSE_OUT,turnBack);

function turnOn(event:MouseEvent) {
        removeEventListener(Event.ENTER_FRAME, moveBack);
        this.addEventListener(Event.ENTER_FRAME, moveOn);
}

function turnBack(event:MouseEvent) {
        removeEventListener(Event.ENTER_FRAME, moveOn);
        this.addEventListener(Event.ENTER_FRAME, moveBack);
}
```

```
function moveOn(event:Event):void {
	if (currentFrame < totalFrames) {
		nextFrame();
	}
	else {
		removeEventListener(Event.ENTER_FRAME, moveOn);
	}
}

function moveBack(event:Event):void {
	if (currentFrame > 1) {
		prevFrame();
	}
	else {
		removeEventListener(Event.ENTER_FRAME, moveBack);
	}
}
```

Finalmente colocamos el siguiente código en el primer fotograma de la escena 1:

```
var miLoader:Loader = new Loader();

cargar_bt.addEventListener(MouseEvent.CLICK, cargarMC);
descargar_bt.addEventListener(MouseEvent.CLICK, descargarMC);

function cargarMC(miEvento:MouseEvent):void {
	miLoader.load(new URLRequest("externos/pelicula.swf"));
	contenedor.addChild(miLoader);
}

function descargarMC(miEvento:MouseEvent):void {
	contenedor.removeChild(miLoader);
}
```

10. Tween

Objetivo: Interpolación de objetos mediante ActionScript en lugar de Interpolación de Movimiento en Línea de Tiempo.

En este ejercicio veremos cómo realizar animaciones de movimiento mediante el comando Tween de ActionScript. Veremos un banner animado ejecutando de manera cíclica la disolución o desvanecimiento de las siguientes cuatro imágenes:

Imagen 1

Imagen 2

Imagen 3

Imagen 4

Las cuatro imágenes en jpg tienen las mismas dimensiones y deben ser importadas a la película de flash la cual mantendrá las dimensiones de dichas imágenes.

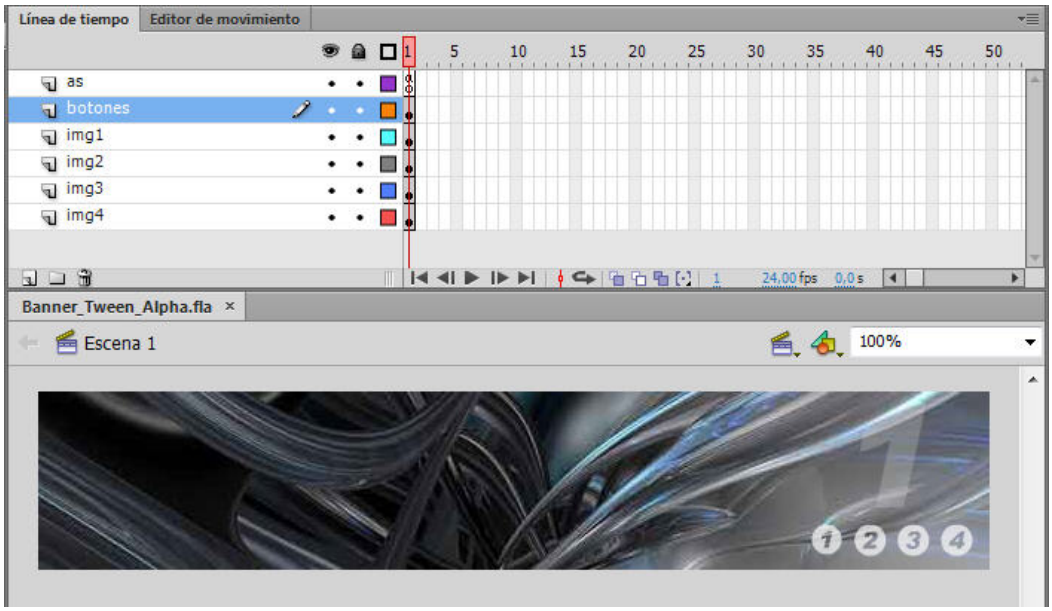


Fig 1. **Escena 1**

En la figura 1 vemos como las imágenes han sido importadas y colocadas en la misma posición en el escenario 1, se debe ubicar cada imagen en una capa y convertirla en símbolo tipo clip de película (movieclip) ya que les aplicaremos interpolaciones con ActionScript, y es necesario entonces que cada movieclip tenga un nombre de instancia único para poder animarlo. Tendremos entonces cuatro movieclips en el escenario con los nombres de instancia: **img1**, **img2**, **img3** e **img4**.

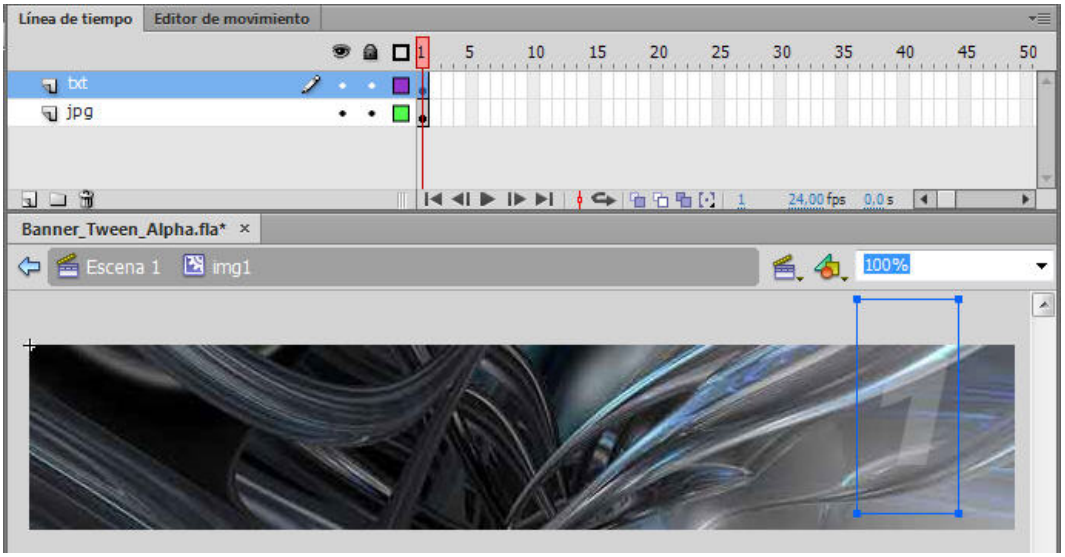


Fig 2. **Movlieclip img1**

La figura 2 muestra cómo podemos a su vez colocar textos o cualquier otro elemento dentro de cada movieclip, en este caso hemos agregado una numeración a cada imagen, se muestra el movieclip **img1** que además de la imagen contiene un campo de texto estático con el número correspondiente, análogamente se coloca cada numeración a los demás movieclips posicionados debajo de **img1**.

En la capa **botones** colocaremos cuatro símbolos tipo botón que utilizaremos para alterar el flujo normal de desvanecimiento mostrando la imagen que deseemos en cualquier instante de tiempo. La figura 3 muestra el botón **b1**, en su estado *Reposo* muestra un círculo al cual se le ha eliminado la figura central demarcada por el número 1 y se le ha reducido el valor *alpha* a un 70%, en el estado *Sobre* y *Presionado* se tiene el mismo objeto con *alpha* en 100% y la *zona activa* del botón será el círculo completo.

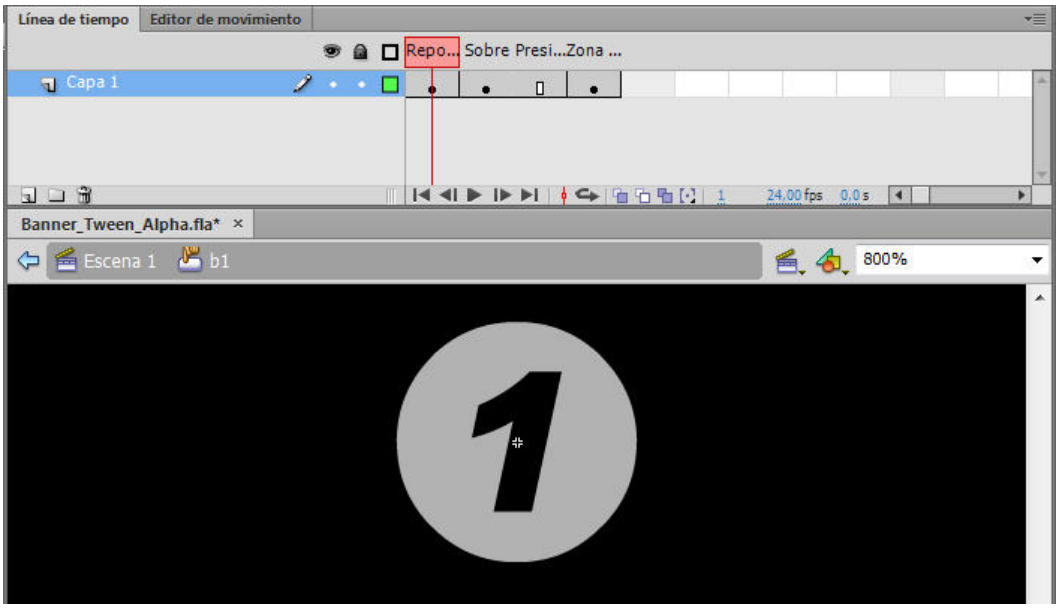


Fig 3. **Botón b1**

En la capa **as** colocaremos el código ActionScript.

En este ejercicio aplicaremos el comando **Tween** para el cual debemos indicar los siguientes parámetros o argumentos:

```
Tween(obj:Object, prop:String, func:Function, begin:Number, finish:Number,
      duration:Number, useSeconds:Boolean)
```

Argumentos:

- obj: nombre de instancia del símbolo a ser interpolado
- prop: propiedad a ser interpolada (Ej. alpha, x,y, scaleX, scaleY,etc.)
- func: aceleración (Ej. Circ, Quart, Cubic, Elastic, Bounce, Back, Lineal, etc.)
- begin: valor inicial de la propiedad

- finish: valor final de la propiedad
- duration: duración de la interpolación
- useSeconds: indica si es duración en segundos o en fotogramas

A su vez usaremos el comando **setTimeout** que detiene el llamado a una función una cantidad de tiempo indicada. Y finalmente el comando **clearTimeout** que anula la ejecución de un setTimeout.

Veamos ahora el ActionScript de nuestro ejercicio:

```
import fl.transitions.Tween;  // librerías de flash
import fl.transitions.easing.*;
import flash.events.MouseEvent;

var fade1:Tween; // declaración de variables tipo Tween
var fade2:Tween;
var fade3:Tween;
var fade4:Tween;

var sto:Number;  // variable para detener el flujo normal de la animación

// al cargarse el swf se muestra la foto 1

sto = setTimeout(foto2,4000); // se define el tiempo antes de llamar a la función foto2
// se asigna el valor a sto en caso de necesitar detener el flujo de la animación

function foto1():void { // función que desvanece las imagenes 2,3,4 y muestra la imagen 1
	fade1 = new Tween(img1,"alpha",Regular.easeInOut,img1.alpha,1,1,true);
	fade2 = new Tween(img2,"alpha",Regular.easeInOut,img2.alpha,0,1,true);
	fade3 = new Tween(img3,"alpha",Regular.easeInOut,img3.alpha,0,1,true);
	fade4 = new Tween(img4,"alpha",Regular.easeInOut,img4.alpha,0,1,true);
	sto = setTimeout(foto2, 4000); // se define el tiempo antes de llamar a la función foto2
}

function foto2():void { // función que desvanece las imagenes 1,3,4 y muestra la imagen 2
	fade1 = new Tween(img1,"alpha",Regular.easeInOut,img1.alpha,0,1,true);
	fade2 = new Tween(img2,"alpha",Regular.easeInOut,img2.alpha,1,1,true);
	fade3 = new Tween(img3,"alpha",Regular.easeInOut,img3.alpha,0,1,true);
	fade4 = new Tween(img4,"alpha",Regular.easeInOut,img4.alpha,0,1,true);
	sto = setTimeout(foto3, 4000); // se define el tiempo antes de llamar a la función foto3
}

function foto3():void { // función que desvanece las imagenes 1,2,4 y muestra la imagen 3
	fade1 = new Tween(img1,"alpha",Regular.easeInOut,img1.alpha,0,1,true);
	fade2 = new Tween(img2,"alpha",Regular.easeInOut,img2.alpha,0,1,true);
	fade3 = new Tween(img3,"alpha",Regular.easeInOut,img3.alpha,1,1,true);
	fade4 = new Tween(img4,"alpha",Regular.easeInOut,img4.alpha,0,1,true);
```

```
	sto = setTimeout(foto4, 4000); // se define el tiempo antes de llamar a la función foto4
}

function foto4():void { // función que desvanece las imagenes 1,2,3 y muestra la imagen 4
	fade1 = new Tween(img1,"alpha",Regular.easeInOut,img1.alpha,0,1,true);
	fade2 = new Tween(img2,"alpha",Regular.easeInOut,img2.alpha,0,1,true);
	fade3 = new Tween(img3,"alpha",Regular.easeInOut,img3.alpha,0,1,true);
	fade4 = new Tween(img4,"alpha",Regular.easeInOut,img4.alpha,1,1,true);
	sto = setTimeout(foto1, 4000); // se define el tiempo antes de llamar a la función foto1
}

// BOTONES

b1.addEventListener(MouseEvent.MOUSE_UP, f1); // activación del boton b1
b2.addEventListener(MouseEvent.MOUSE_UP, f2); // activación del boton b2
b3.addEventListener(MouseEvent.MOUSE_UP, f3); // activación del boton b3
b4.addEventListener(MouseEvent.MOUSE_UP, f4); // activación del boton b4

function f1(miEvento:MouseEvent):void {
	// función que detiene el flujo de animación normal y carga la imagen 1
	clearTimeout(sto);
	foto1();
}
function f2(miEvento:MouseEvent):void {
	// función que detiene el flujo de animación normal y carga la imagen 2
	clearTimeout(sto);
	foto2();
}
function f3(miEvento:MouseEvent):void {
	// función que detiene el flujo de animación normal y carga la imagen 3
	clearTimeout(sto);
	foto3();
}
function f4(miEvento:MouseEvent):void {
	// función que detiene el flujo de animación normal y carga la imagen 4
	clearTimeout(sto);
	foto4();
}
```

Al iniciarse la película estará visible la imagen 1, posteriormente las siguientes en numeración y las transiciones de desvanecimiento se realizarán cada 4 segundos.

11. Parent

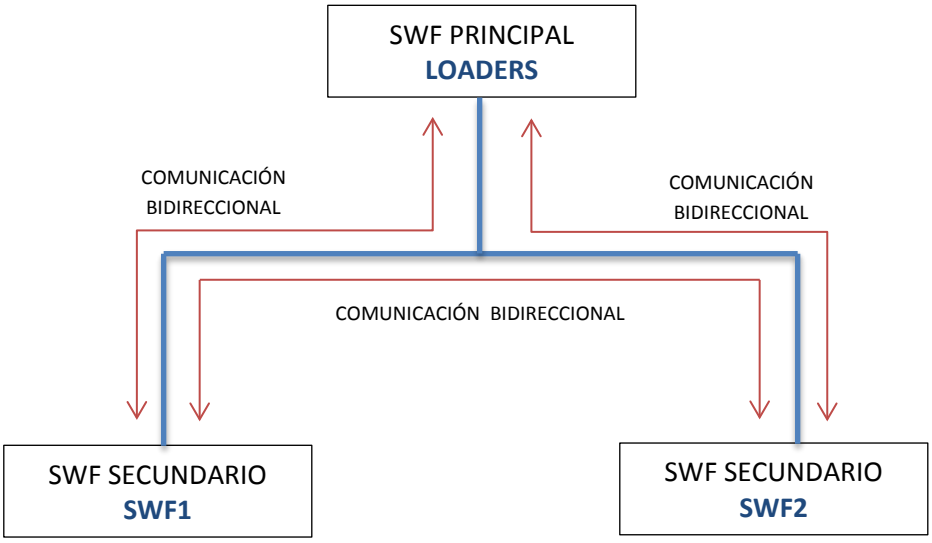
Objetivo: Comunicación entre diferentes películas de Flash.

Con este ejercicio veremos cómo se puede hacer interactuar diversas películas de Flash, utilizando los comandos de ActionScript que permiten la comunicación entre películas. Sabemos que el comando loadMovie nos permite cargar una película de flash dentro de otra, de ésta manera podemos crear una estructura de SWF´s (principal y secundarios), dentro de la cual la comunicación será siempre de modo vertical entre un SWF padre y un SWF hijo, ya sea en sentido ascendente o descendente.

Para la comunicación con un MovieClip superior o padre (comunicación ascendente), utilizaremos el comando "**MovieClip(parent[.parent])**", indicando tantos ".parent" como niveles necesitemos subir.

Para la comunicación con un sub-MovieClip (comunicación descendente), utilizaremos variables tipo **Object**, e internamente los nombres de instancia de los símbolos que tengamos definidos en nuestro flash.

Esta funcionalidad la crearemos mediante código ActionScript; preparemos previamente nuestros escenarios siguiendo la siguiente estructura de películas de flash:



Desarrollaremos entonces tres películas de flash : **LOADERS.FLA**, **SWF1.FLA** y **SWF2.FLA**.

La figura 1 a continuación muestra nuestro Flash principal llamado LOADERS.FLA en el fotograma inicial:

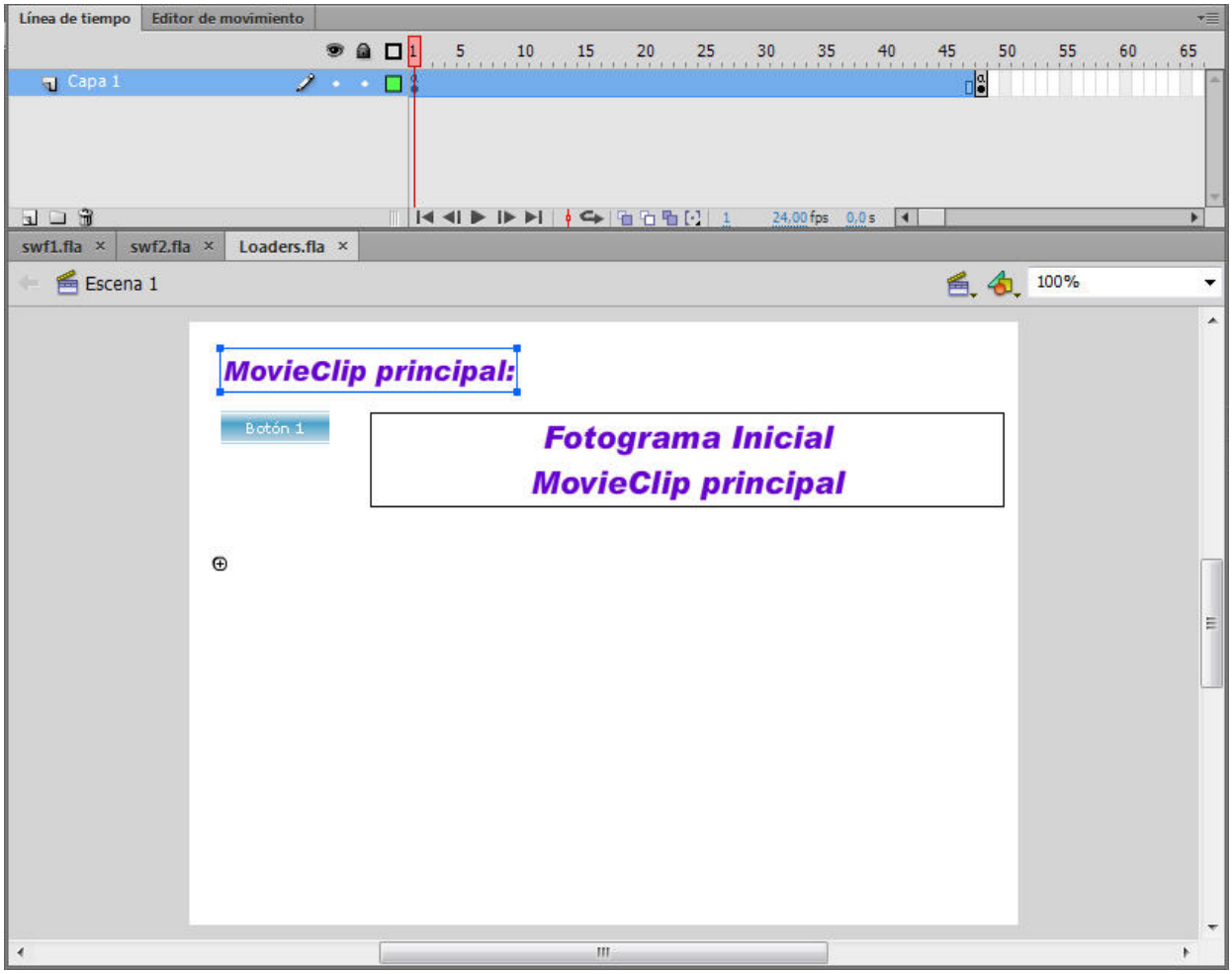


Fig. 1. **Loaders.fla (frame 1)**

Observemos que en el fotograma 1 se tiene únicamente una capa que sitúa en el escenario principal un campo de texto estático con el título: **MovieClip principal**. También un botón con el título **Botón 1** con nombre de instancia:"**boton**". Luego se tiene una caja de texto dinámico con nombre de instancia "**caja**" y su contenido: "**Fotograma Inicial \r MovieClip principal**". Inicialmente se detiene el flujo de la línea de tiempo colocando un comando **stop();** en el panel de acciones correspondiente al fotograma 1. Finalmente se ha colocado en la parte inferior de los anteriores elementos un clip de película vacío con el nombre de instancia "**cargador**".

Posteriormente se ha realizado un cambio en la caja de texto dinámico (**caja**) en el fotograma 48, como muestra la figura 2, colocando en la misma el texto: "**Fotograma Final \r MovieClip principal**", y análogamente se ha detenido el flujo de línea de tiempo mediante un segundo **stop();**. Para alcanzar este fotograma se debe restablecer el flujo en la línea de tiempo previamente detenido, mediante código ActionScript.

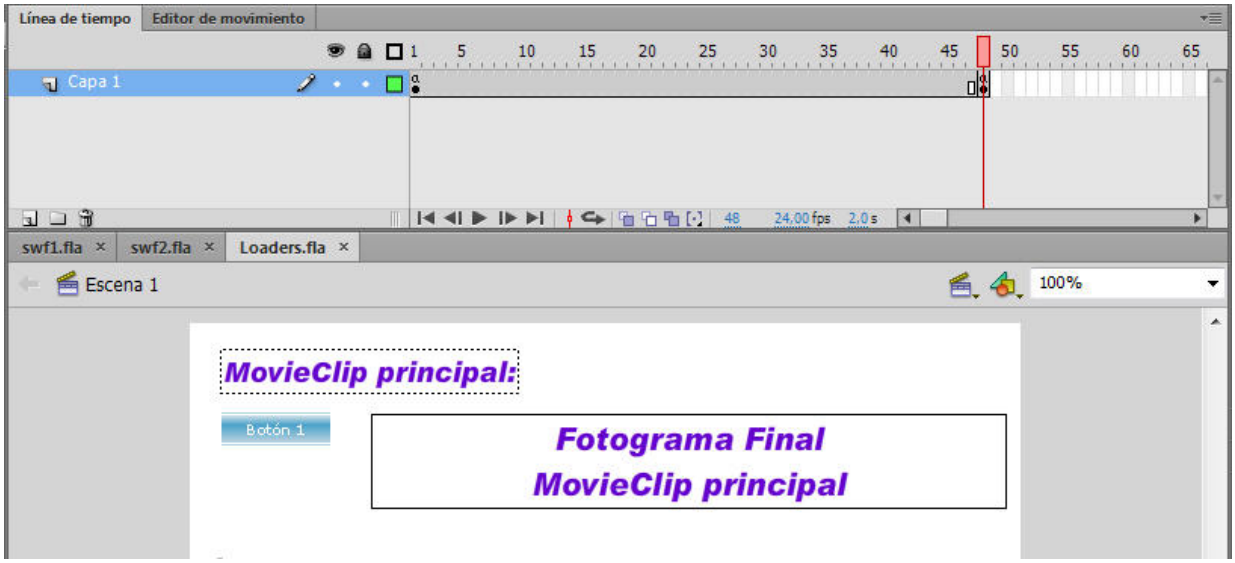


Fig. 2. **Loaders.fla (frame 48)**

Antes de estudiar el código ActionScript veremos la construcción de las películas de flash secundarias para una mejor comprensión del código.

La figura 3 muestra los elementos que componen la película secundaria SWF1 en el fotograma inicial.

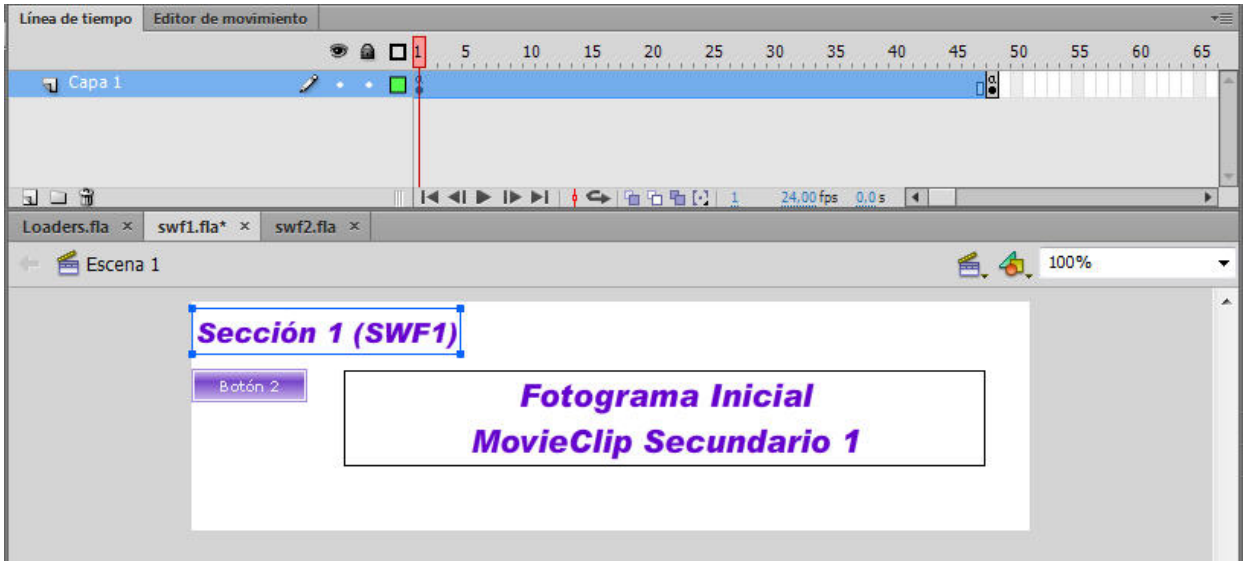


Fig. 3. **SWF1.fla (frame 1)**

En la misma se tiene una caja de texto estático con el título: "**Sección 1 (SWF1)**". También un botón 2 con el nombre de instancia: "**boton**", el cual a pesar de ser el mismo nombre que usamos para el botón de la película principal, no genera inconvenientes dado que pertenece a un ámbito

distinto. Y finalmente se tiene un campo de texto dinámico denominado **caja** con el texto: "**Fotograma Inicial \r Movieclip Secundario 1**".

En la figura 4 se muestra el cambio realizado en el fotograma 48 de la película SWF1 que consiste en modificar el texto del campo **caja** colocando lo siguiente: "**Fotograma Final \r Movieclip Secundario 1**".

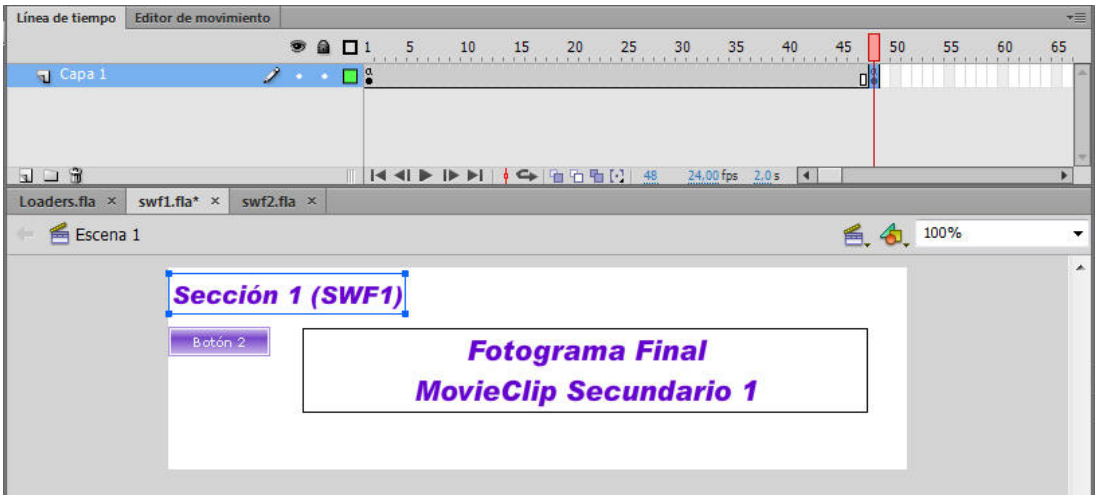


Fig. 4. **SWF1.fla (frame 48)**

Análogamente el flujo de tiempo de esta película será detenido tanto en el fotograma inicial (1) como en el fotograma final (48).

La siguiente figura muestra el fotograma inicial de la última película que se utilizará en este ejercicio, SWF2:

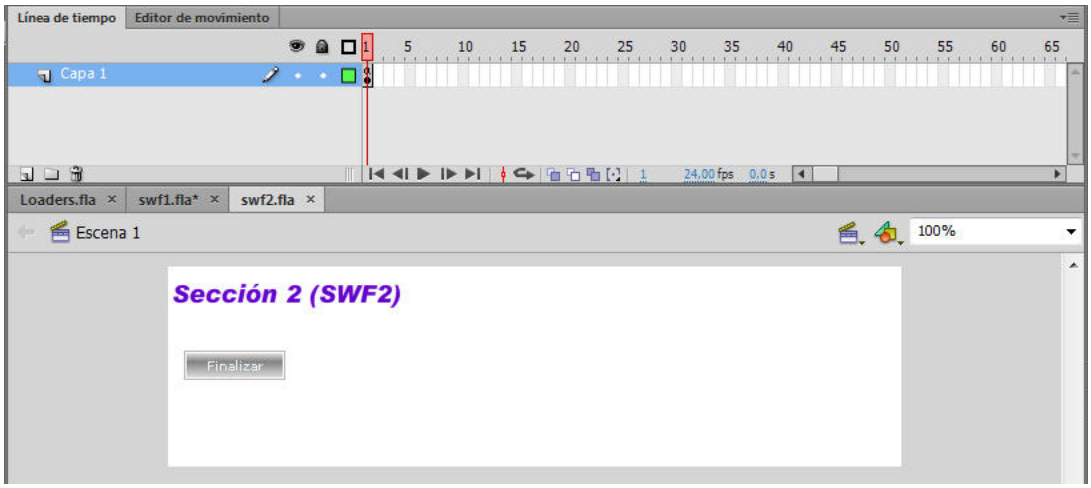


Fig. 5. **SWF2.fla (frame 1)**

En este único fotograma colocaremos un texto estático como título con el texto: "**Sección 2 (SWF2)**", y un tercer botón también con nombre de instancia: "**boton**".

Finalmente importaremos a la biblioteca de SWF2 un sonido en formato mp3 el cual se reproducirá mediante ActionScript en el fotograma inicial. La siguiente figura muestra el sonido importado, seleccionando este elemento en la biblioteca se debe incorporar el sonido al archivo SWF2.swf mediante sus propiedades, para esto presionamos el botón derecho del mouse sobre el elemento y seleccionamos “propiedades”.

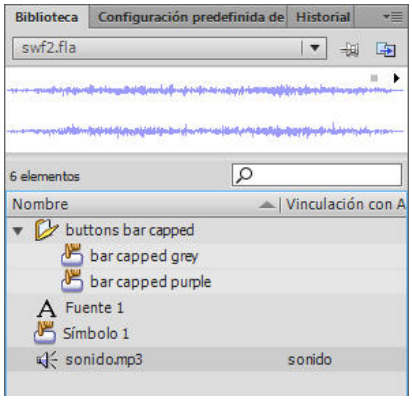


Fig. 6**. Biblioteca de SWF2**

La siguiente figura muestra las propiedades del archivo mp3 importado a la biblioteca y en esta ventana seleccionaremos la opción “Exportar para ActionScript”, automáticamente se seleccionará también la opción “Exportar en fotograma 1” la cual mantendremos seleccoinada. En el campo Clase definiremos el nombre de la clase que utilizaremos para reproducir el sonido en el escenario, inicialmente saldrá el nombre del archivo mp3 importado, lo modificaremos colocando un nombre más simple: “**sonido**”.

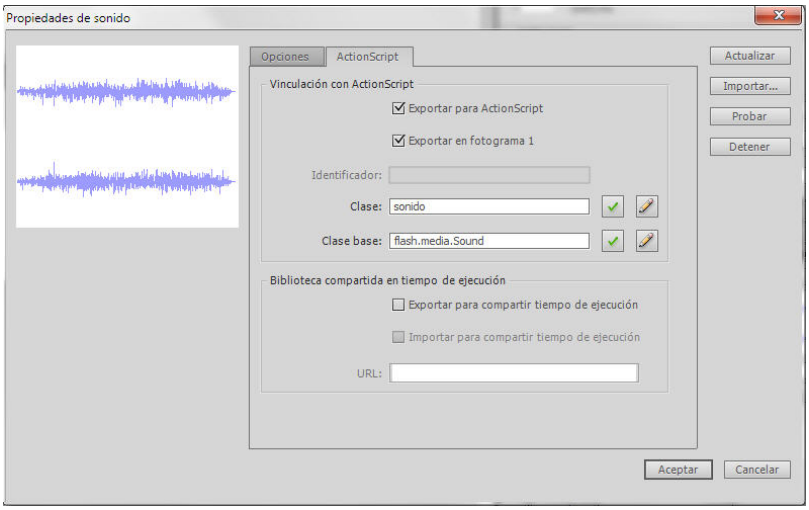


Fig. 7. **Propiedades del mp3**

En el campo Clase base dejaremos el que nos ha indicado Flash: “flash.media.Sound”, pues es la clase predefinida del programa para lograr la reproducción del sonido.

Veamos ahora los comandos de ActionScript de cada uno de estas películas de Flash que hemos construido.

ActionScript de Loaders.fla.

Mediante el siguiente código para el fotograma 1 efectuaremos las siguientes acciones :

• la carga de las dos películas secundarias (SWF1 y SWF2) dentro de la película principal (LOADERS). Con el objetivo de mostrar dos maneras diferentes, cargaremos la película SWF1 dentro del clip de película vacío denominado **cargador**, y la película SWF2 la cargaremos en pleno escenario principal y relocalizándola con sus propiedades de posicionamiento (X y Y).

• asignación de la funcionalidad del botón "**Boton 1**", la cual consiste en activar desde la película principal la línea de tiempo de la película secundaria SWF1, y adicionalmente la interrupción en la reproducción de un sonido cargado en la película secundaria SWF2.

Acciones del fotograma 1 de LOADERS:

```
import flash.display.LoaderInfo; // librerias de flash
import flash.display.Loader;
import flash.net.URLRequest;
import flash.events.MouseEvent;
import flash.events.Event;

stop(); // detiene el flujo de la línea de tiempo del movieclip principal

var mi_cargador1:Loader = new Loader(); // variables de carga para las películas secundarias
var mi_cargador2:Loader = new Loader();
var objeto:Object; // variables tipo object para la comunicación con un sub-movieclip
var objeto2:Object;

var mi_ruta1:URLRequest = new URLRequest("swf1.swf"); // variables URLRequest para establecer las
var mi_ruta2:URLRequest = new URLRequest("swf2.swf"); // rutas a los swf secundarios

mi_cargador1.load(mi_ruta1); // carga de los swf secundarios
mi_cargador2.load(mi_ruta2);

mi_cargador2.x = 300; // posicionamiento de la película swf2
mi_cargador2.y = 300;

objeto = cargador.addChild(mi_cargador1); // la variable objeto ahora direcciona a la película swf1
objeto2 = addChild(mi_cargador2); // la variable objeto2 direcciona a la película swf2

boton.addEventListener(MouseEvent.MOUSE_UP,llamar); // funcionalidad del boton Boton1

function llamar(miev:MouseEvent):void { // funcionalidad del boton BOTON1
```

```
        objeto.content.caja.text = "Se reproduce la linea de tiempo del SWF2"; // texto de la caja
        objeto.content.play(); // se activa el flujo de la linea de tiempo de la pelicula SWF1
        objeto2.content.micanal.stop(); // se detiene el sonido de la pelicula SWF2
}
```

Acciones del fotograma 48 de LOADERS:

```
stop(); // detiene el flujo de la línea de tiempo del movieclip principal
```

ActionScript de SWF1.fla.

Mediante el siguiente código para el fotograma 1 activaremos desde la película secundaria **SWF1** la línea de tiempo de la película principal **Loaders**.

Acciones del fotograma 1 de SWF1:

```
import flash.events.MouseEvent;  // librería de flash

stop(); // se detiene el flujo de línea de tiempo

boton.addEventListener(MouseEvent.MOUSE_UP,llamar);

function llamar(miev:MouseEvent):void { // funciones que activan el clip de película principal
        MovieClip(parent.parent.parent).caja.text = "Se reproduce la linea de tiempo\rdel MovieClip
         principal";
        MovieClip(parent.parent.parent).play();
}
```

Acciones del fotograma 48 de SWF1:

```
stop(); // detiene el flujo de la línea de tiempo del movieclip principal
```

ActionScript de SWF2.fla.

Mediante el siguiente código para el fotograma 1 le asignaremos la funcionalidad al botón de SWF2 la cual consiste en las dos siguientes acciones:

• activar desde la película SWF2 la línea de tiempo de la película secundaria SWF1.

• activar desde la película SWF2 la línea de tiempo de la película principal Loaders.

Acciones del fotograma 1 de SWF2:

```
import flash.events.MouseEvent; // librerías de flash
import flash.media.SoundChannel;

boton.addEventListener(MouseEvent.MOUSE_UP,llamar);

function llamar(miev:MouseEvent):void { // activación de los otros clips de película
        MovieClip(parent.parent).cargador.play(); // SWF2
        MovieClip(parent.parent).objeto.content.caja.text = "Finalizando..."; // Loaders
}

var mimusica:sonido = new sonido(); // se define variable de sonido extraída de la biblioteca

var micanal:SoundChannel = mimusica.play(10000); // se reproduce el sonido con variable de canal
```

Teniendo ya publicadas en una misma carpeta las tres películas que intercomunicaremos (Loaders, SWF1 y SWF2), ejecutamos la película principal Loaders y verificamos que la comunicación bidireccional efectivamente funcionará.

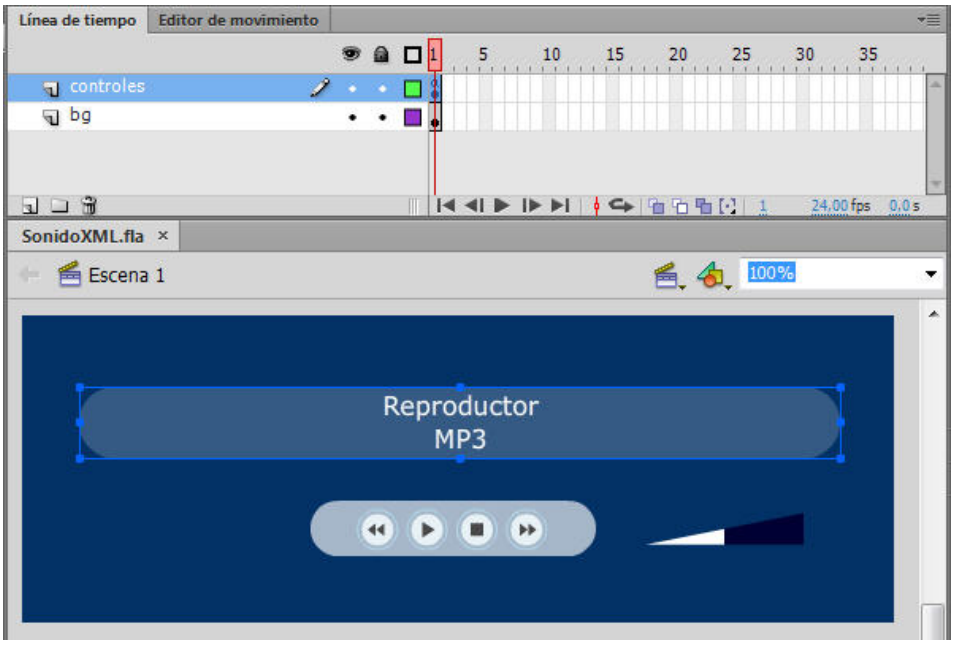
1. XML

Objetivo: Carga de archivos externos: Datos(XML) y Audio (MP3) . Creación de película de flash modificable sin reedición del archivo FLA.

Con este ejercicio veremos cómo se puede efectuar la carga dinámica de archivos de sonido mediante el uso de un documento XML, método recomendado cuando necesitamos comunicarnos con un SWF desde una página o aplicación web administrada con manejadores de contenido (CMS), o bien que pueda ser modificado sin tener necesariamente que editar el archivo FLA, simplemente modificar sus archivos asociados.

Al finalizar contaremos con un reproductor de archivos mp3 en la película de flash SonidoXML.swf, esta película será informada mediante un archivo XML del listado de canciones a ser reproducidas desde una carpeta que las contiene. Dicha carpeta deberá tener la misma de ubicación de los archivos SWF y XML.

Para comenzar se tiene en el escenario principal los controles comunes de un reproductor de sonido. También un campo de texto dinámico multilineal (opción que se selecciona en las propiedades del campo de texto) donde mostraremos el nombre de cada canción y su intérprete, y finalmente se tiene un movieclip elaborado como controlador de volumen.



Fi.1. **Escena 1**

La figura 1 muestra nuestro escenario principal. El campo de texto dinámico lleva por nombre de instancia “info”.

Seguidamente tenemos cuatro símbolos con los siguientes nombres de instancia y funciones:

• **rr**: botón para retroceder el reproductor a la canción anterior.

• **playpausa**: movieclip para pausar y recomenzar la reproducción de la música.

• **detener**: botón para detener la reproducción.

• **ff**: botón para adelantar el reproductor a la siguiente canción.

Los símbolos **rr**, **detener** y **ff** son del tipo botón, **playpausa** es tipo movieclip ya que el mismo elemento funcionará para pausar y reactivar el sonido, por lo que mostrará dos aspectos dependiendo del estado del reproductor como muestran las figuras a continuación.

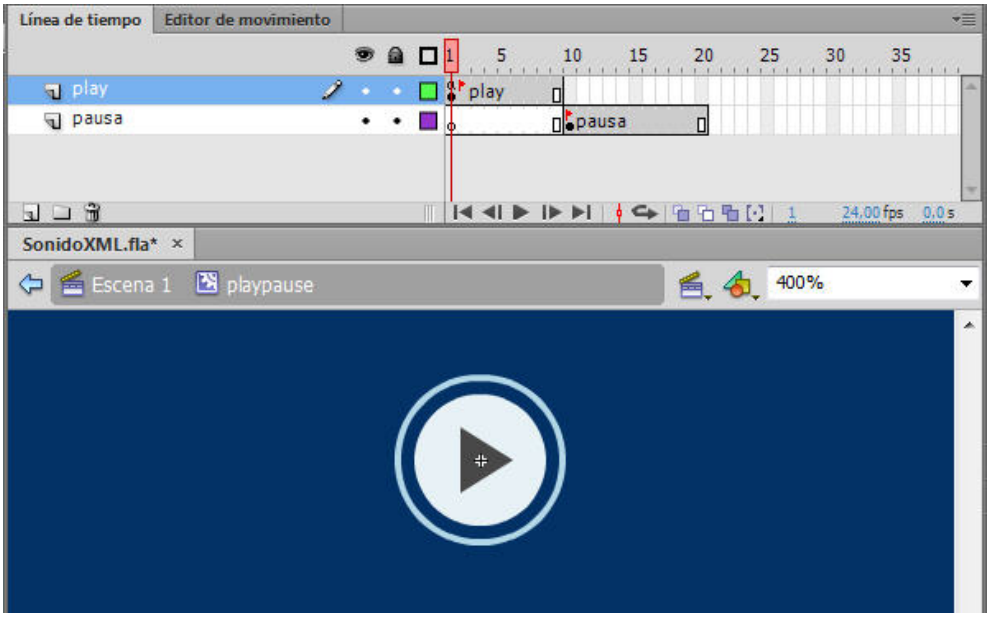


Fig. 2. **Movieclip playpause (frame 1)**

En la figura 2 vemos el contenido del movieclip **playpausa** en su fotograma inicial. Muestra un botón con el símbolo comúnmente usado para reproducir sonido. A este fotograma inicial le colocaremos una ***etiqueta*** para nombrar el momento de tiempo "**play**" como muestra la figura 3.

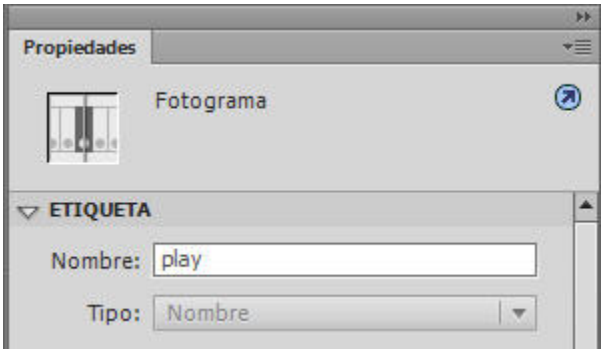


Fig. 3. **Propiedades del fotograma 1 del movieclip playpausa**

Posteriormente nos ubicamos en el fotograma 10 y cambiamos el símbolo del este escenario colocando en la misma posición otro elemento con el símbolo comúnmente usado para pausar una reproducción como vemos en la figura 4.

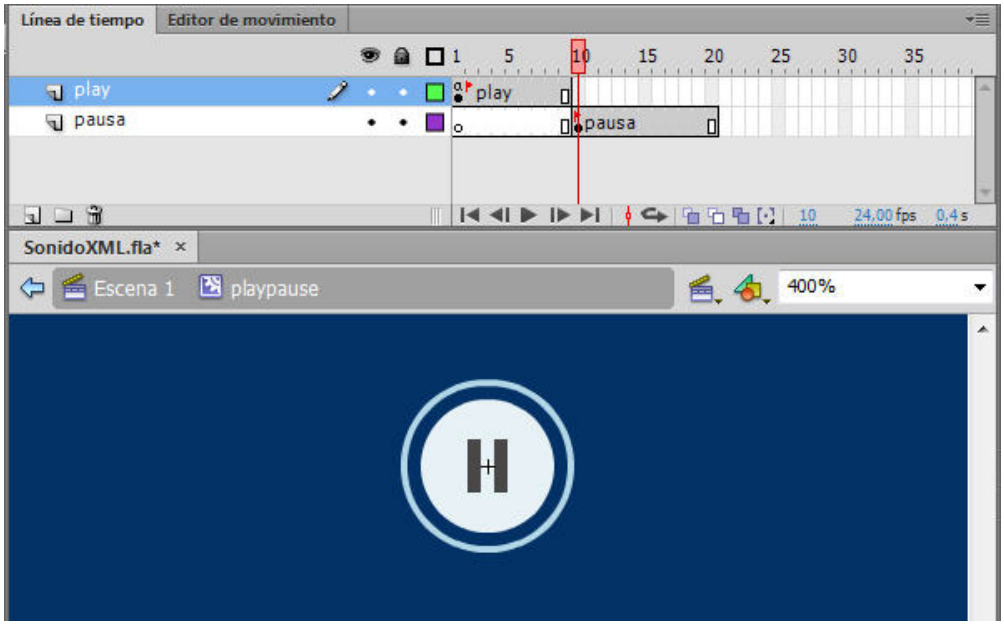


Fig. 4. **Movieclip playpause (frame 10)**

A este fotograma 10 le asignaremos también una etiqueta para identificar ese instante de tiempo como "**pausa**" como se muestra a continuación.

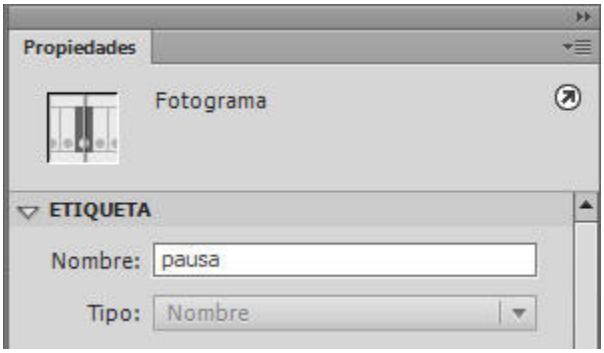


Fig. 5. **Propiedades del fotograma 10 del movieclip playpausa**

Mediante estas dos etiquetas nos desplazaremos del fotograma 1 al 10 y viceversa con el uso de ActionScript según se esté reproduciendo o no el sonido del reproductor.

Para el control de volumen se tiene un símbolo tipo movieclip con nombre de instancia "volumen", en él encontramos los elementos que presenta la figura 6 a continuación.

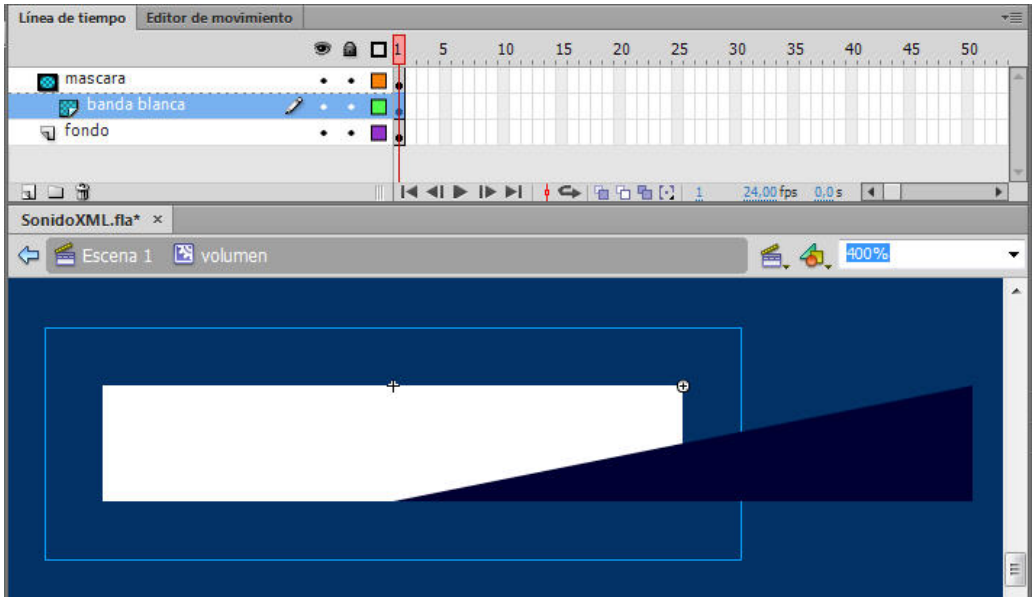


Fig. 6. **Movieclip volumen**

Dentro del movieclip **volumen** se tienen tres capas como muestra la figura. En la capa inferior se encuentra un triángulo con el objeto de indicar la magnitud del volumen, en la segunda capa un movieclip que contiene una barra rectangular de color blanco cuyo ***Punto de Registro*** estará localizado en el borde derecho, así este borde será nuestro indicador de volumen. Finalmente en la capa superior se tiene el mismo triángulo que la capa **fondo**, y esta última es máscara de la capa que contiene la barra.

Teniendo ya listos los elementos que controlarán el sonido, y antes de colocar el código ActionScript veamos la edición del archivo XML que nos indicará el listado de canciones que podremos reproducir con este swf.

Musica.xml

```
<Canciones>
        <cancion>Conteo regresivo</cancion>
        <cancion>Cuando me enamoro</cancion>
        <cancion>Mi niña bonita</cancion>
        <cancion>Yerbatero</cancion>
        <cancion>Me falta todo</cancion>
</Canciones>
```

Este archivo contiene únicamente un listado de nodos de XML con el nombre de las canciones en los archivos mp3 que cargará el reproductor, el nombre de la carpeta de ubicación y la extensión ".mp3" se le colocará con código al momento de cada llamado.

La carpeta con las canciones de nuestro ejemplo la llamaremos **música,** estará ubicada en la misma carpeta donde localizamos tanto el SWF que desarrollamos como el archivo XML anteriormente descrito y contendrá en este ejercicio los siguientes archivos:

Conteo regresivo.mp3
Cuando me enamoro.mp3
Mi niña bonita.mp3
Yerbatero.mp3
Me falta todo.mp3

La cantidad de archivos y sus nombres son irrelevantes ya que nuestro reproductor será capaz de actualizarse según lo que indique el archivo XML. En este sentido la importancia radica en el hecho de que el archivo XML debe reflejar exactamente el contenido de la carpeta **música**.

Una vez preparados todos los elementos necesarios para nuestro reproductor podemos colocar el código ActionScript que mostramos a continuación, el cual ubicaremos en el fotograma 1 del escenario principal de nuestro flash:

```
import flash.net.URLRequest;  // librerías de flash
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.events.*;
import flash.display.*;
import flash.net.*;

// VARIABLES PARA CARGA DEL XML

var ruta:URLRequest = new URLRequest("musica.xml");  // variable para indicar ruta al archivo XML
var loader:URLLoader = new URLLoader();  // variable para efectuar la carga del archivo XML
var cargador:URLLoader = new URLLoader(); // variable para cargar archivo XML
var miXML:XML;  // variable para cargar los datos dentro del archivo XML
var canciones:Array = new Array(); // arreglo para almacenar nombres de los mp3 indicados en el XML
var num:Number = 0;  var s:String; // variable para guardar el número de canciones indicadas en el XML

// VARIABLES PARA CONTROL DE SONIDO

var mi_canal:SoundChannel = new SoundChannel(); // variable para iniciar la reproducción
var i:Number = 0; // variable para registrar la canción que se debe reproducir

var milisegundos:Number = new Number(0);  // variable para registrar el segundo en que se hace pausa
var intervalo:Number = new Number(1); // variable para manejar el registro del tiempo de
```

```
reproducción

var mi_sonido:Sound = new Sound(); // variable para cargar cada mp3
var mi_ruta:URLRequest; // variable para indicar la ruta de cada mp3

var vol:SoundTransform; // variable para indicar volumen de sonido

// CARGA DE XML

loader.addEventListener(Event.COMPLETE, loadHandler); // Detecta cuando la carga del XML se ha completado

loader.load(ruta); // se ejecuta la carga del archivo XML

function loadHandler(miEvento:Event):void { // funciones a ejecutar al completar la carga del XML
	cargador = miEvento.target as URLLoader // se descarga el archivo en la variable cargador
	miXML = new XML(cargador.data); // se descargan los datos en la variable miXML

	for each(var nodo:XML in miXML.elements()) {  // se coloca cada nombre de canción dentro del arreglo canciones
		canciones[num] = nodo;
		num++;  // se indica nueva posición en el arreglo
	}
	num--; // num adquiere el número de canciones que se indican en el XML

	// AJUSTE DE VOLUMEN A LA MITAD

	vol = new SoundTransform();  // variable para control de volumen de sonido
	vol.volume = 0.5; // se ajusta volumen a la mitad
	cargar_cancion(); // se carga la canción actual

	// las siguientes tres líneas de comando ajustan el volumen, necesarias pues
	// la canción debe estar en reproducción para poderle ajustar dicho volumen

	mi_canal = mi_sonido.play(); // se comienza a reproducir la canción
	mi_canal.soundTransform = vol;  // se ajusta el volumen
	mi_canal.stop(); // se detiene la reproducción
}

// CARGA DE DATOS DE LAS CANCIONES

function cargar_cancion():void { // función que ejecuta la carga de un mp3
		mi_ruta = new URLRequest("musica/" + canciones[i] + ".mp3");
		mi_sonido = new Sound();
		mi_sonido.load(mi_ruta);
		cargar_info();
}
```

```
function cargar_info():void {  // función que carga la información de la canción dentro del mp3
	mi_sonido.addEventListener(Event.ID3, cargar_info_data);
}

function cargar_info_data(miEv:Event):void { // función que escribe la info en el campo de texto
	info.text = mi_sonido.id3.songName;
	info.text = info.text + "\r" + mi_sonido.id3.artist;
}

// BOTON PLAY/PAUSA

playpausa.addEventListener(MouseEvent.CLICK,  playpausar); // Detecta acción sobre el botón play/pausa

function playpausar(miev:MouseEvent) {
	if(playpausa.currentFrame==1) { // Si la canción está en pausa se inicia reproducción
		cargar_cancion(); // se carga la canción actual indicada por la variable i
		mi_canal = mi_sonido.play(milisegundos) // inicia la reproducción desde el segundo almacenado
		intervalo = setInterval(contar,10);  // cada 10 milisegundos se registra el tiempo
		playpausa.gotoAndStop("pausa");  // se muestra el botón de pausa
	}
	else { // Si la canción está en reproducción se coloca en pausa
		mi_canal.stop(); // se detiene la reproducción
		playpausa.gotoAndStop("play"); // se muestra el botón de play
		clearInterval(intervalo); // se detiene el registro de tiempo de reproducción
	}
}

function contar() { // función que cuenta los milisegundos de reproducción
 milisegundos = milisegundos+10;
}

// RECARGA DE CANCIONES AL FINALIZAR

mi_sonido.addEventListener(Event.SOUND_COMPLETE,  recargar); // Detecta cuando termina la reproducción

function recargar(miEvento:Event):void { // función que carga el próximo mp3
	i++;
	if (i>num) i=0;
	cargar_cancion();
}

// BOTON DETENER

detener.addEventListener(MouseEvent.CLICK, parar); // Detecta acción sobre el botón stop
```

```
function parar(miev:MouseEvent) { // funciones para detener la reproducción
	clearInterval(intervalo); // detiene el registro de tiempo de reproducción
	milisegundos = 0; // se reinicia el registro de tiempo de reproducción
	mi_canal.stop(); // se detiene la reproducción de sonido
	playpausa.gotoAndStop("play"); // se muestra el botón de play
}

// BOTON AVANZAR

ff.addEventListener(MouseEvent.CLICK, avanzar); // Detecta acción sobre el botón avanzar

function avanzar(miEvento:MouseEvent) { // funciones para avanzar canción
	i++; // se avanza el contador de canción
	if (i>num) i=0; // si la canción actual es la última se regresa a la primera
	mi_canal.stop();  // se detiene canción actual
	cargar_cancion(); // se carga nueva canción
	mi_canal = mi_sonido.play(0,10); // se reinicia reproducción
}

// BOTON RETROCEDER

rr.addEventListener(MouseEvent.CLICK, retroceder); // Detecta acción sobre el botón retroceder

function retroceder(miEvento:MouseEvent) { // funciones para retroceder canción
	i--; // se retrocede el contador de canción
	if (i<0) i=num; // si la canción actual es la primera se envía a la ultima
	mi_canal.stop(); // se detiene canción actual
	cargar_cancion(); // se carga nueva canción
	mi_canal = mi_sonido.play(0,10); // se reinicia reproducción
}

// BOTON VOLUMEN

volumen.dial.addEventListener(MouseEvent.MOUSE_DOWN, desplazar); // Detecta presión al botón de volumen
volumen.dial.addEventListener(MouseEvent.MOUSE_UP, soltar); // Detecta liberación del botón de volumen

function desplazar(miEvento:MouseEvent):void { // función que permite arrastre de la barra de volumen
	// se permite arrastre dentro del área del triángulo del movieclip volumen:
	miEvento.currentTarget.startDrag(false,new
Rectangle(0,miEvento.currentTarget.y,100,miEvento.currentTarget.y));
	// se ajusta el volumen dependiendo de la posición donde se coloque la barra de volumen:
	miEvento.currentTarget.addEventListener(Event.ENTER_FRAME, ajustar_volumen);
}

function ajustar_volumen(miEvento:Event):void { // función que ajusta el volumen
```

```
        vol = new SoundTransform(); // se reinicia variable de control de volumen
        vol.volume = volumen.dial.x/100; // se calcula el valor del volumen según la posición de la
barra
        mi_canal.soundTransform = vol; // se asigna el valor del volumen al canal
}

function soltar(miEvento:MouseEvent):void { // función que detiene ajuste del volumen
        miEvento.currentTarget.stopDrag(); // se detiene el arrastre de la barra
        // se detiene el ajuste del volumen en el canal:
        miEvento.currentTarget.removeEventListener(Event.ENTER_FRAME, ajustar_volumen);
}

// STAGE

stage.addEventListener(MouseEvent.MOUSE_UP, soltar_stage); // detecta liberacion del mouse
                                        // fuera del movieclip volumen
function soltar_stage(miEvento:MouseEvent):void { // funcion que detiene ajuste del volumen
        volumen.dial.stopDrag(); // se detiene el arrastre de la barra
        // se detiene el ajuste del volumen en el canal:
        volumen.dial.removeEventListener(Event.ENTER_FRAME, ajustar_volumen);
}
```

Una vez colocado el código ActionScript podemos reproducir los archivos de audio ubicados en la carpeta música, sin importar la cantidad de archivos o el nombre que se les asigne. Para el correcto funcionamiento sólo debemos mantener la misma ubicación de los archivos y la extensión mp3 de los archivos de audio.