Flash CS4 para la animación y el grafismo digital

Salvador Linares Mustarós
Queralt Viladevall Valldeperas

PID_00215905

UOC
Universitat Oberta de Catalunya

Índice

Introducción..... 5

1. Animación en Flash y fotogramas..... 7
1.1. Introducción 7
1.2. Instrucciones para crear un programa en Flash estilo fenaquistiscopio 8
1.3. Actividades 17

2. Capas..... 19
2.1. Introducción 19
2.2. Instrucciones para crear un programa con dos capas en Flash ... 19
2.3. Actividades 22

3. Películas dentro de películas..... 24
3.1. Introducción 24
3.2. Instrucciones para crear un clip de película 24
3.3. Actividades 32

4. Botones y campos de texto..... 34
4.1. Introducción 34
4.2. Un programa con un botón y los tres tipos de cajas de texto 35
4.3. Actividades 40

5. Action..... 41
5.1. Introducción 41
5.2. Unas ideas de Action 41
5.3. Cuando un programa “habla” 44
5.4. Actividades 45

Introducción

Este módulo va acompañado de un total de dieciocho programas efectuados en Flash.

Su intención principal es ofrecer un contenido teórico autosuficiente con unos primeros ejemplos de trabajo.

Estamos seguros de que una vez eliminado el temor inicial al programa, habréis adquirido confianza para generar múltiples ideas que aumentarán vuestra creatividad hasta límites insospechados, tal y como nos habéis mostrado año tras año.

1. Animación en Flash y fotogramas

1.1. Introducción

Flash se fundamenta en la capacidad humana para percibir movimiento aparente a partir de la observación de secuencias de imágenes estáticas.

En el siglo XIX, este efecto se conocía como la teoría de persistencia de la retina. Esta teoría, hoy considerada un mito, explicaba que una imagen proyectada todavía se visualizaba unas fracciones de segundo después de que hubiera desaparecido la proyección. Esta idea permitía explicar por qué el cerebro podía enlazar varias imágenes proyectadas y crear una sola imagen visual, móvil y continua.

La teoría la propuso el físico belga Joseph-Antoine Ferdinand Plateau en 1829. Para demostrar su teoría, Plateau invento en 1832 el fenaquistiscopio, uno de los precursores del cinematógrafo.

El aparato consiste en una serie de dibujos de un mismo objeto, ligeramente distintos y dispuestos de manera uniforme en una placa circular. Cuando la placa gira a una determinada velocidad, la visión de las imágenes a través de un espejo crea la ilusión de una imagen en movimiento.

Al poco tiempo de su invención, Plateau estimó que el número de imágenes que se tenían que visualizar por segundo si se quería tener una imagen de movimiento óptima era dieciséis. Este es el motivo por el que las primeras películas utilizaron dieciséis fotogramas por segundo.

Hay muchos otros aparatos que utilizan el efecto de percibir movimiento aparente a partir de la observación de secuencias de imágenes estáticas, como por ejemplo el taumátropo, el zoótropo, el praxinoscopio o el zoopraxiscopio. La animación en Flash se basa también en este efecto, pues se proyectan fotogramas a una velocidad determinada, normalmente acotada entre 12 y 24 fotogramas por segundo.

En este primer apartado, aprenderemos a crear un pequeño programa en el que proyectaremos once imágenes de un caballo fotografiado por Muybridge, para producir la sensación de galope.

1.2. Instrucciones para crear un programa en Flash estilo fenaquistiscopio

1) Primero. Obtenemos los dibujos o imágenes que formarán un ciclo repetitivo en formato jpg, todos de las mismas dimensiones que la imagen Muybridge_horse_gallop.jpg (la encontraréis en la carpeta *programa1*), y los guardamos con nombres que nos permitan conocer el orden dentro del ciclo.

Nota

Para obtener las mismas dimensiones, primero se ha creado un archivo psd en el que se han pegado copias de los fragmentos de imágenes y se han ido moviendo hasta obtener una posición parecida. (El .psd está dentro de la carpeta *programa1*.)

Figura 1. Imagen original Muybridge.jpg

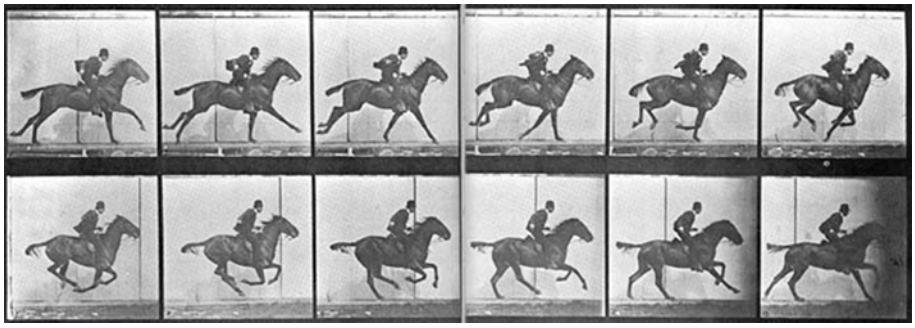
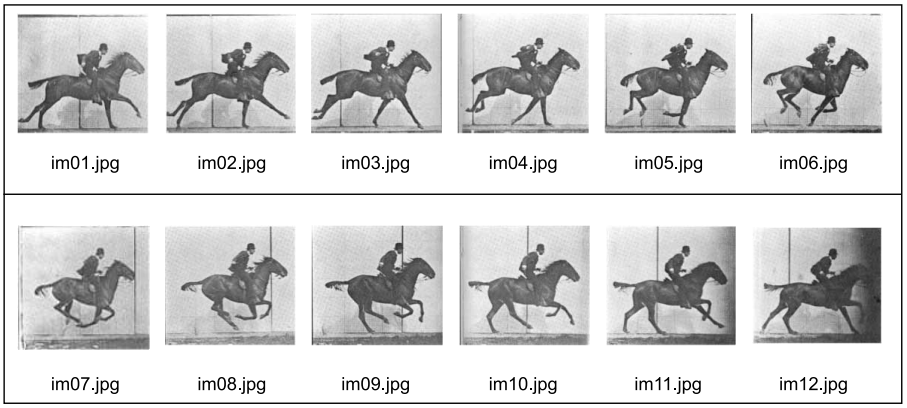


Figura 2. Visualización de las doce imágenes de igual tamaño extraídas de la imagen Muybridge.jpg.

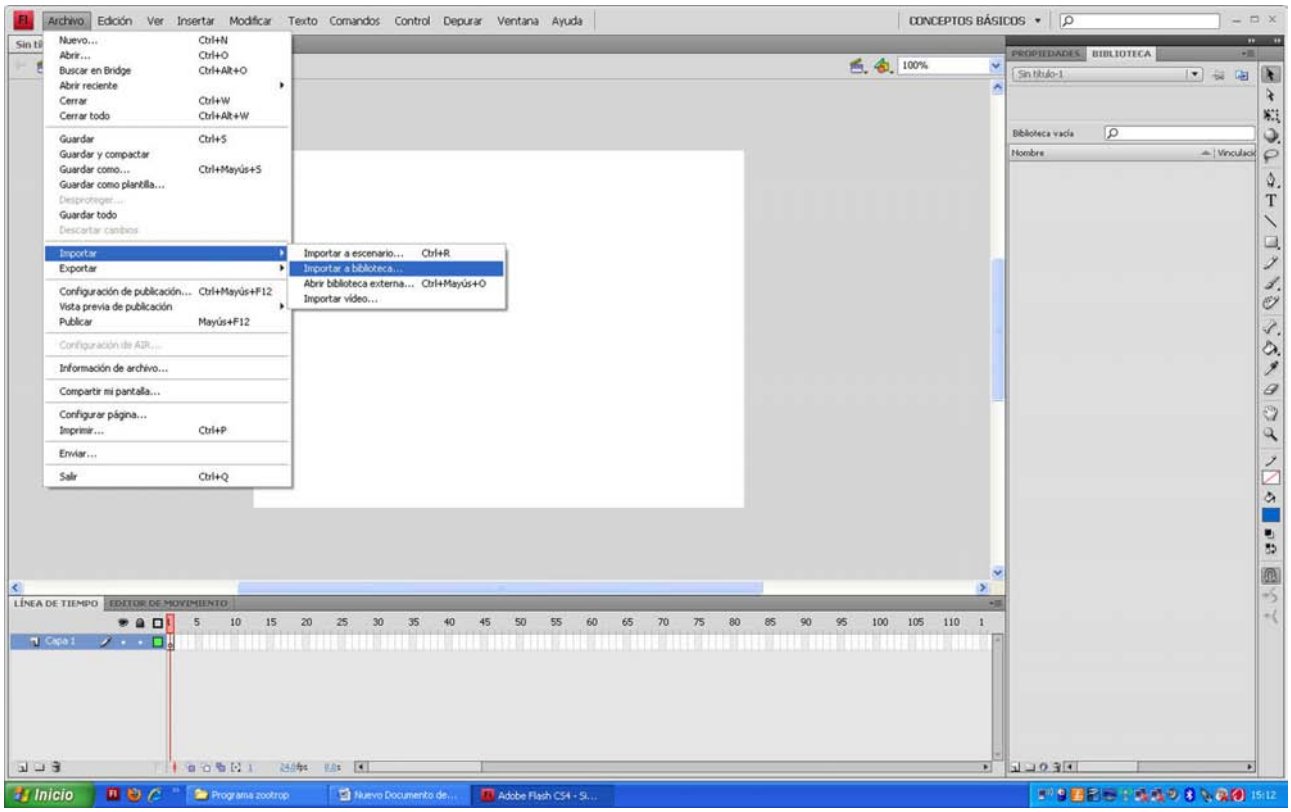


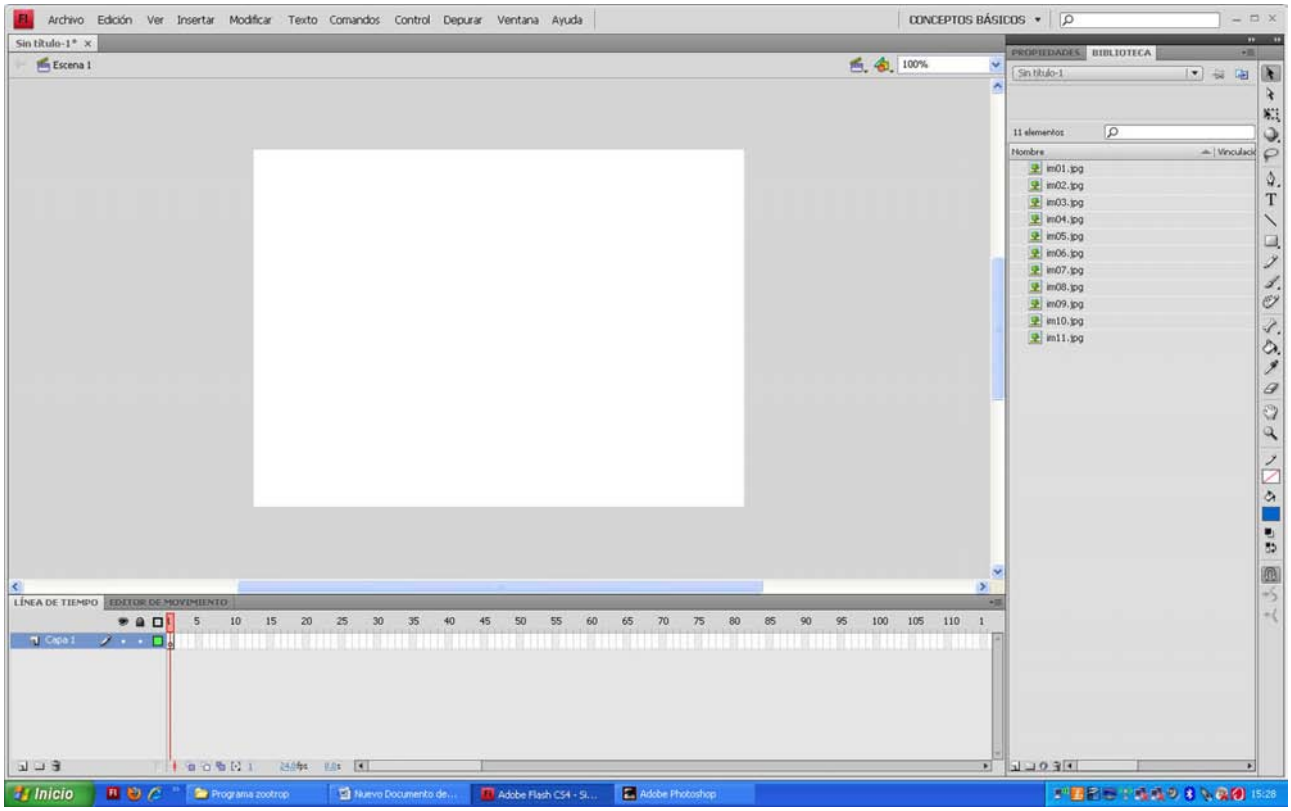
Nota: Dado el mal estado de la última imagen, en este ejercicio prescindiremos de la imagen 12.

2) Segundo. Abrid Flash, e id a *Archivo -> Nuevo -> Archivo de Flash(AS 3.0)*.

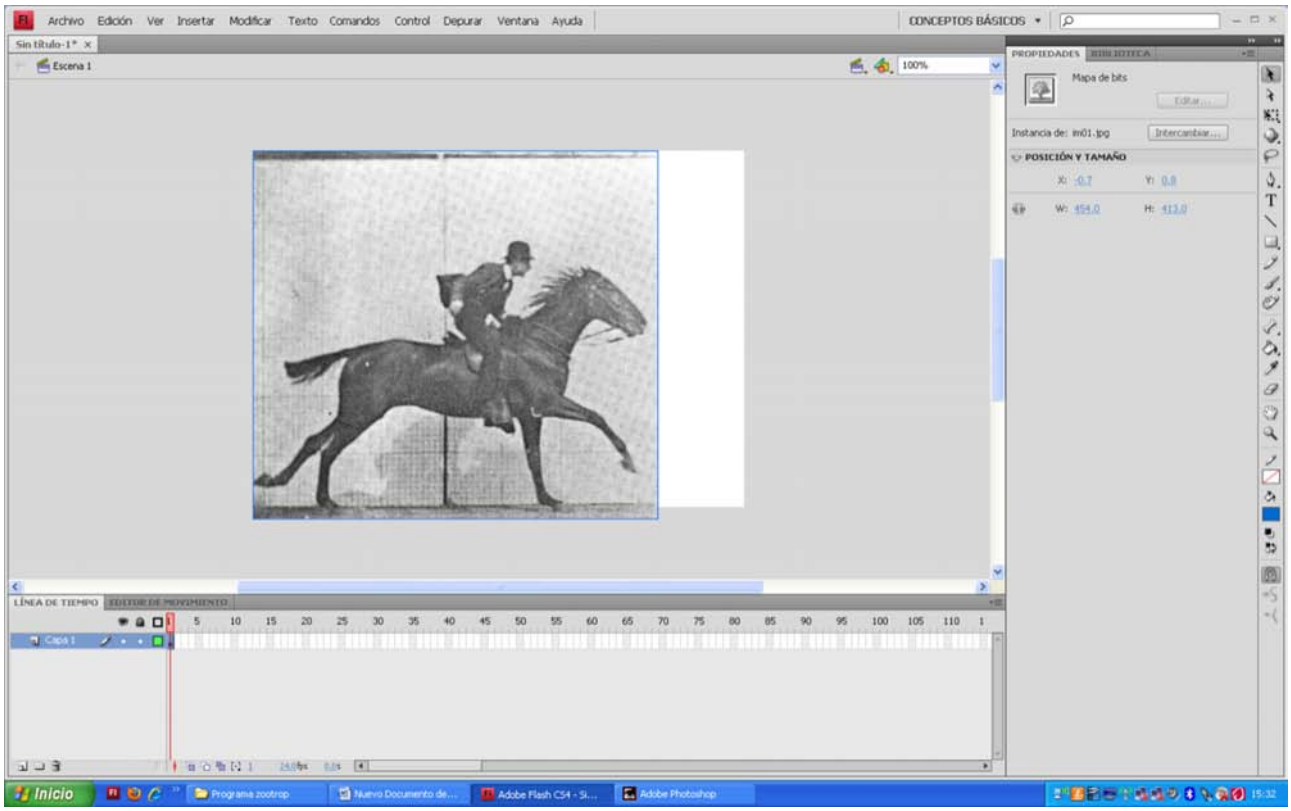
Recordad guardar de vez en cuando con el nombre que queréis. Por ejemplo, programa1 o zoótropo.

3) Tercero. Importad a la biblioteca todas las imágenes creadas.



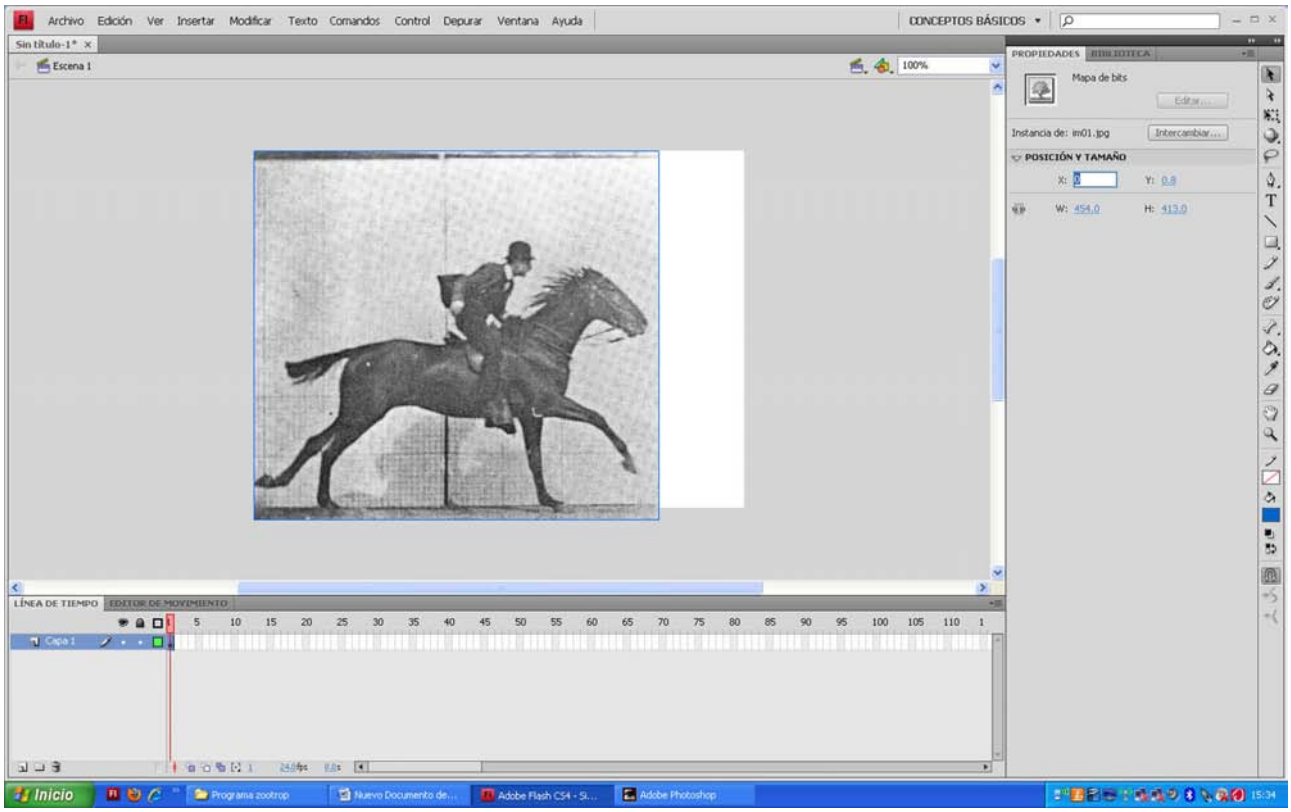


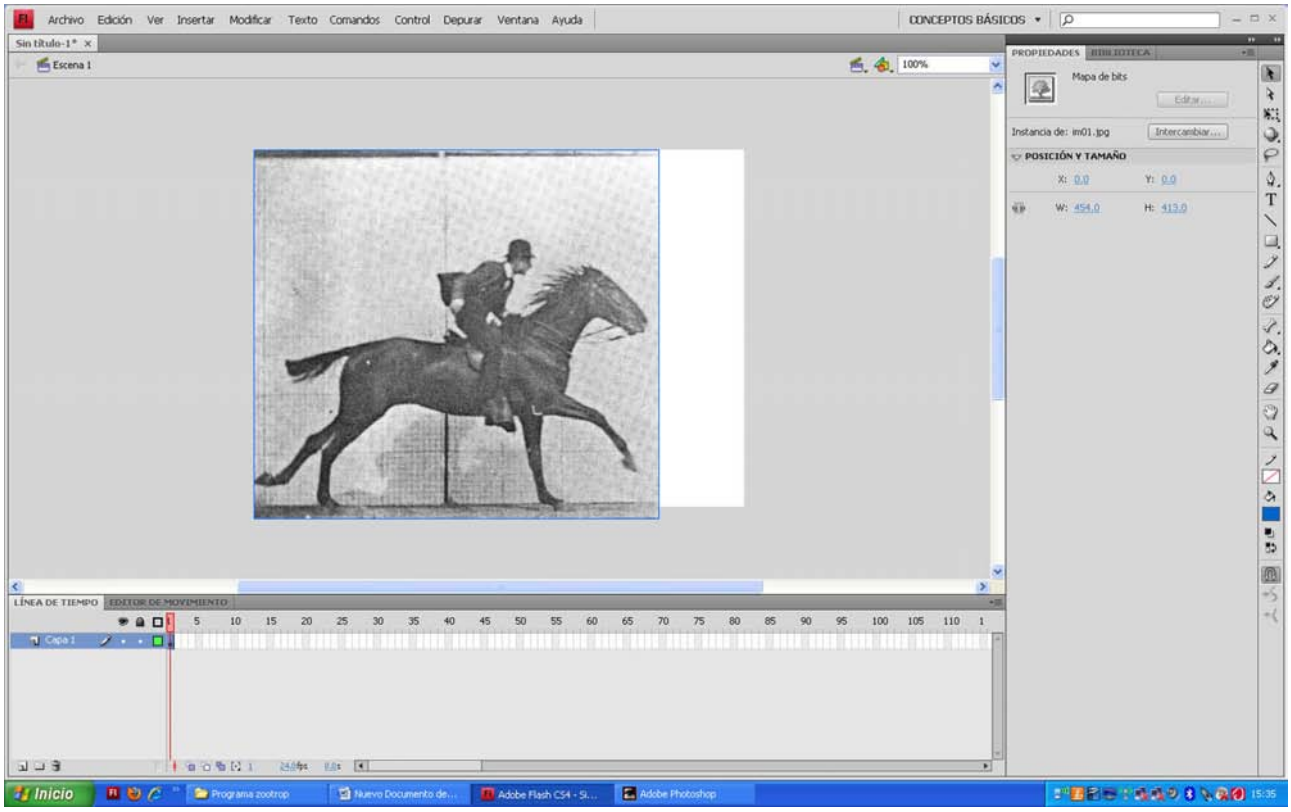
4) Cuarto. Arrastrad im01.jpg de la biblioteca al escenario y haced clic en propiedades. Observad que mi imagen im01.jpg tiene un tamaño de 450 de ancho por 413 de alto.



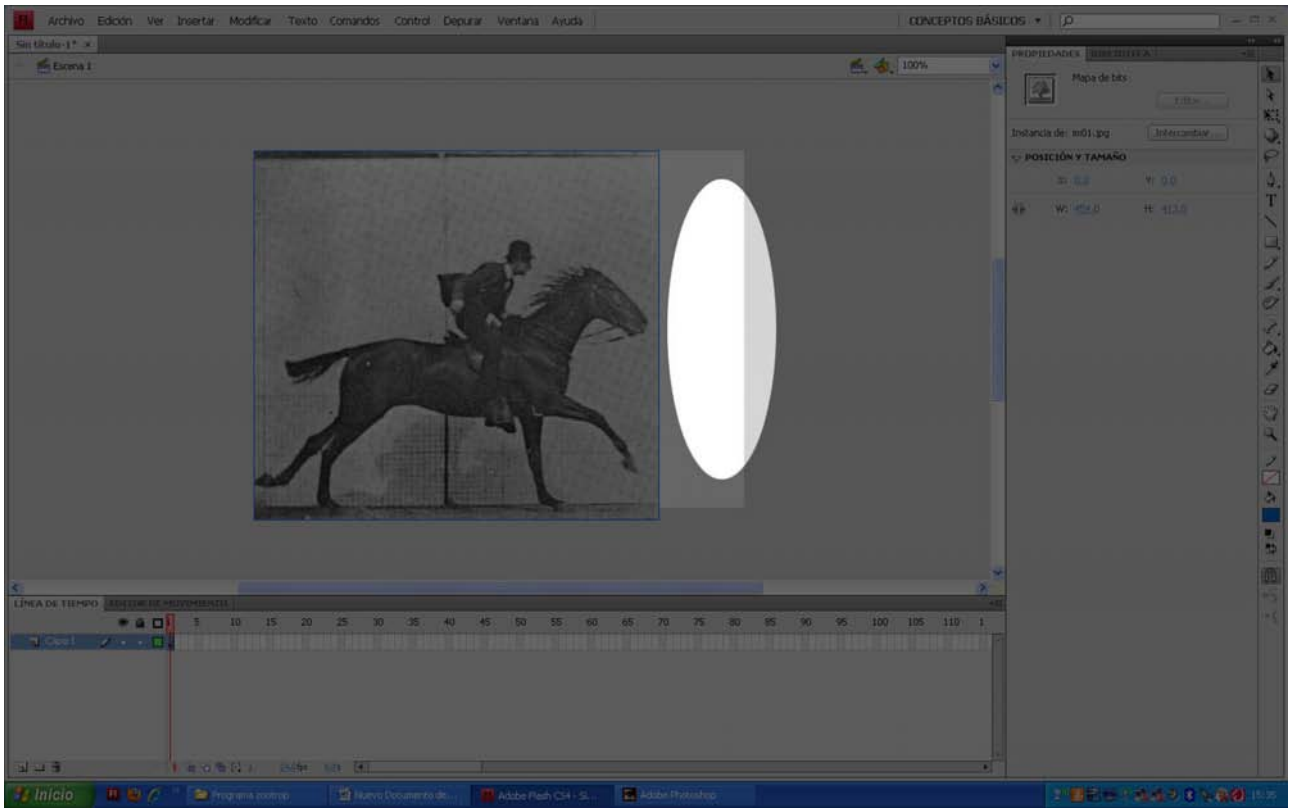
Recordad que si queréis ver las propiedades de una imagen, tenéis que seleccionar la imagen haciendo clic sobre la misma.

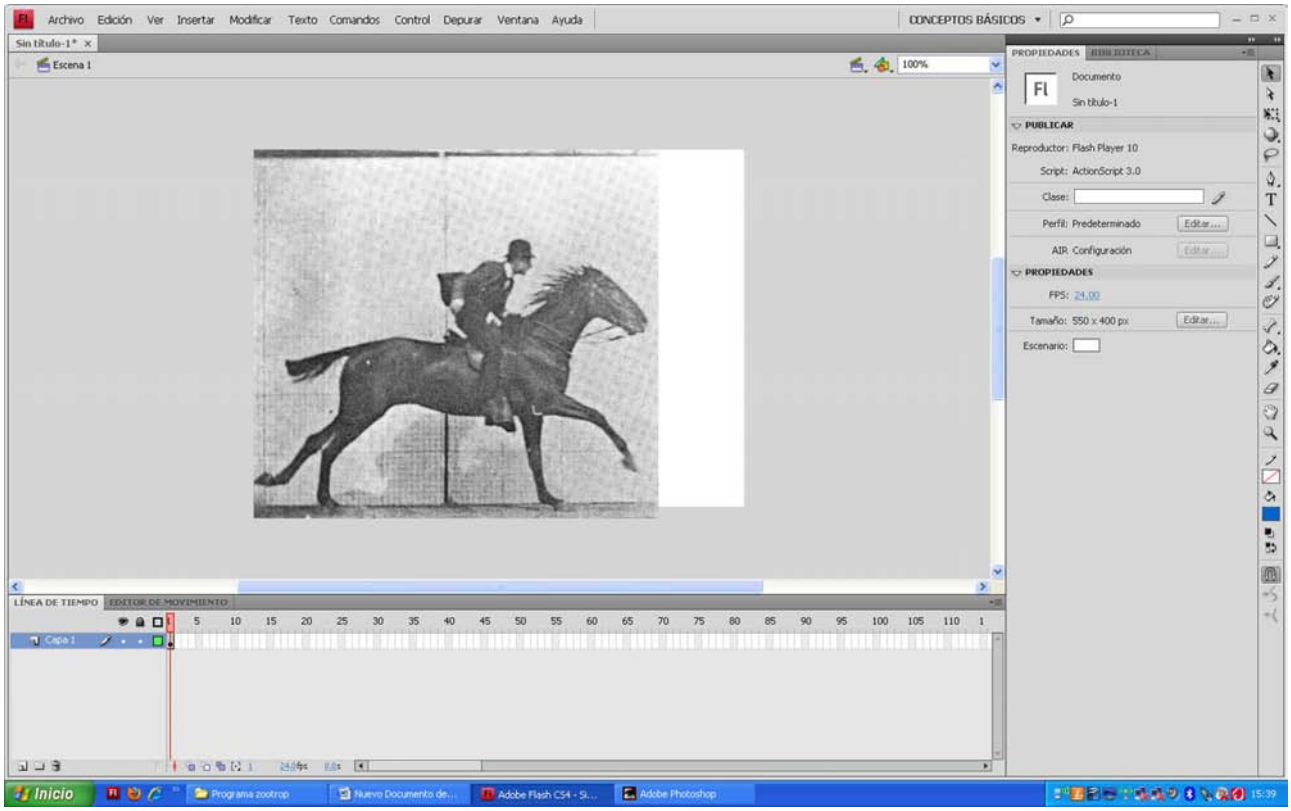
5) Quinto. Haced clic donde dice posición y tamaño *x* y poned 0 en la *x* y 0 en la *y*. De este modo, la imagen estará bien centrada respecto a nuestro escenario. Los puntos $x = 0$ e $y = 0$ corresponden a la esquina superior izquierda, que es nuestro centro de coordenadas.



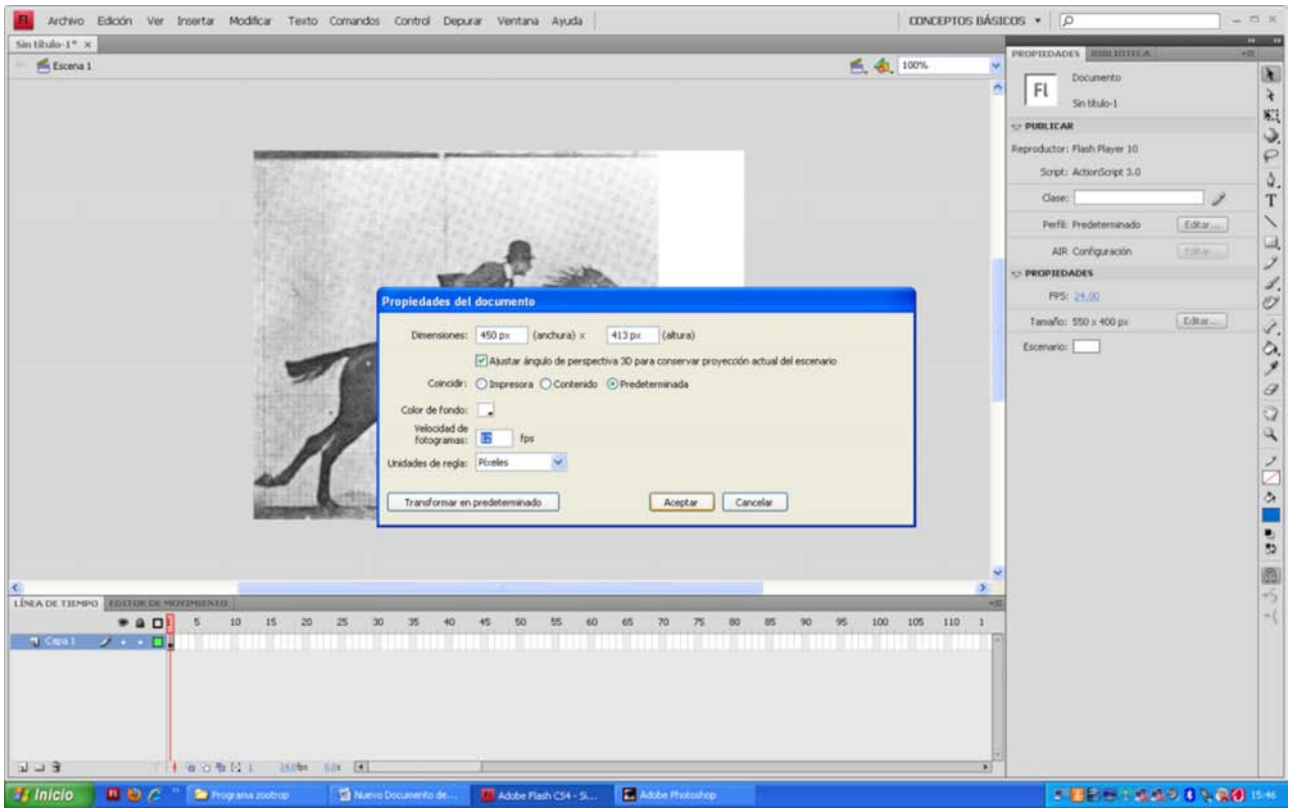


6) Sexto. Haced clic en cualquier posición fuera de la imagen para ver las propiedades del escenario.



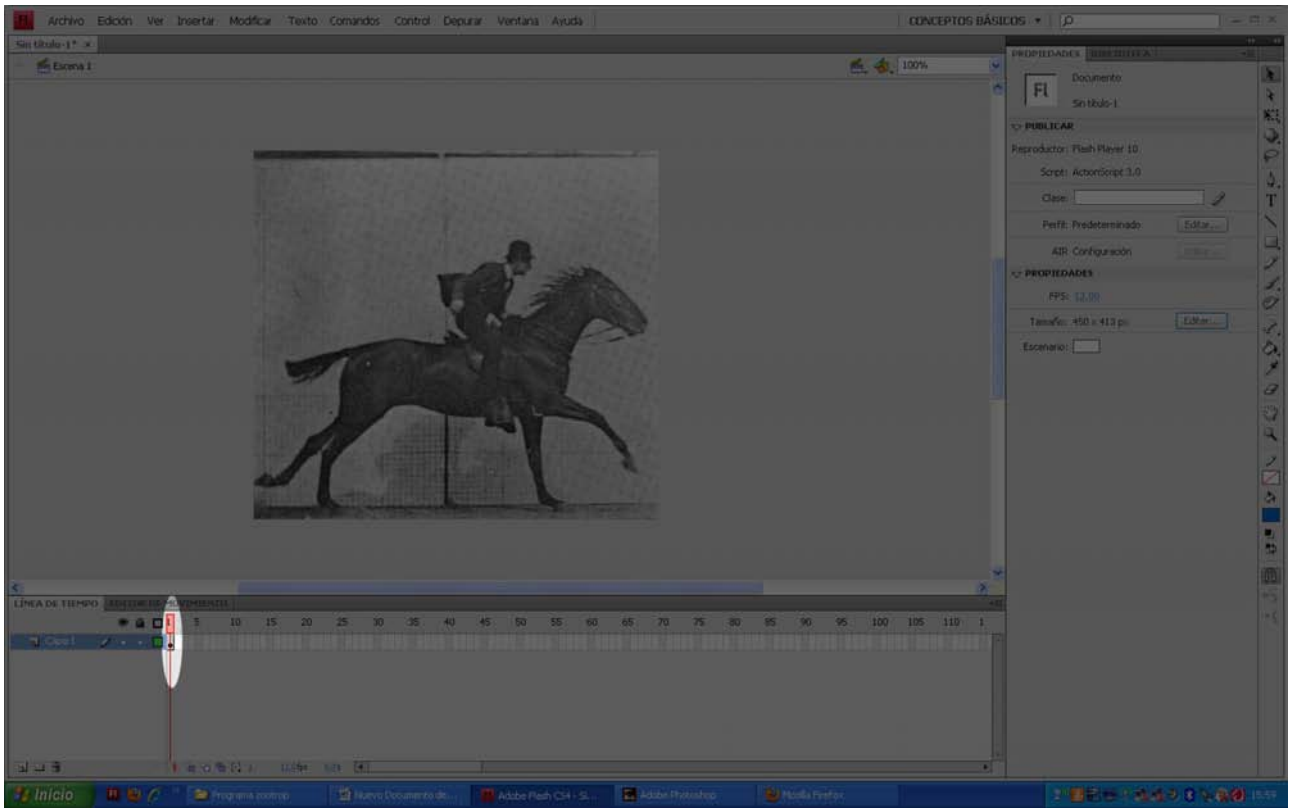


Observad que el tamaño del escenario es 550 x 400. Si hacéis clic en *Editar*, podéis cambiar estos valores por 450 y 413, y así haréis que el escenario tenga el mismo tamaño que las imágenes.

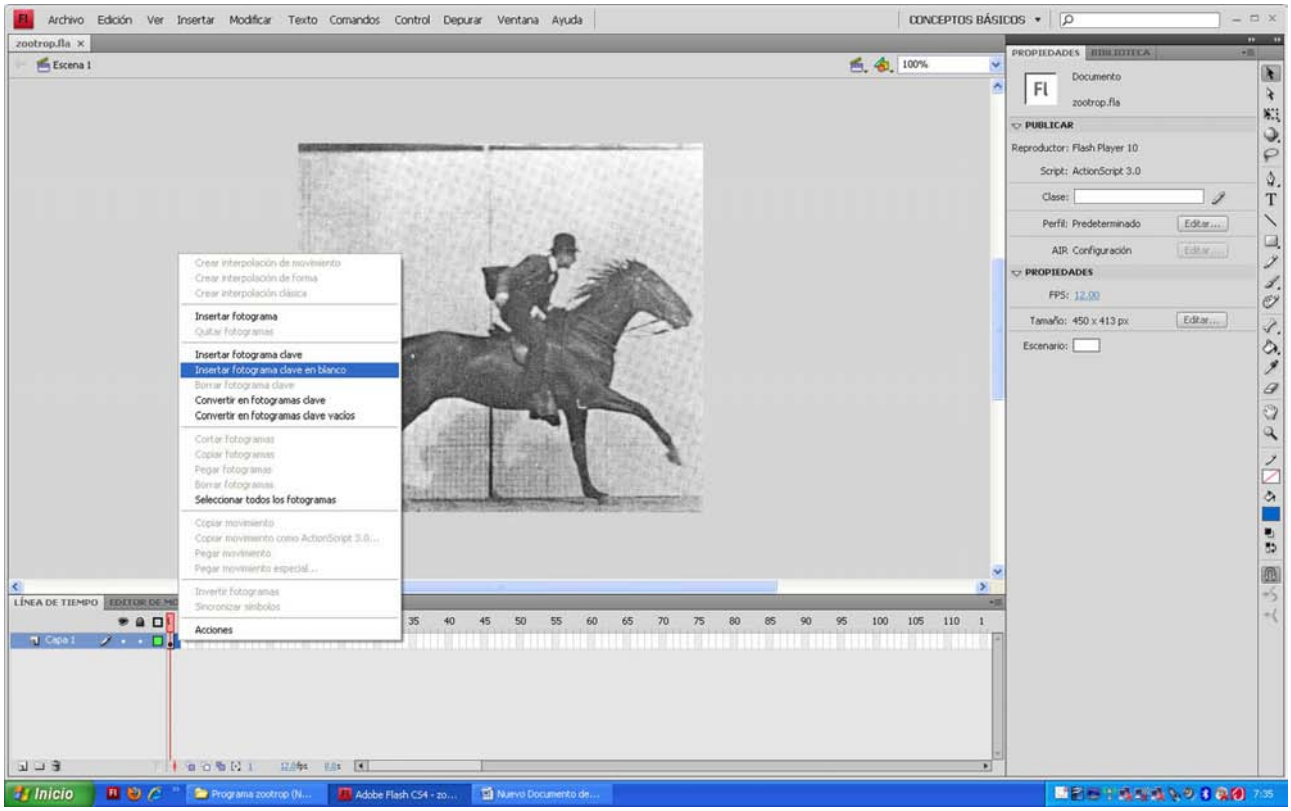


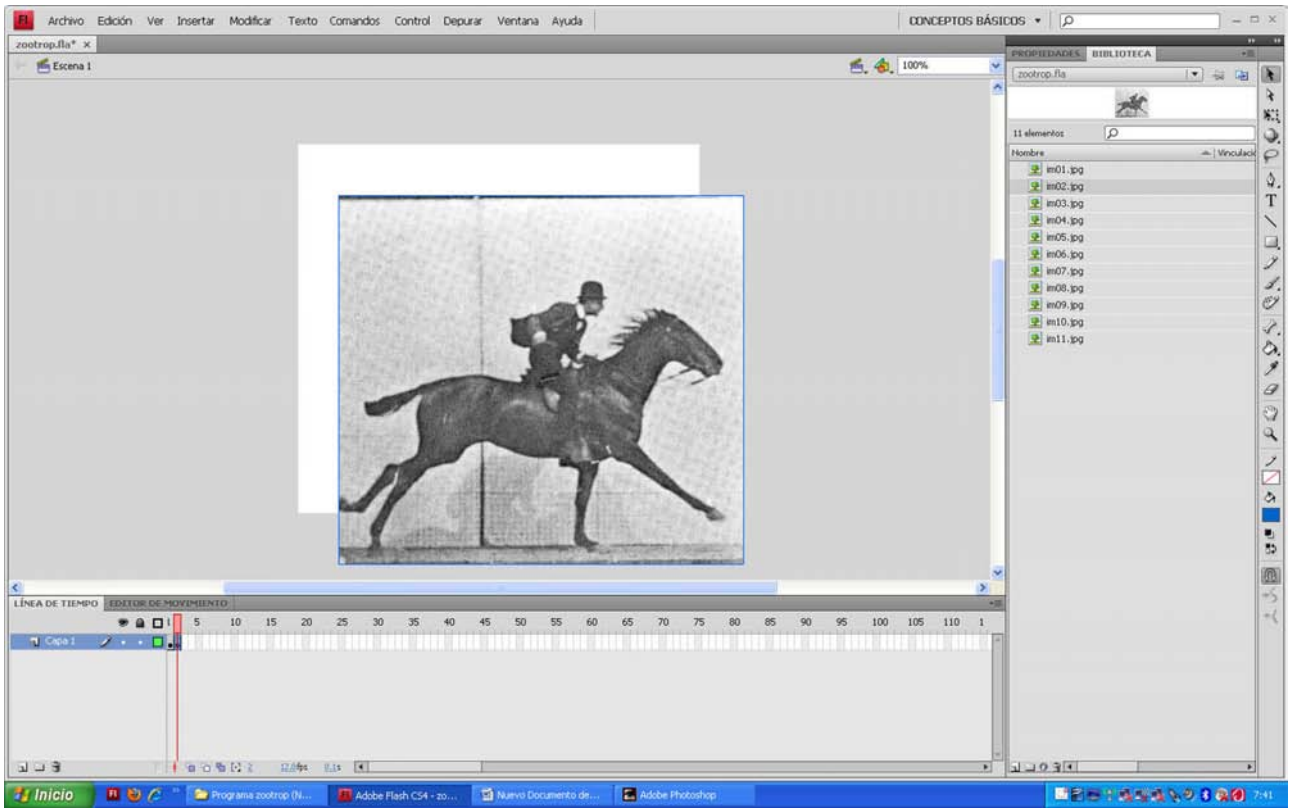
7) Séptimo. Poned velocidad de fotogramas a doce fotogramas por segundo (habitualmente, en web se suele utilizar 12 fps).

Podréis observar que en la línea del tiempo el cuadradito que se denomina marco (*frame*) o fotograma es ahora de color gris: el programa nos indica así que no está vacío.

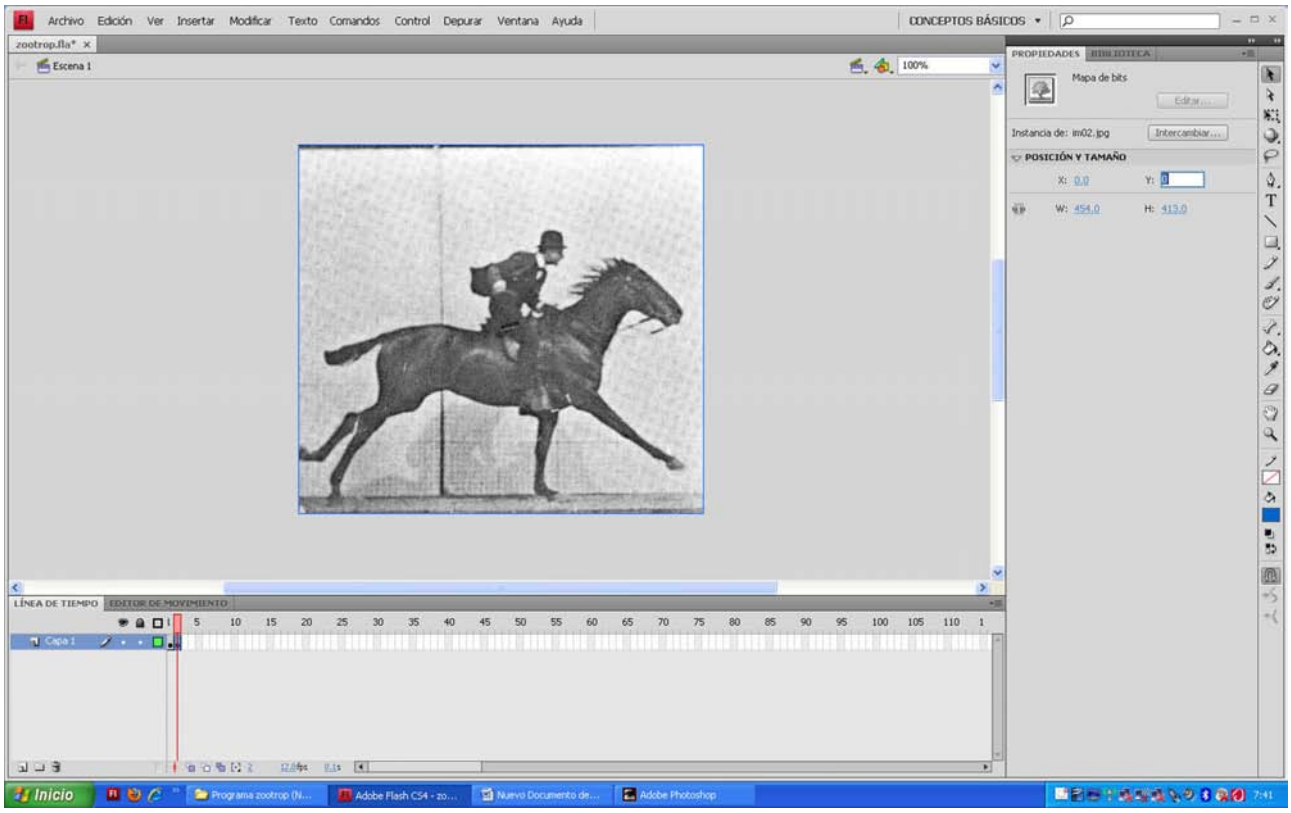


8) Octavo. Si hacéis clic con el botón derecho del ratón en el marco 2, se abre un menú contextual. Seleccionad la opción de crear fotograma en blanco y veréis que aparece un marco nuevo y el escenario queda vacío. Arrastrad la imagen de la biblioteca im02.jpg sobre el escenario.

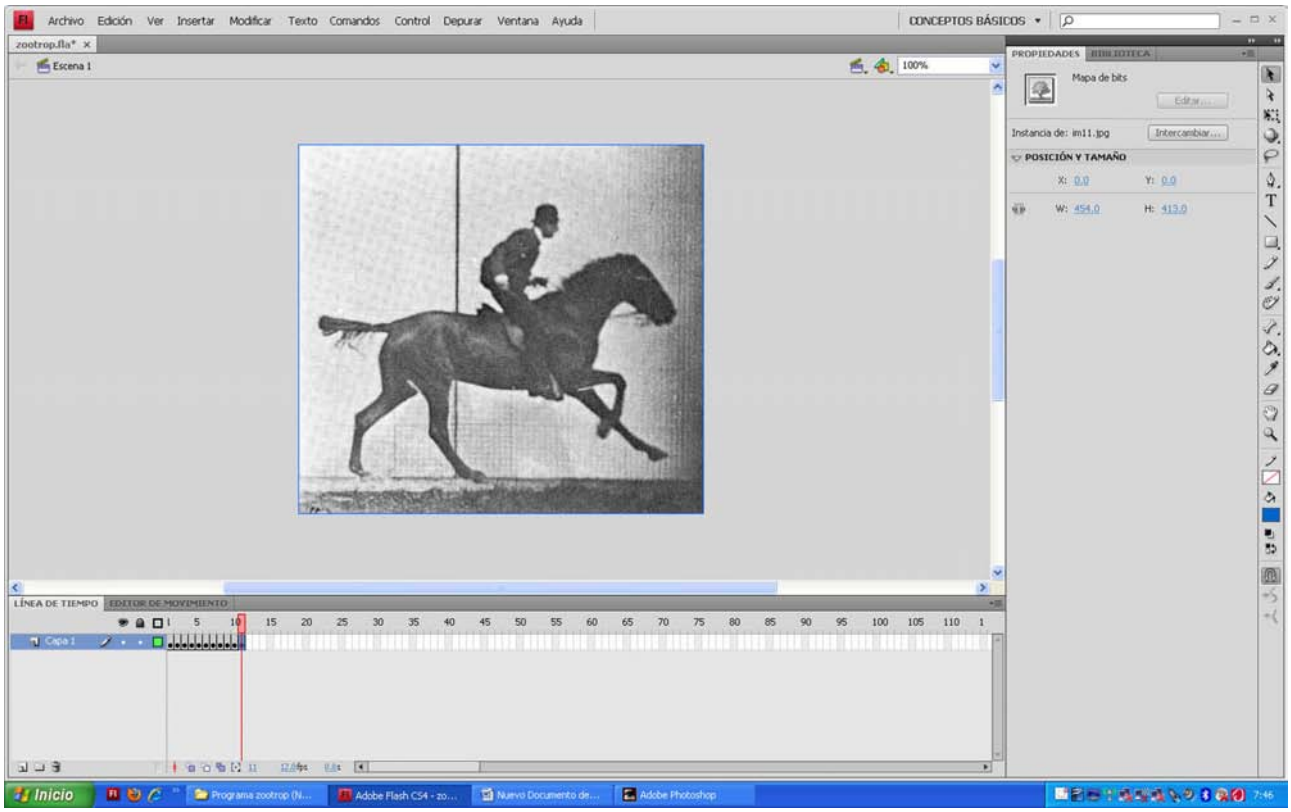




Volved a poner la *x* y la *y* de propiedades de la imagen a 0 para tenerlo centrado respecto a la imagen anterior.



9) Noveno. Repetid el proceso hasta insertar la imagen im11.jpg.



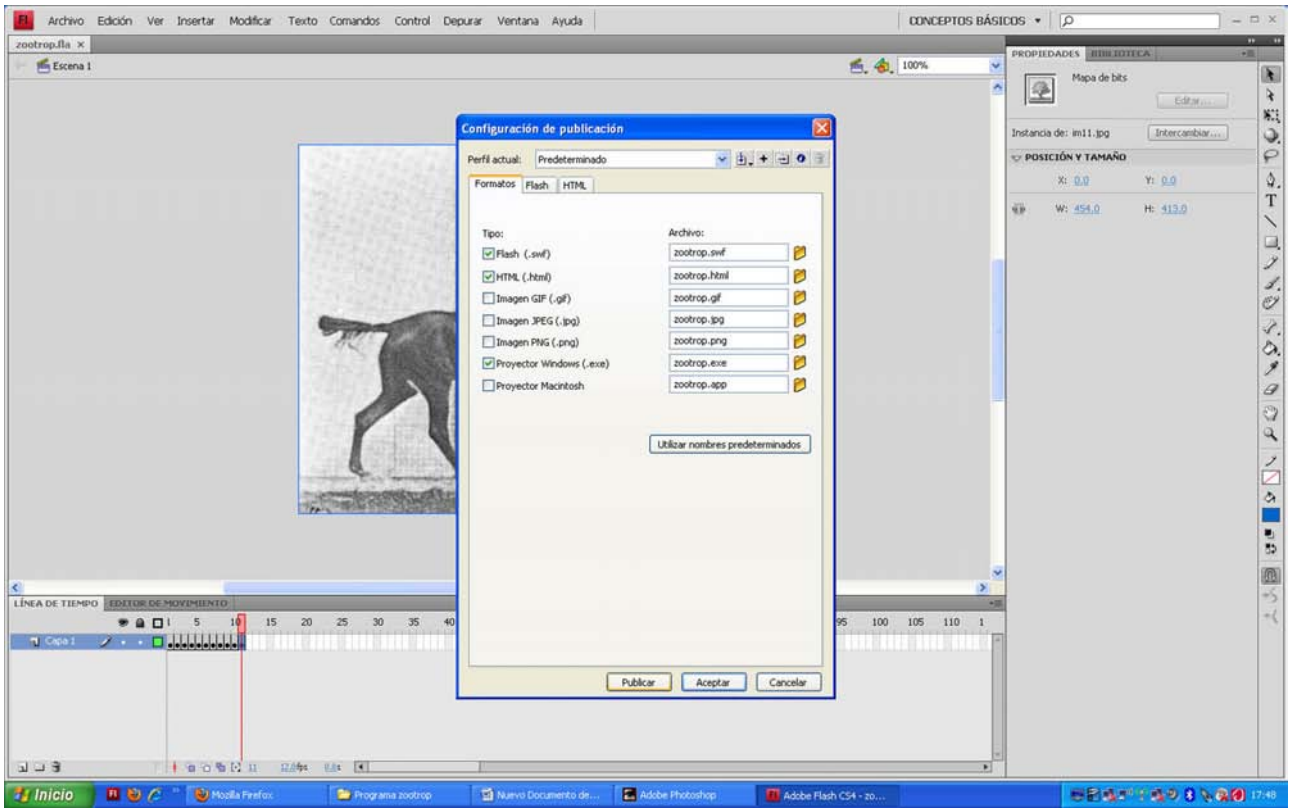
10) Décimo. Finalmente, solo tenéis que generar el swf, exe o html para ver el resultado.

Id a *Archivo* -> *Configuración de publicación* y seleccionad las opciones html, swf y exe.

Nota

Si tenéis un Mac, elegid app en lugar de exe.

Por último, seleccionad *Publicar* y en el mismo lugar en el que tengáis el archivo fla obtendréis de regalo los tres archivos ejecutables. Haced doble clic sobre cada uno para ver su resultado.



1.3. Actividades

Actividad 1

Buscad vídeos en la página http://www.youtube.com relacionados con los aparatos presentados en este apartado 1. Personalmente, os recomendamos las direcciones siguientes:

http://www.youtube.com/watch?v=glk6xrt0q3k

http://www.youtube.com/watch?v=rsf7n9siitg

http://www.youtube.com/watch?v=6rmoawgn7bi&feature=related

http://www.youtube.com/watch?v=v7-gccjbhbg&feature=related

Actividad 2

Intentad modificar la velocidad de fps para ver los efectos sobre la película.

Actividad 3

Intentad ver algún trozo de la serie de dibujos *Los Picapiedra*. Por ejemplo, el fragmento que podemos encontrar en la dirección siguiente:

http://www.youtube.com/watch?v=l5hxfp3-bzg&feature=related

Este es un ejemplo claro de lo que se conoce como animación limitada. Podéis observar que los personajes hablan moviendo los ojos y la boca mientras el cuerpo y el fondo están estáticos. Esta idea permite elaborar mucha producción de material en poco tiempo, con lo que se puede atender mucha demanda.

Actividad 4

Cread un nuevo programa en Flash. Con las herramientas de dibujo, cread en el primer marco una cara con una boca sonriente. Haced clic con el botón derecho del ratón en el marco 2 y cuando salga el menú contextual, seleccionad la opción de crear fotograma clave y veréis que aparece un nuevo marco con una imagen exactamente igual a la del marco anterior. Borrad la boca y haced-la triste. Jugad con la velocidad de fotogramas para que, cuando ejecutéis el programa, aparezca un programa tipo taumátropo.

2. Capas

2.1. Introducción

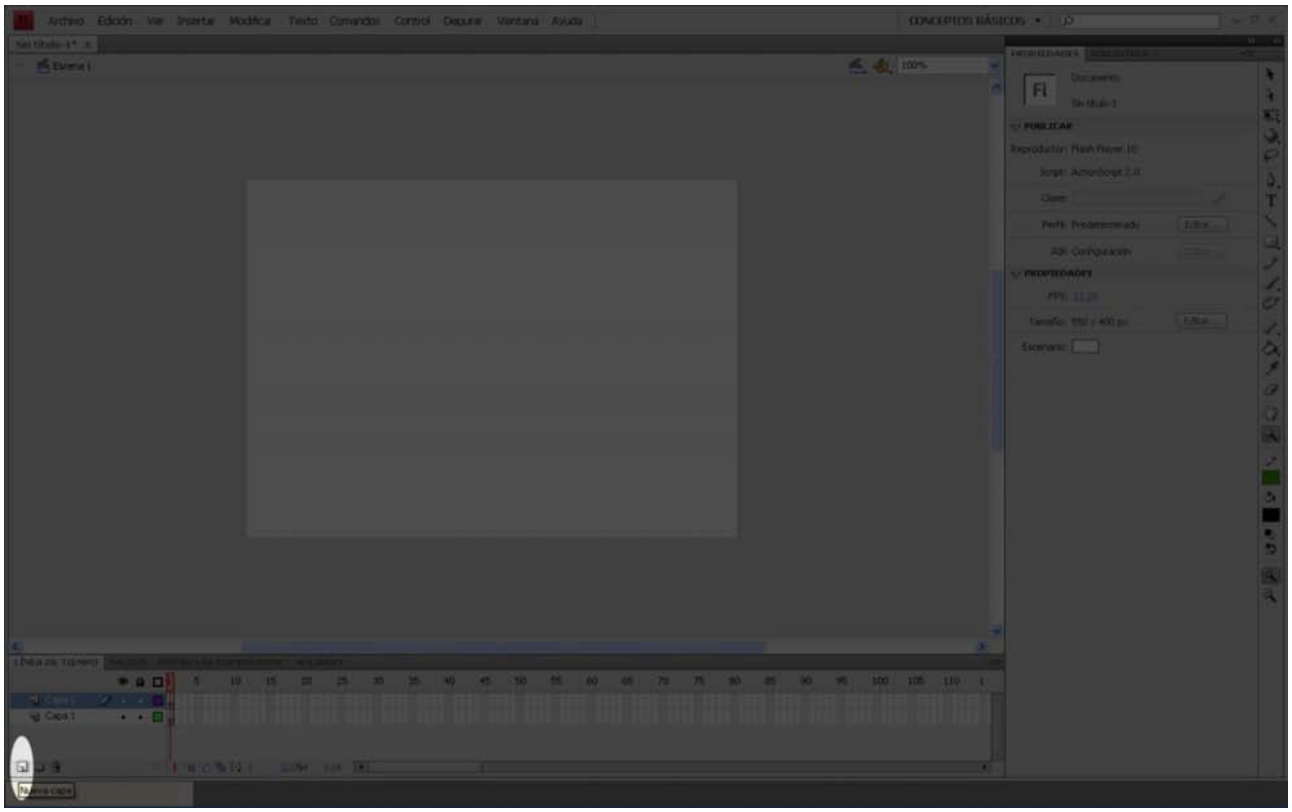
Flash ofrece una opción muy interesante que consiste en trabajar por capas. Para aquellos que sois nuevos en el tema de las capas, podéis entender las capas como hojas de acetato transparente, situadas una sobre otra, en las que es posible trabajar objetos de manera independiente. Esto ha facilitado mucho el trabajo, puesto que si por ejemplo queremos eliminar o mover de lugar un objeto y no hemos trabajado en capas, es posible que o no lo podamos hacer o que quede un espacio en blanco, lo que obliga a rehacer el fondo o los objetos de debajo, con lo que se pierde tiempo en cada cambio.

2.2. Instrucciones para crear un programa con dos capas en Flash

1) Primero. Abrid Flash, id a *Archivo* -> *Nuevo* -> *Archivo de Flash* (AS 3.0) y guardadlo de vez en cuando con el nombre que queráis.

En este ejemplo, programa2.

2) Segundo. Haced clic en *Nueva capa* y observad cómo se crea una nueva capa.



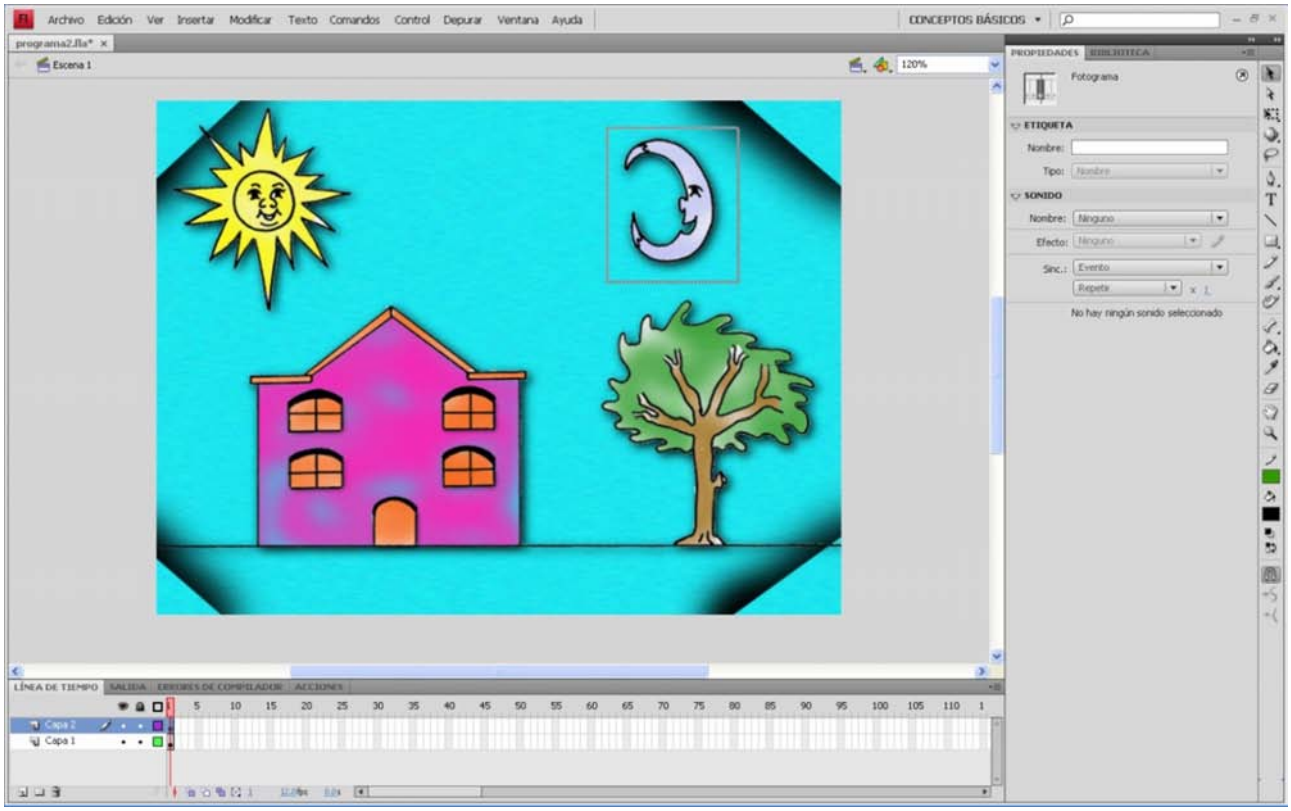
3) Tercero. Seleccionad el primer fotograma de la capa 1 y dibujad o importad algún objeto.

Por ejemplo, nosotros importaremos la imagen fondo.jpg

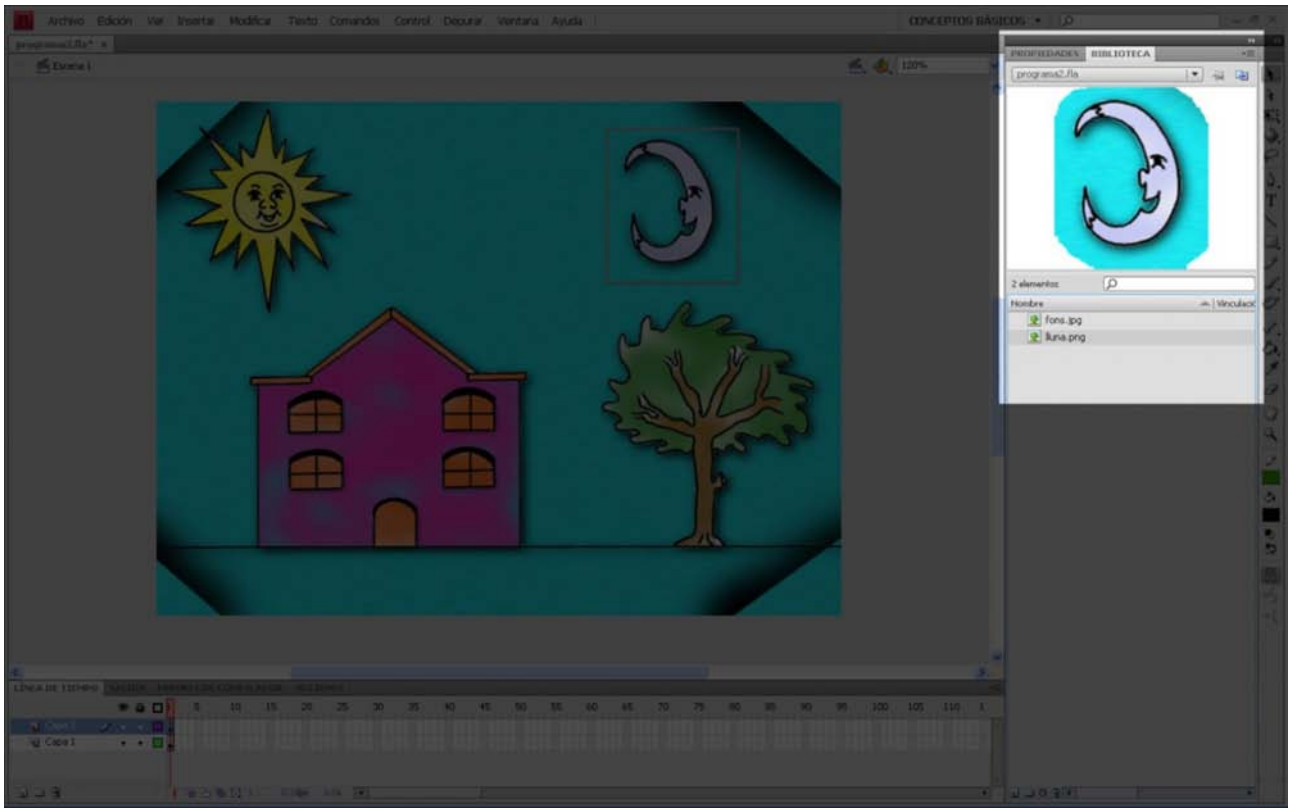
Ajustad las medidas del escenario a las medidas de la imagen de fondo y, si es necesario, ampliad el escenario para tener mejor visión del conjunto.

4) Cuarto. Haced clic en el fotograma 1 de la capa 2 y elaborad un segundo dibujo o importad una nueva imagen.

Por ejemplo, nosotros importaremos la imagen luna.png y la colocaremos en la parte superior derecha del escenario.



Si hacéis clic en la pestaña *Biblioteca*, veréis las imágenes importadas al programa.



Es muy interesante observar que Flash respeta las transparencias de los formatos png.

Por último, solo hay que comentar que a pesar de que podríamos haber integrado la imagen luna en el fondo, como por ejemplo la casa, el sol o el árbol, en ciertas ocasiones nos puede ser muy útil tenerla separada.

Por ejemplo, si queremos dar la impresión de que al pasar el ratón por encima de la luna, esta desaparece. Trabajar en capas puede ser muy útil y os recomendamos que utilicéis esta técnica.

2.3. Actividades

Actividad 1

Intentad importar al escenario una imagen modificada de una ya importada al escenario. Por ejemplo, pintad sobre la luna y guardad la imagen con el mismo nombre luna.png, y volved a importarla al escenario del programa 2. Observad las dos opciones que ofrece el menú Flash cuando intentáis llevar a cabo esta opción. Sin guardar los cambios, probad cada una de las dos opciones para contrastar los resultados de elegir cada una de las opciones.

Actividad 2

Cread un taumátropo de modo que haya un fondo fijo en dos fotogramas y dos imágenes ligeramente distintas, una en el fotograma 1 y la otra en el fotograma 2. Modificad la velocidad de fps para obtener la sensación de movimiento de la película óptima.

3. Películas dentro de películas

3.1. Introducción

Flash permite crear películas dentro de películas. Se denominan clips de películas. Esto nos puede permitir reutilizar mucho trabajo hecho anteriormente. Por ejemplo, imaginemos que hemos hecho una película en la que una abeja (creada como un clic de película en el que la abeja vuela arriba y abajo) vuela sobre un campo de flores. Podemos crear un nuevo programa de una abeja volando sobre un estanque solo cambiando la imagen de fondo. También podemos copiar y pegar más clips de la abeja de modo que obtendremos una gran cantidad de abejas volando por el campo de flores o el estanque.

Nota

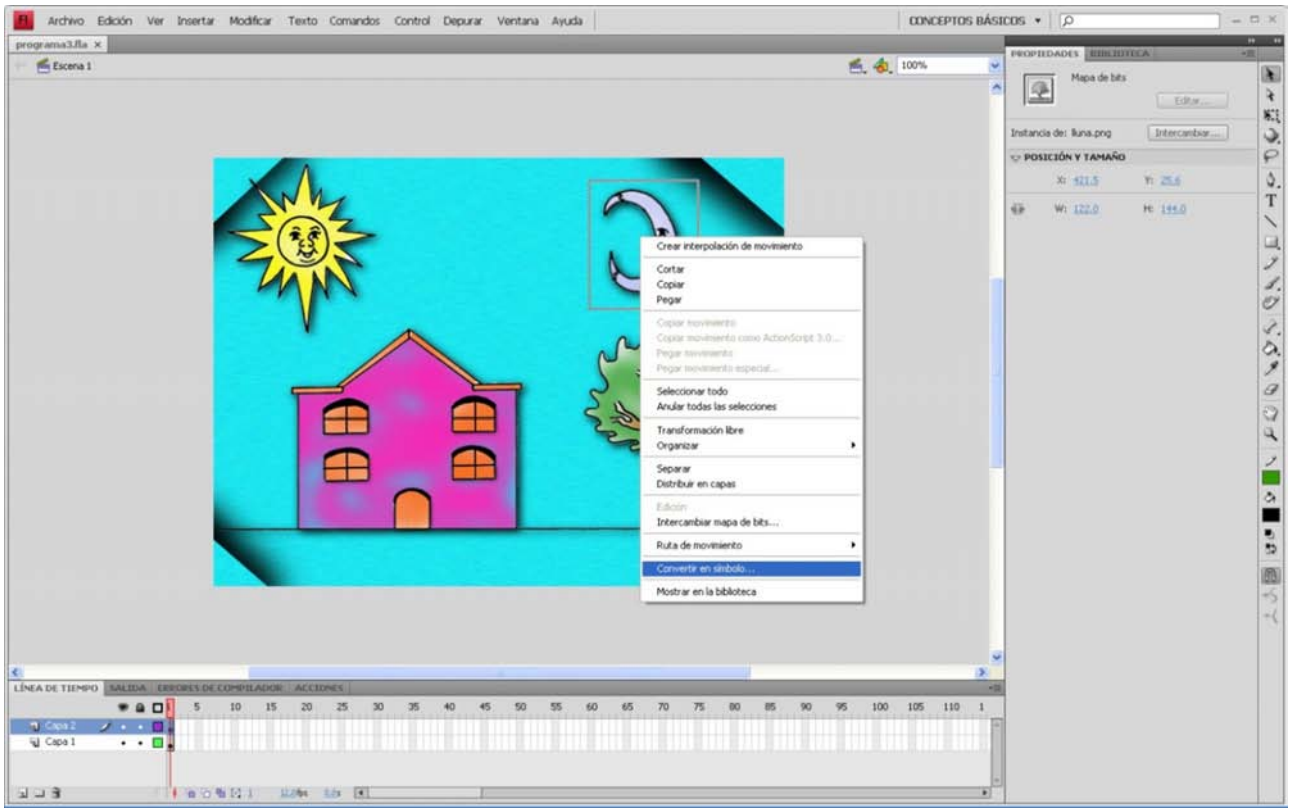
Cambiando las medidas de los nuevos clips, conseguiremos más sensación de profundidad.

3.2. Instrucciones para crear un clip de película

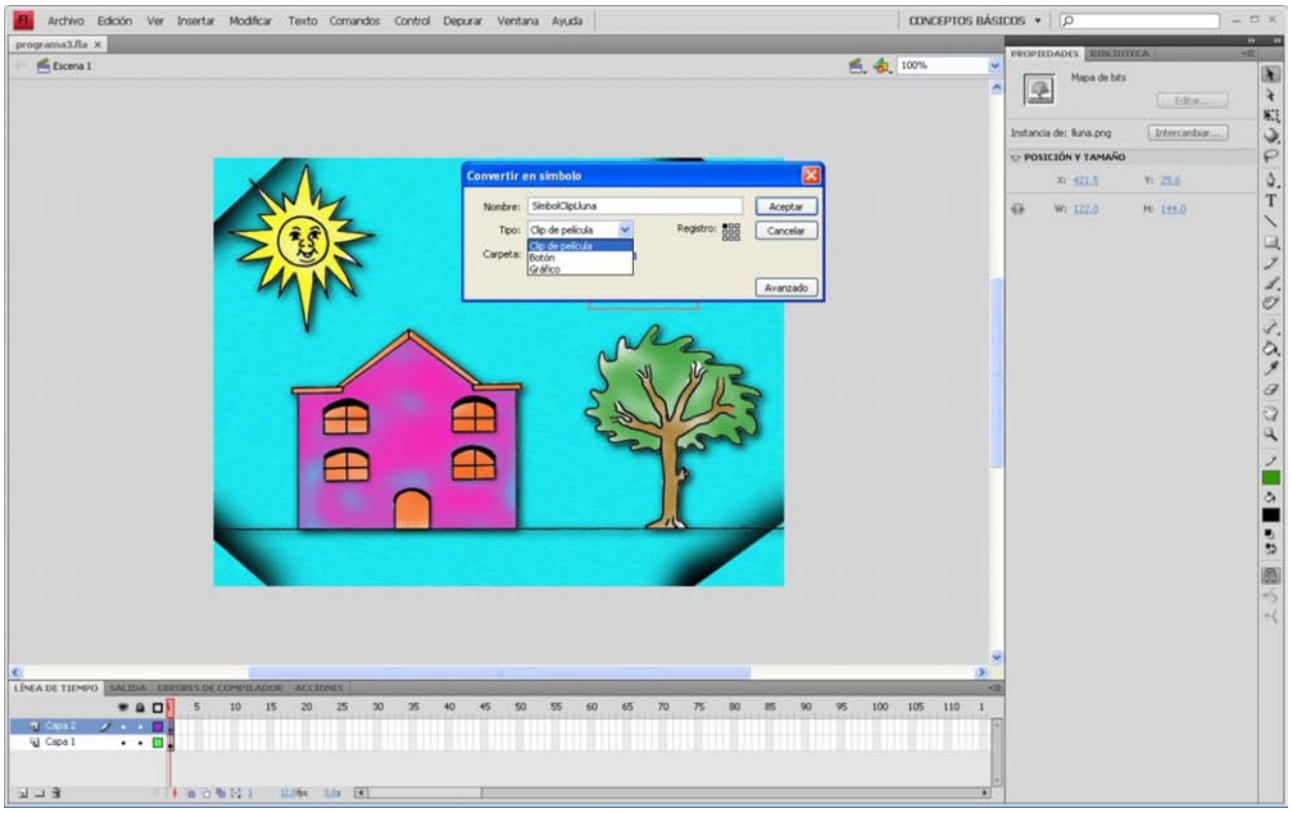
1) Primero. Abrid el programa2 y guardadlo con el nombre que queráis.

Por ejemplo, programa3.

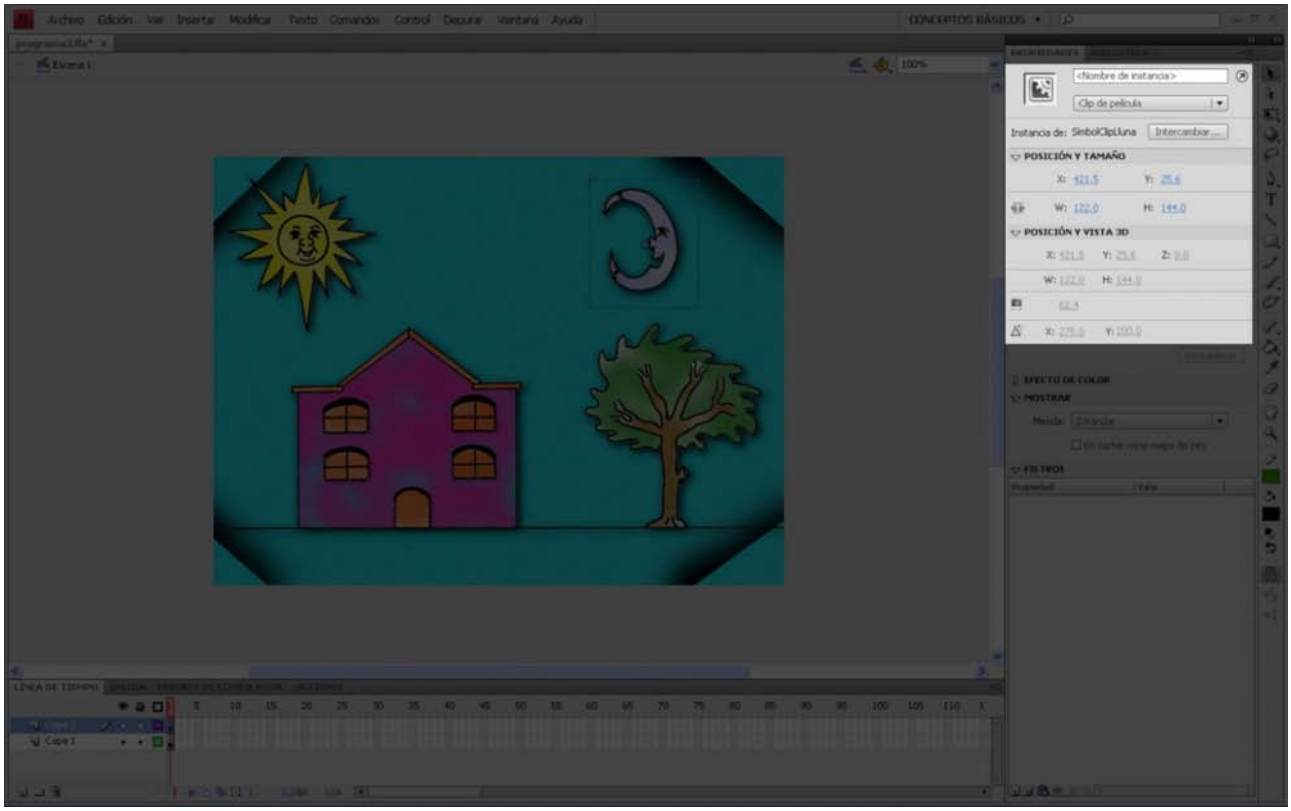
2) Segundo. Situad el ratón sobre la luna y seleccionad el botón derecho del ratón. Observad que aparece un menú contextual. Haced clic sobre el apartado *Convertir en símbolo* para convertir la imagen en símbolo clip de película.



3) Tercero. Aparece una ventana de diálogo. Escribid el nombre simboloClipLuna para tener un nombre que identifique la película.

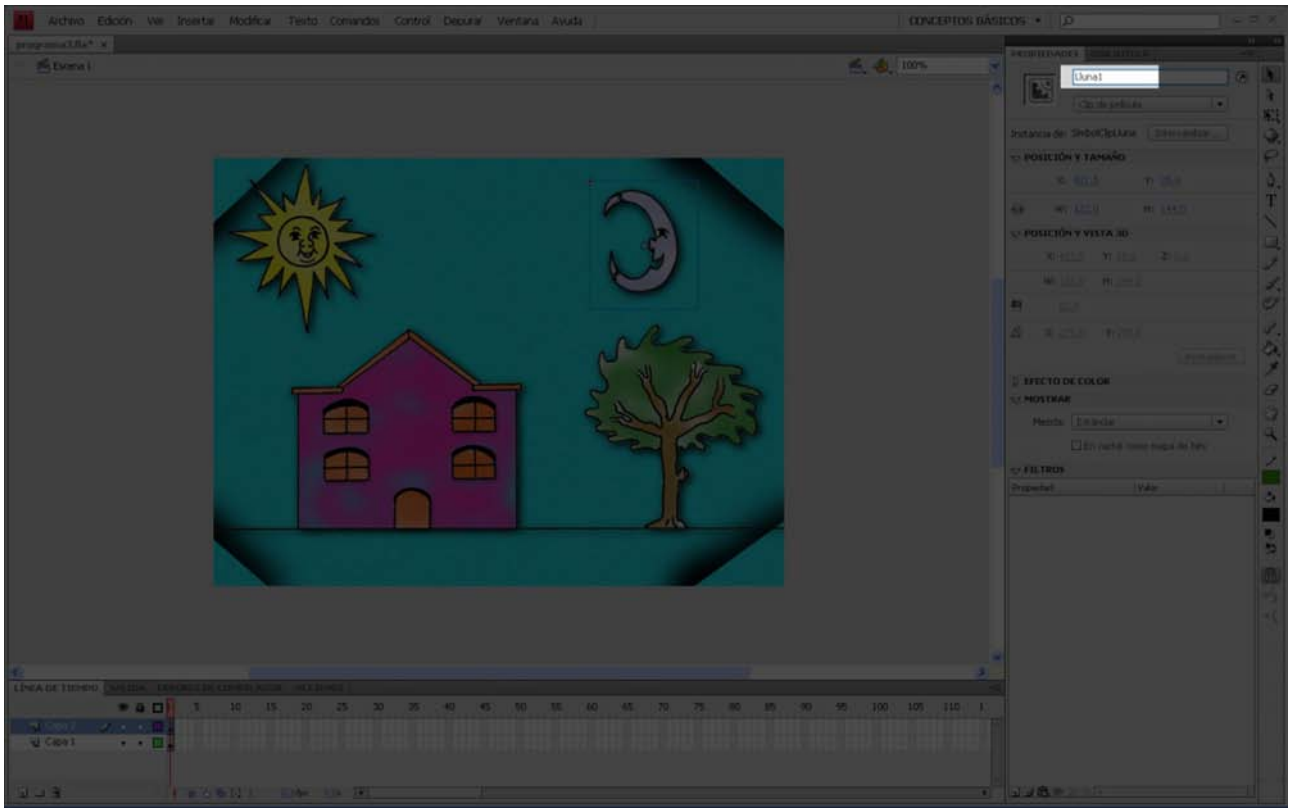


4) Cuarto. Observad que una vez terminado el diálogo, aparecen a la derecha las propiedades de una película *Luna* que ha sustituido la imagen de la luna.



Podéis ver que *W* es la anchura de la película y *H*, la altura (de *width* y *height*).

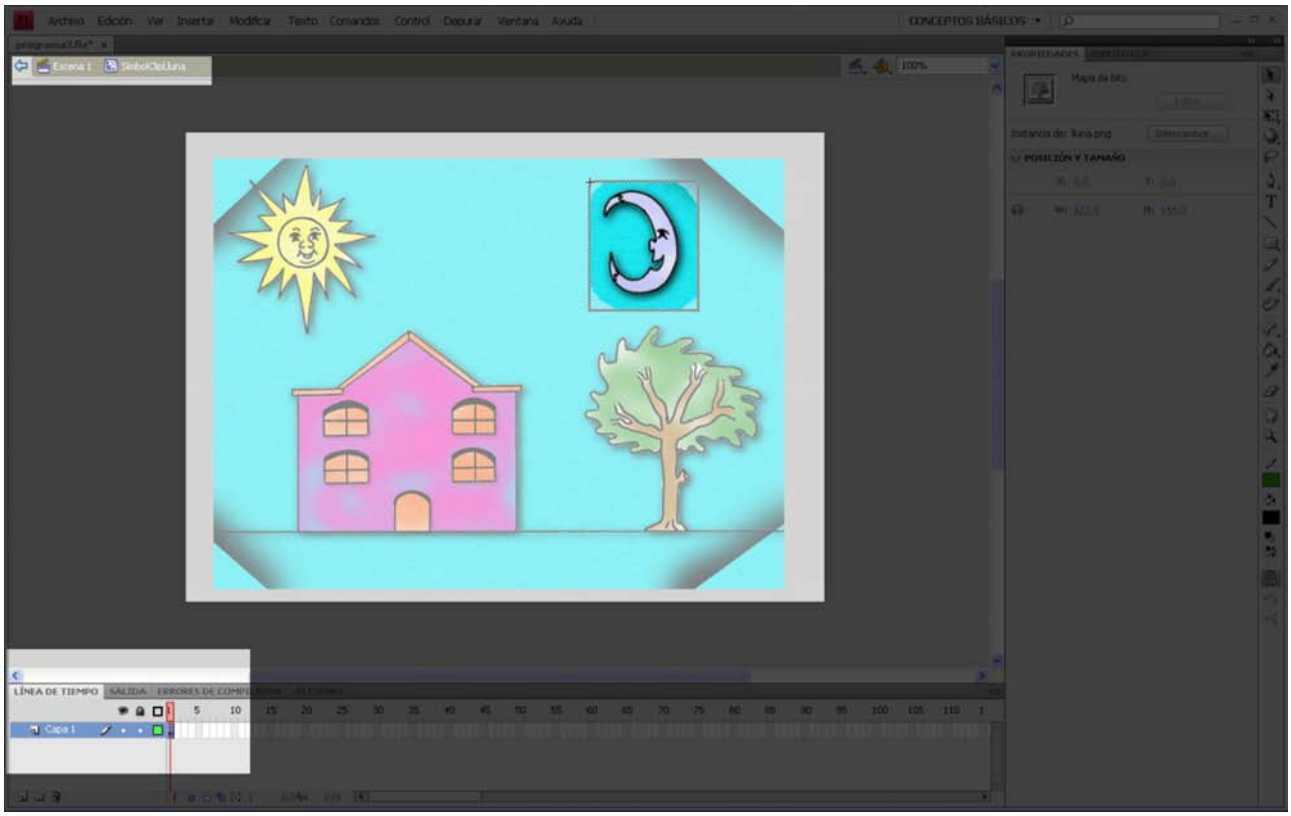
5) Quinto. Poned el nombre luna1 al clip de película.



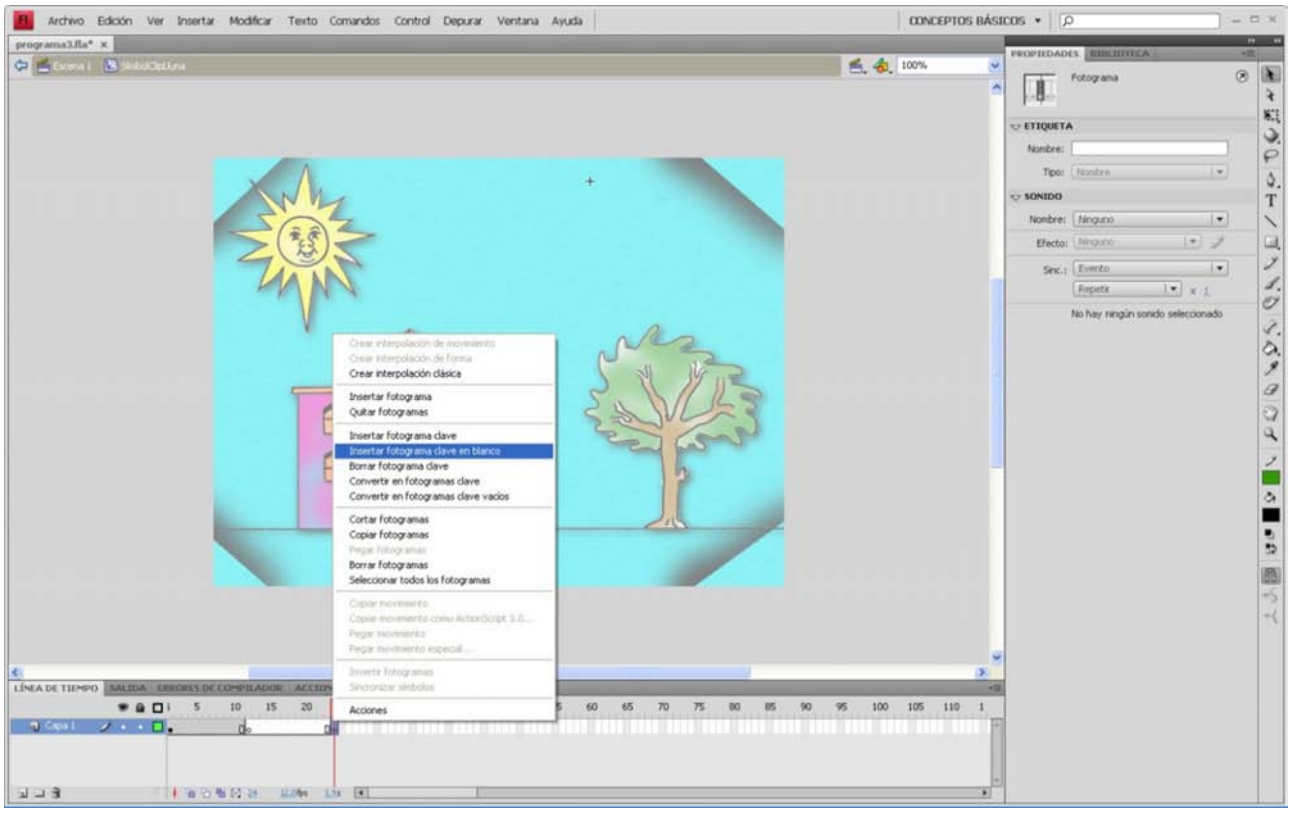
Nota

Si copiamos y pegamos este clip de película, vamos añadiendo copias del simboloClipLuna al escenario. Recordad ir cambiando el nombre a cada película copiada para tener bien referenciados todos los objetos que aparecen en el escenario.

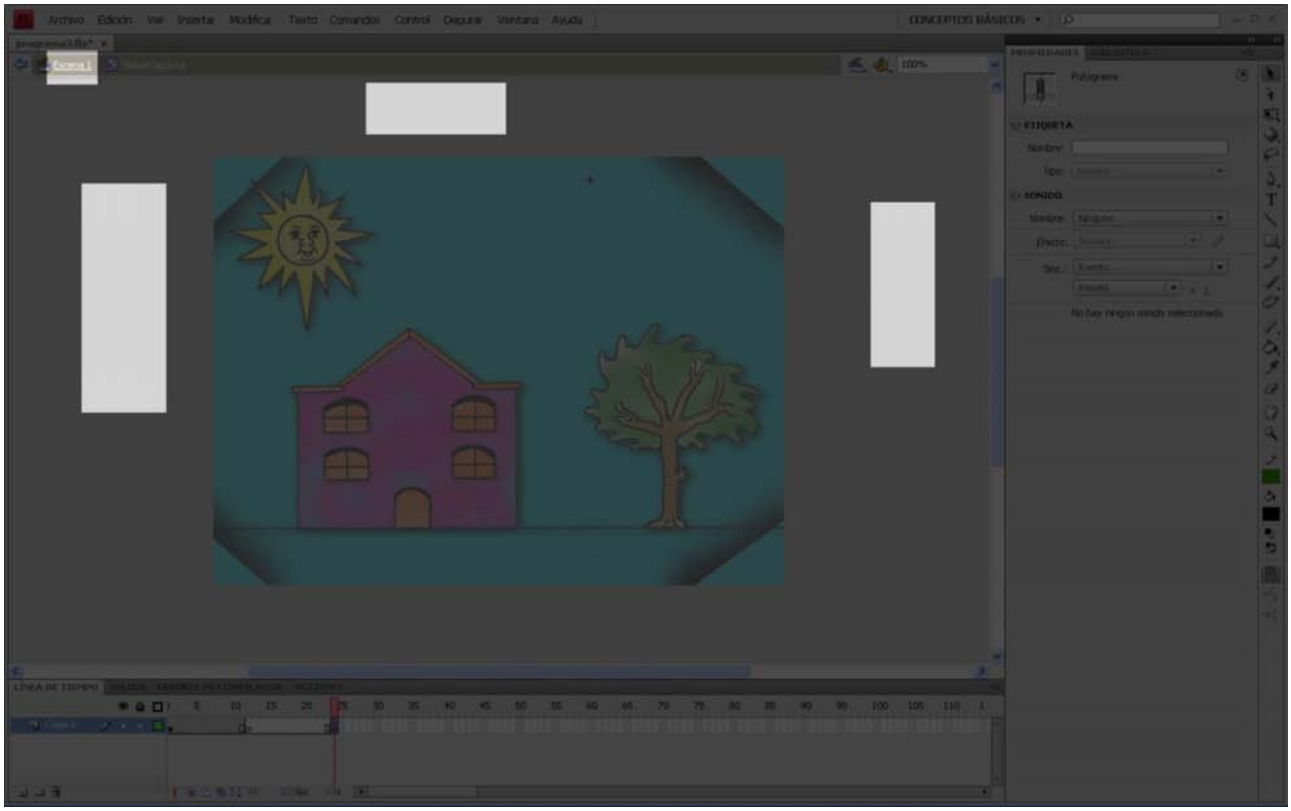
6) Sexto. Haced doble clic sobre *luna1*. Observad que ahora estáis en el escenario simboloClipLuna. La parte del escenario que no forma parte de la luna aparece con una transparencia para diferenciar lo que es del clip simboloClipLuna de lo que no lo es. Observad también que la línea de tiempo es completamente distinta de la película original, puesto que ahora tenemos solo una capa y no dos.



7) Séptimo. Insertad en el fotograma 13 un fotograma clave en blanco y en el 24, otro fotograma clave en blanco (recordad: botón derecho del ratón).



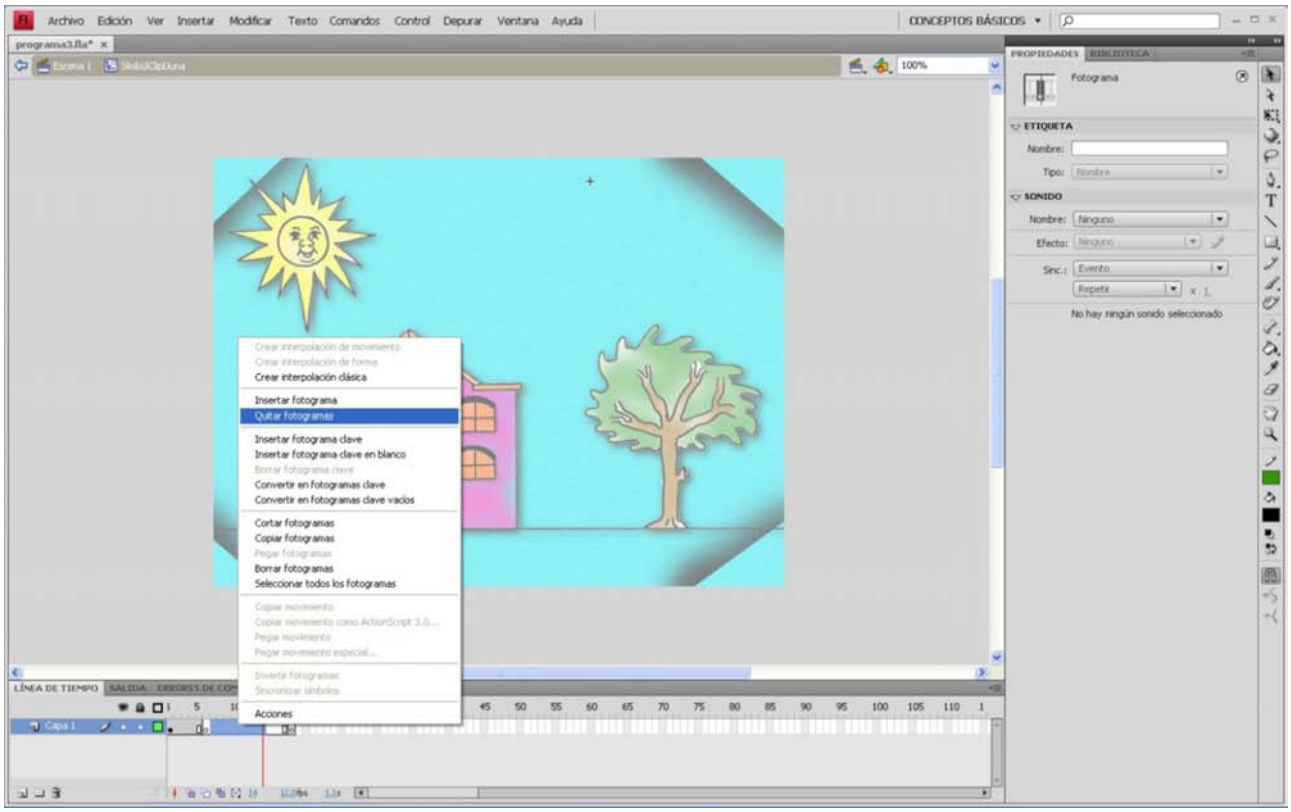
8) Octavo. Haced clic sobre el texto *Escena 1* o doble clip sobre algún lado fuera del escenario para volver a la película principal.



Observad que la película principal solo tiene un marco, en lugar de la película Luna, que tiene 24.

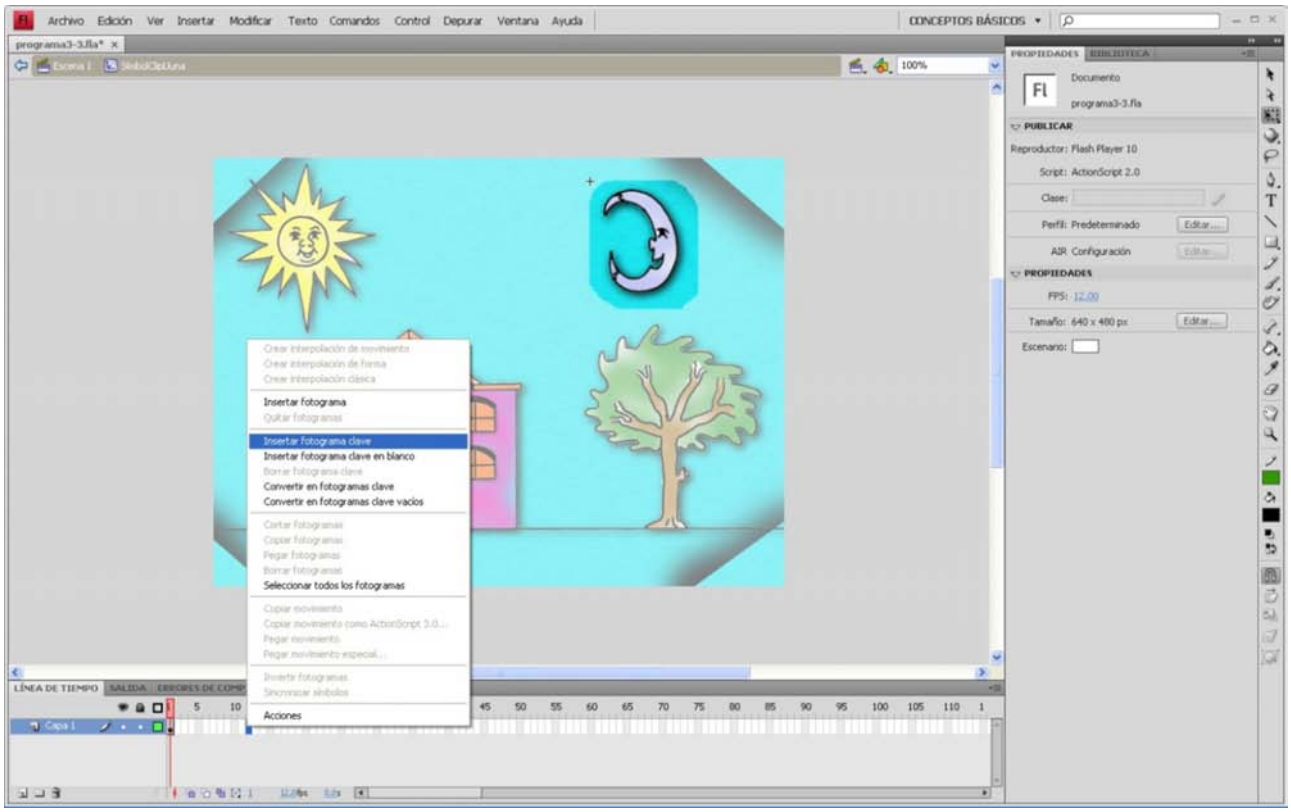
9) Noveno. Si ejecutáis el programa, veréis un efecto de parpadeo en la Luna.

Si creéis que es demasiado lento, podéis eliminar fotogramas de la película SimboloClipLuna seleccionando los fotogramas y, después, la opción *Quitar fotogramas* del menú del botón derecho del ratón.

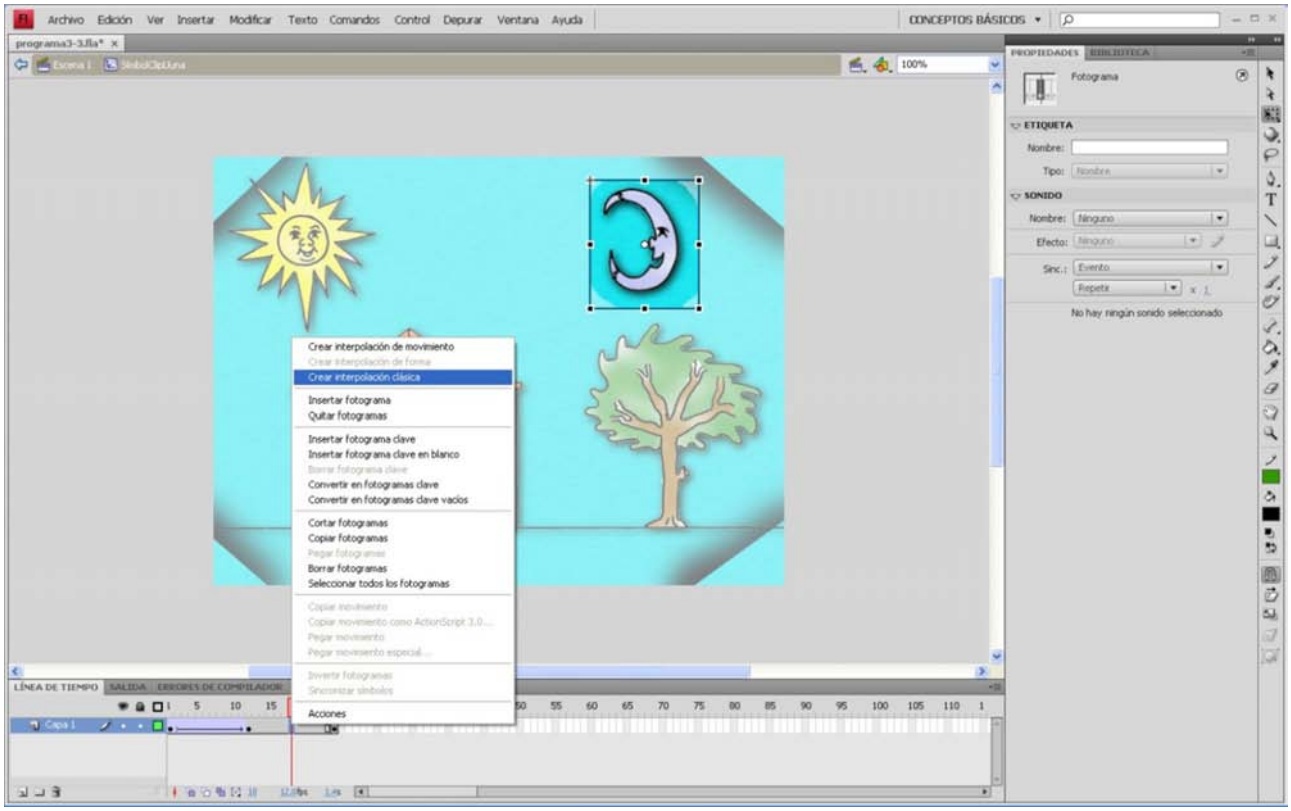


Personalmente, el parpadeo es un efecto muy molesto. A pesar de que cumple perfectamente el objetivo de centrar la atención en el objeto que parpadea, hay otras formas de centrar la atención en los objetos sin ofender tanto la percepción. A modo de ejemplo, para ver otros efectos para centrar la atención en un objeto, a continuación os mostramos el efecto de agrandar o disminuir el objeto de manera cíclica.

Volved a abrir el clip simboloClipLuna y eliminad todos los fotogramas menos el primero. Seleccionad el fotograma 12 y la opción *Inserta fotograma clave*.

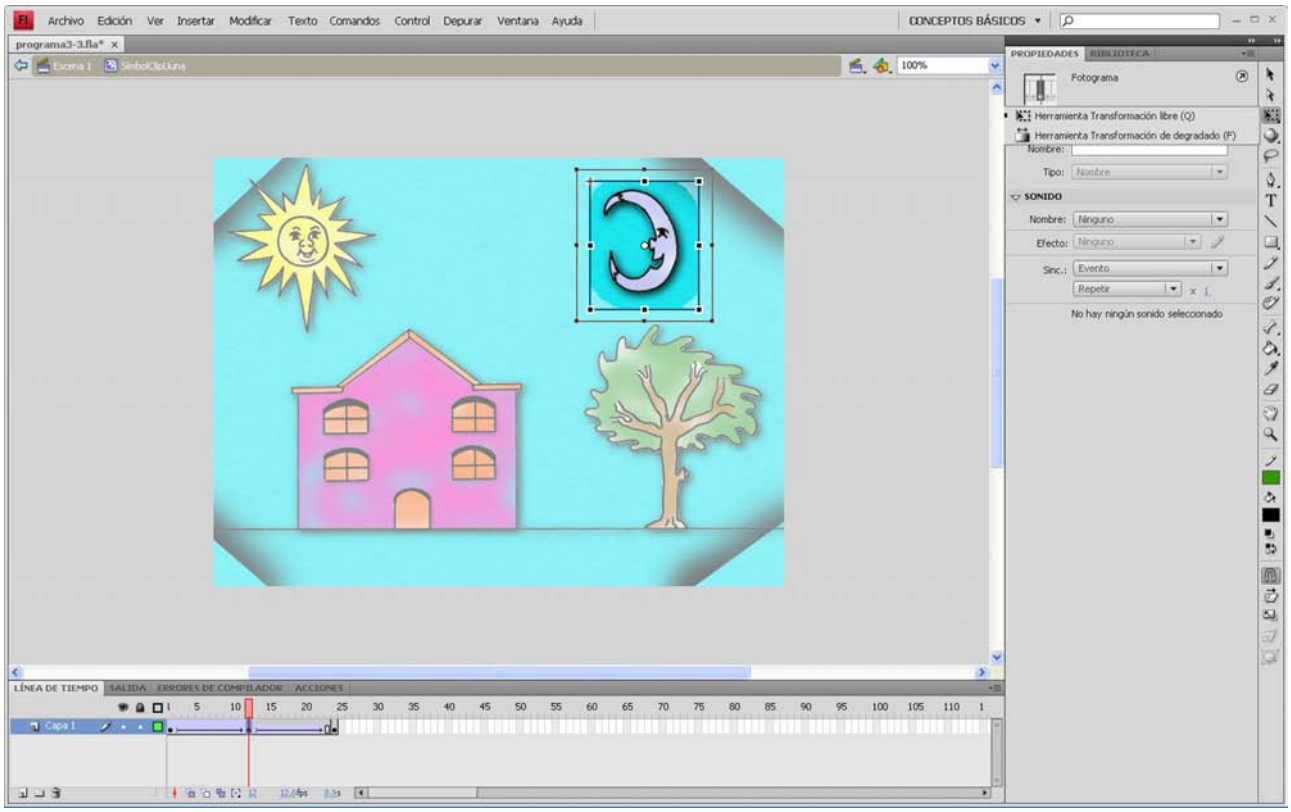


Volved a seleccionar el fotograma 24 y la opción *Inserta fotograma clave*. Os podéis poner en el fotograma 5 o 6 y seleccionar la opción *Crea interpolación clásica*. Después, os ponéis en el fotograma 15 o 16 y volvéis a seleccionar la opción *Crea interpolación clásica*.



Observad que aparecen unas flechas en los fotogramas que nos indican que hay una interpolación.

Os debéis poner en el fotograma 13. Con la herramienta de transformación libre, disminuid un poco el tamaño de la luna.



Ejecutad el programa para ver el efecto creado.

Seguramente os habéis dado cuenta, con este último ejemplo, de la importancia que tienen las transparencias completas alrededor de los objetos, puesto que, si no, el fondo del objeto puede provocar efectos que afean la película.

3.3. Actividades

Actividad 1

Copiad y pegad en el escenario del programa 3 el clip de nombre Luna1. Cambiad el nombre del clip copiado por el de Luna2 y observad el efecto de parpadeo al mismo tiempo.

Haced doble clic en el clip de película Luna1 para poder modificar el clip. Haced algún cambio en el clip (por ejemplo, disminuir el tiempo de parpadeo). Observad que los cambios se aplican a todas las lunas y no solo a la luna

que habéis modificado. Esto se debe de al hecho de que, en realidad, vosotros habéis modificado el clipSimboloLuna y no el clip con nombre Luna1. Salid sin guardar los cambios.

Actividad 2

Copiad y pegad en el escenario del programa 3 el clip con nombre Luna1. Cambiad el nombre del clip copiado por el de Luna2 y observad el efecto de parpadeo al mismo tiempo.

Haced un solo clic en el clip de película Luna1 del escenario para modificar las propiedades del clip (por ejemplo, cambiad las propiedades *H* y *W*). Observad que el cambio se aplica solo al clip de película Luna1. Esto se debe al hecho de que ahora vosotros habéis modificado el clip Luna1 y no el clip SimboloLuna, y por lo tanto los cambios solo se aplican al clip de película seleccionado.

Actividad 3

Cread un campo de flores lleno de flores que se mueven con el viento siguiendo las ideas de este apartado. Cada flor debe ser una instancia de un clip de película simboloClipFlor. Modificad las medidas de las flores.

Actividad 4

Sobre el campo de flores anterior, dibujad un sol en una capa nueva. Convertidlo en un símbolo clip de película. Con la interpolación de movimiento, intentad que haga un recorrido por el cielo.

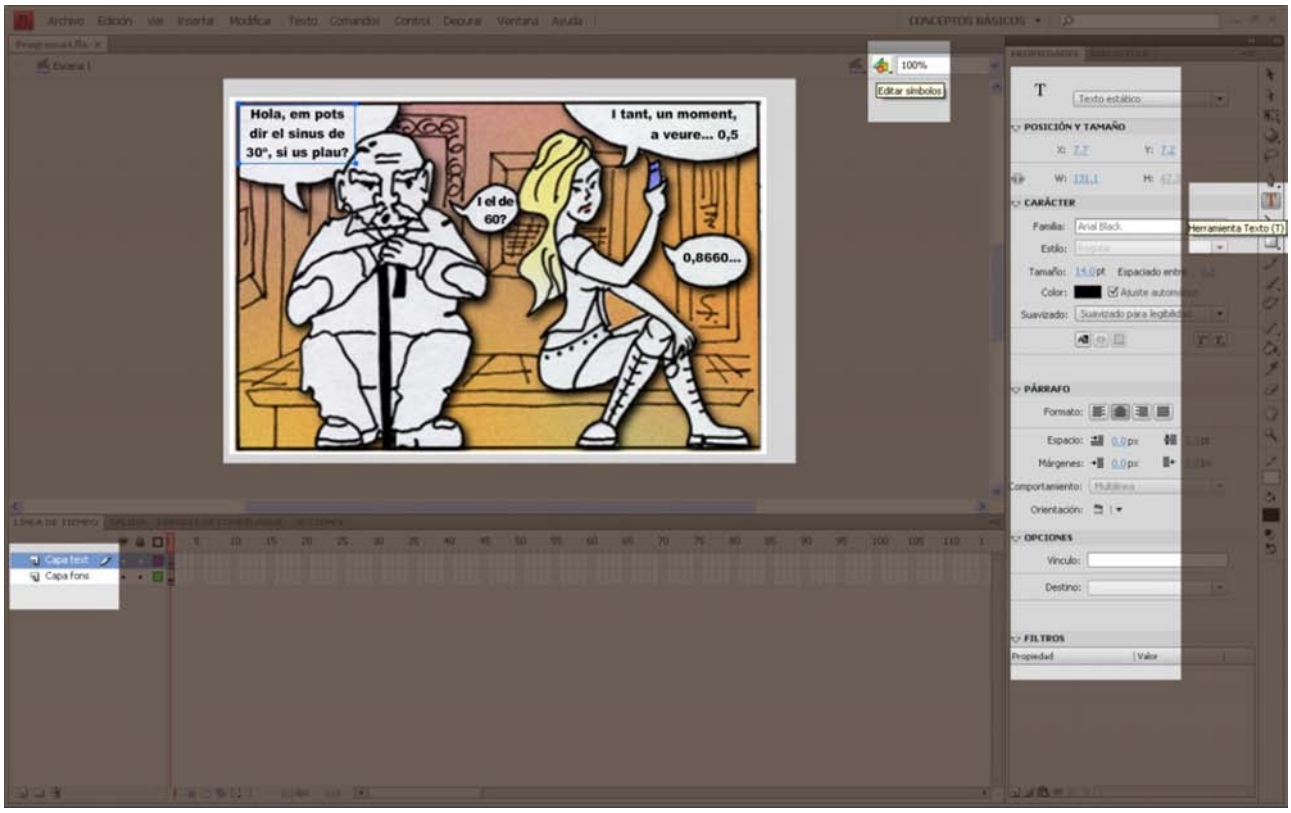
4. Botones y campos de texto

4.1. Introducción

Flash permite insertar textos dentro del programa con la herramienta texto (T).

En el programa 4, tenéis un ejemplo que podéis abrir para ver cómo se trabaja en este sentido.

Hay que comentar, sin embargo, que los textos estáticos no son el único tipo de texto con el que trabaja Flash. Aparte, ofrece la posibilidad de introducir texto dinámico y texto de introducción. Este hecho dota a Flash de la capacidad para comunicarnos con el usuario.



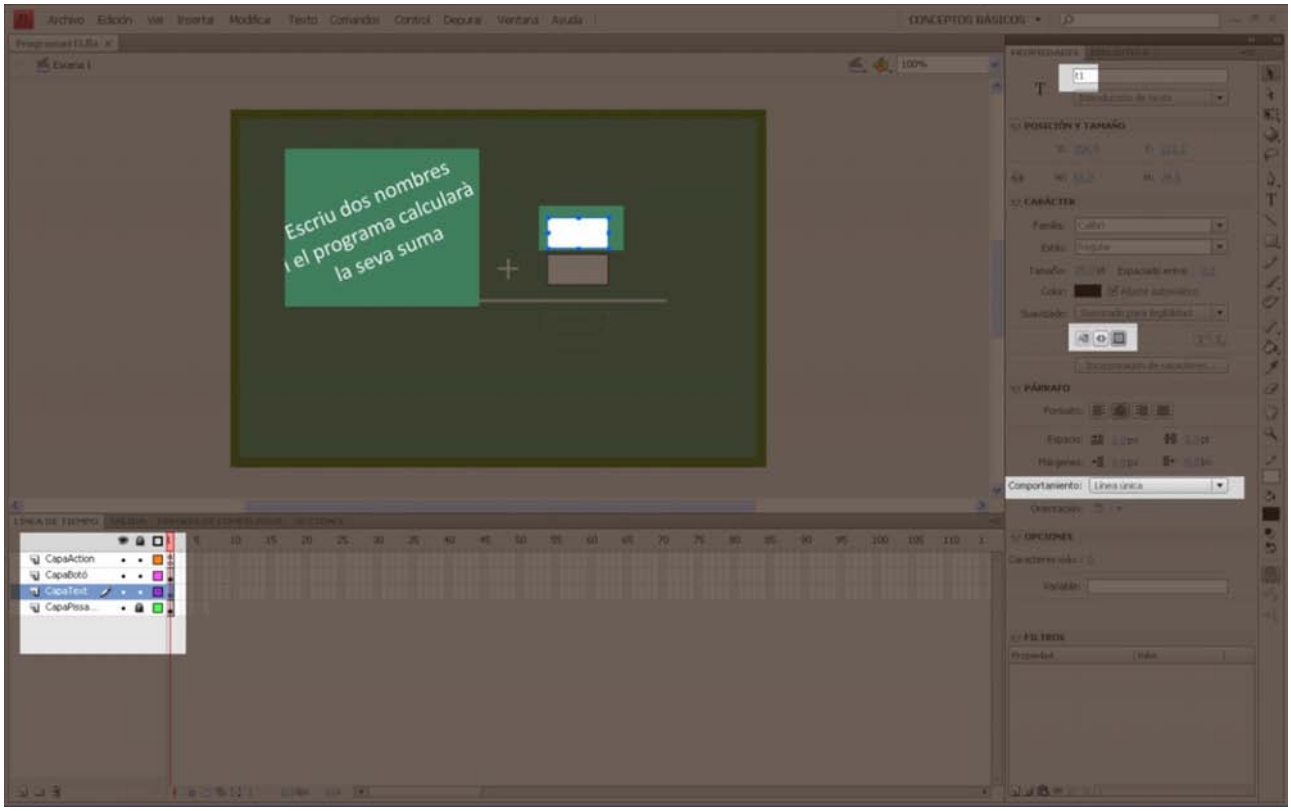
Nota

Aprovechamos este ejemplo para decir que se pueden cambiar los nombres de las capas haciendo doble clic encima.

También aprovechamos para explicar una nueva manera de editar los símbolos, además de hacer doble clic sobre los mismos. La idea es hacer clic sobre la pestaña *Editar símbolos*, destacada en la parte superior derecha de la imagen.

4.2. Un programa con un botón y los tres tipos de cajas de texto

Imaginemos que queremos crear un programa que calcule la suma de dos números introducidos por un usuario.



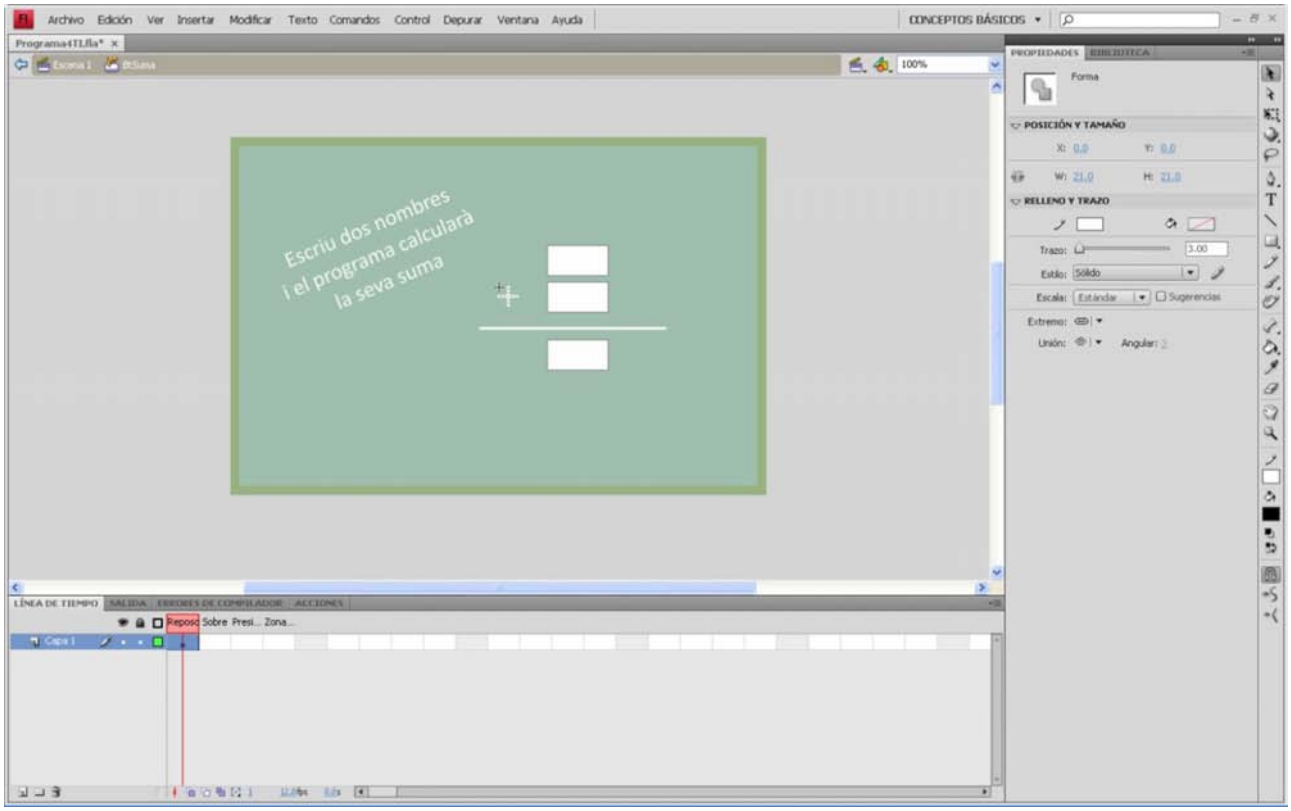
En el programa4TI, encontraréis un ejemplo de interfaz formado por los elementos siguientes.

- Una pizarra como fondo creada en una capa denominada capa Pizarra.
- Un texto estático en el que podemos leer: “Escribe dos números y el programa calculará su suma”.
- Dos textos de introducción (cuidado con elegir línea única y no multilínea, así como con seleccionar el botón para que se visualice un marco a su alrededor y no seleccionar el botón de generar texto como html) con el nombre t1 y t2.
- Un texto[1] dinámico, con el nombre t3, en el que escribiremos los resultados de los dos números introducidos por el usuario.
- Un símbolo[2] tipo botón con aspecto de + que hemos denominado btCalcular. Para crearlo, primero hemos dibujado con la herramienta línea dos líneas que forman una suma. Entonces, seleccionando todo el dibujo, lo

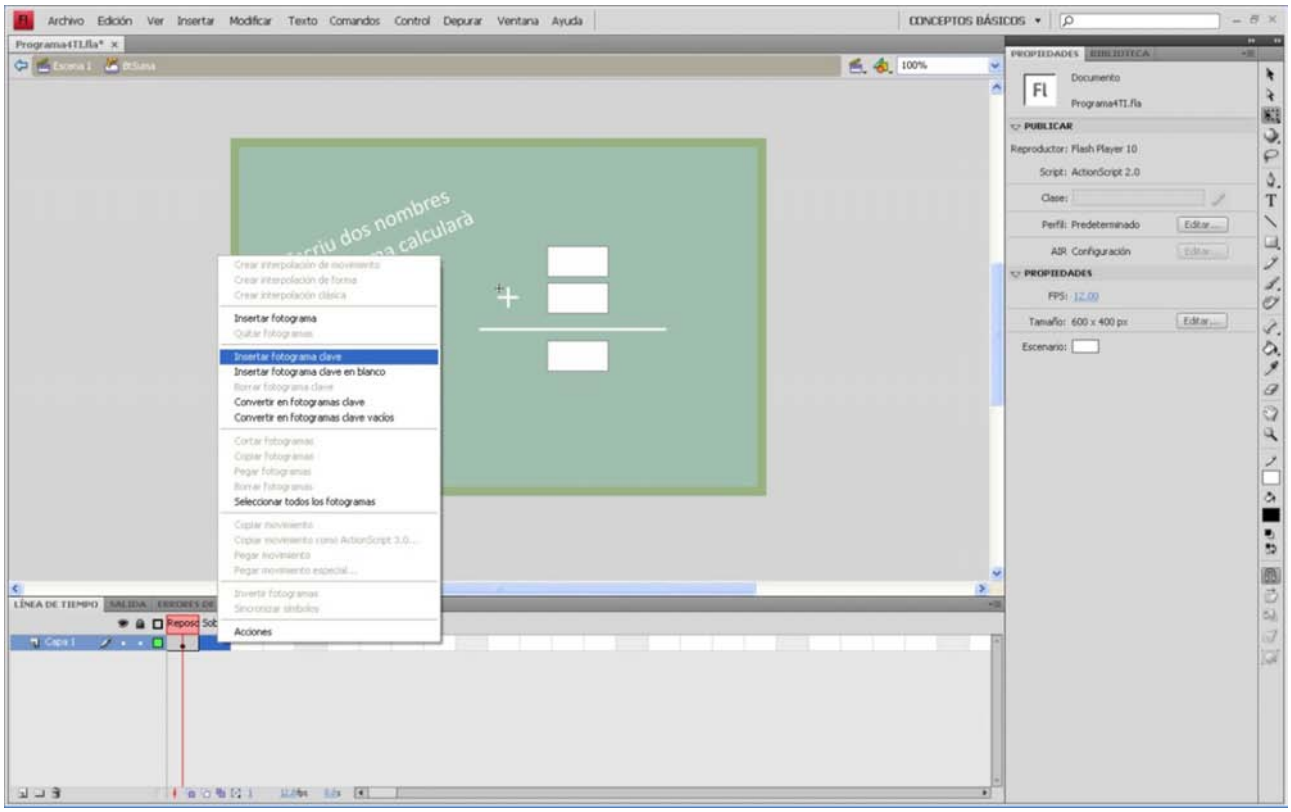
(1)Todos los textos están en la capa de nombre CapaTexto.

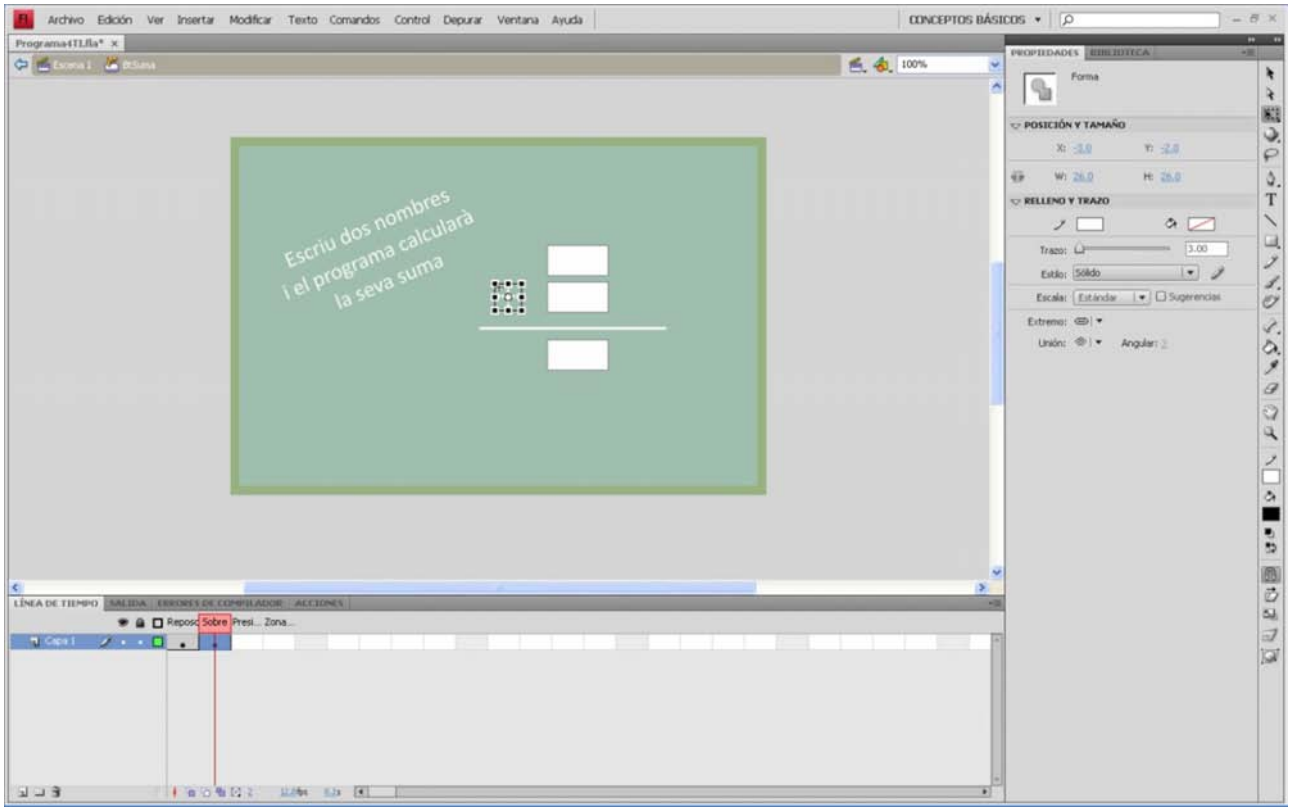
(2)En la capa de nombre CapaBoton.

hemos convertido en un símbolo botón. Si hacemos doble clic encima, podemos editar el botón tal y como vemos en la imagen siguiente.

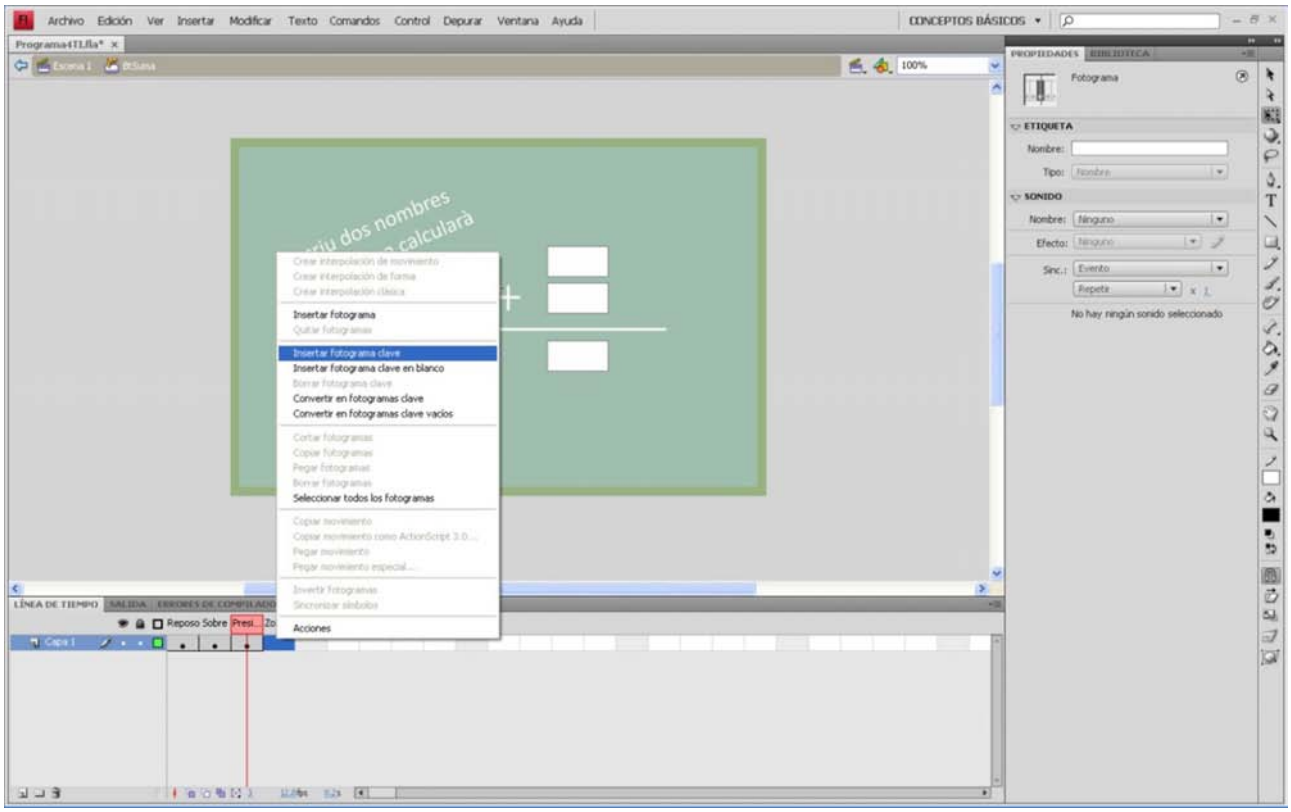


Observemos que el programa utiliza la imagen inicial como fondo del botón en el estado de reposo. Si hacemos clic con el botón derecho del ratón en el marco Sobre y seleccionamos del menú la opción *Insertar fotograma clave*, obtendremos una copia de la imagen inicial en este marco que podremos modificar para tener la imagen del botón cuando estamos encima. Nosotros hemos escalado un poco la imagen para tener un efecto de aumento al pasar por encima.

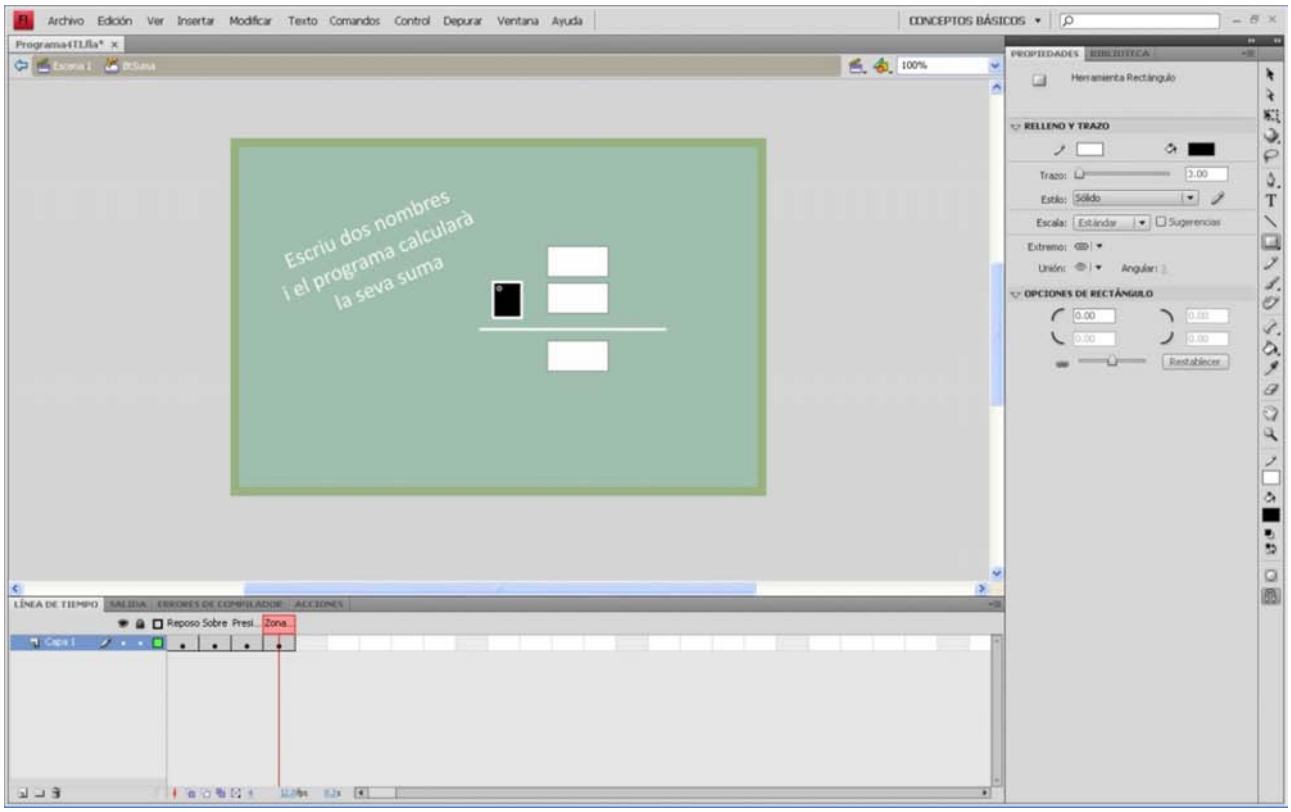




Si volvemos a repetir el proceso, podemos crear la imagen que verá el usuario al hacer clic en el botón. En este caso, no hemos creado ningún efecto.



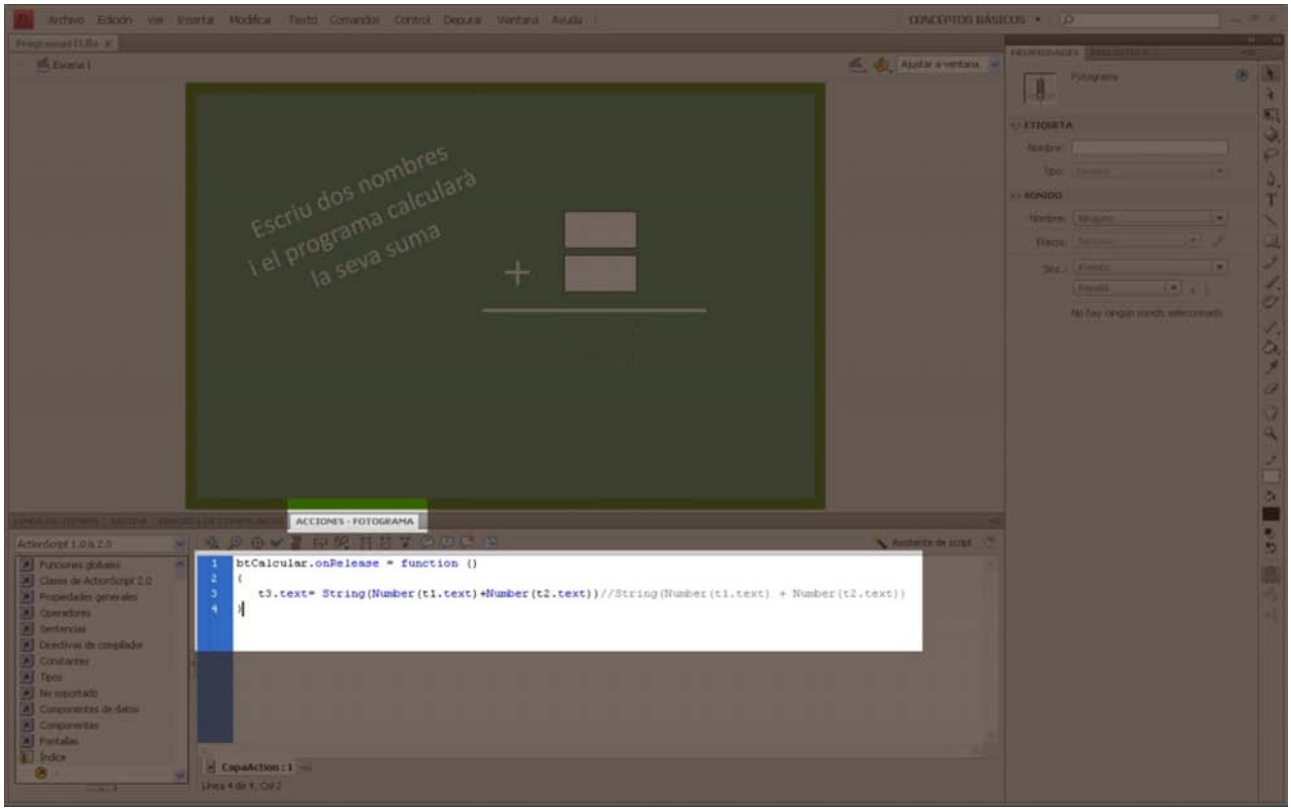
Finalmente, hay que señalar el área alrededor del dibujo inicial que será sensible al ratón. Para esto, creamos un rectángulo que tape la + en el marco Zona.



Para escribir el resultado de la suma de introducción de los dos números, hemos creado una tercera capa que hemos denominado capaAction.

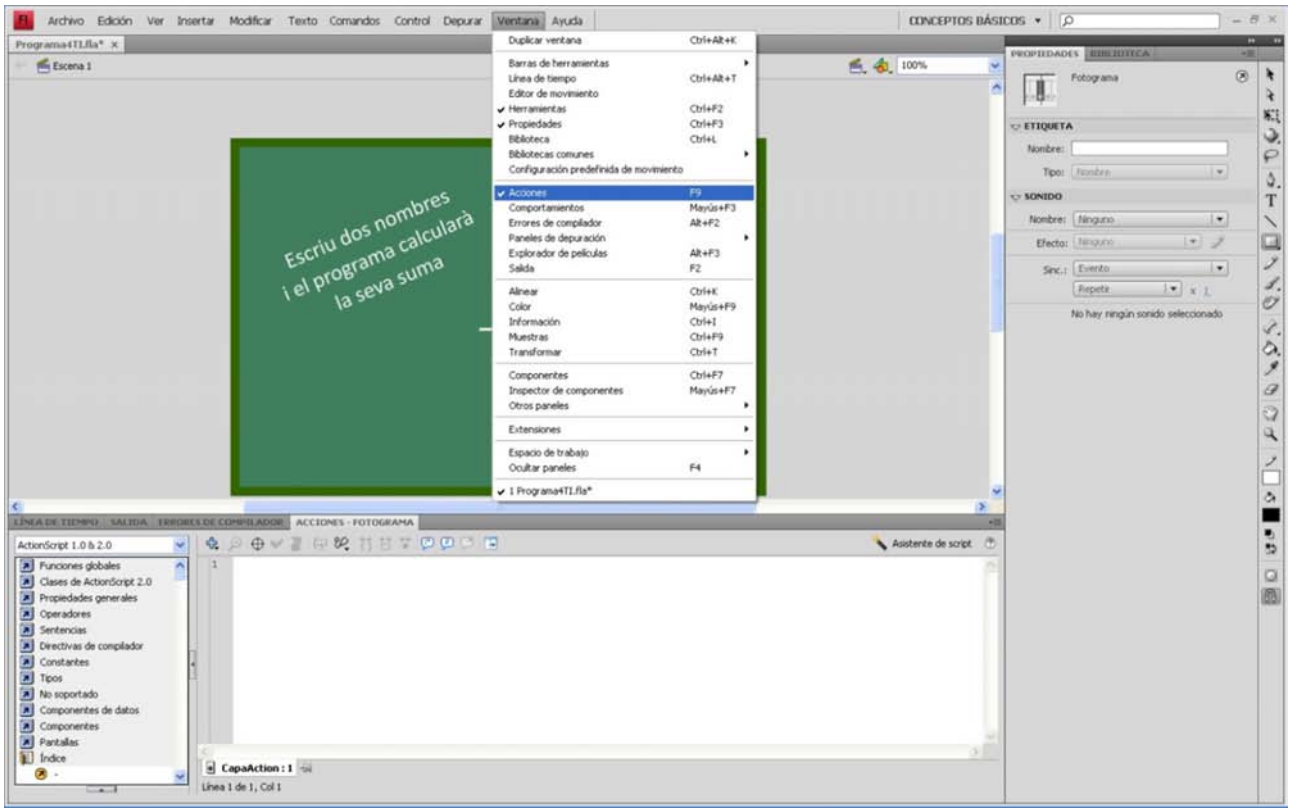
Al seleccionar el marco 1 de esta capa, se ha abierto la ventana *Acciones* y se ha añadido el código siguiente:

```
btCalcular.onRelease = function ()
{
   t3.text= String(Number(t1.text)+Number(t2.text))//String(Number(t1.text) + Number(t2.text))
}
```



Nota

Si no tenéis la ventana *Acciones*, abrid el menú de la pestaña *Ventanas* y haced clic en *Acciones*.



Una vez publicado el programa, si el usuario introduce dos números y hace clic sobre la +, el programa muestra el valor de esta suma en el campo de texto con el nombre t3.

Nota muy importante

Hay que tener presente que es posible que si elegimos fuentes propias y el usuario no las tiene instaladas en su ordenador, puede suceder que el programa no acabe de funcionar bien. Si os pasa algo de este tipo, podéis seleccionar para los textos fuentes del dispositivo en lugar de suavizado para legibilidad, y cambiar la familia a una estándar del sistema, como por ejemplo Arial. Otros problemas pueden surgir al elegir en la configuración de la publicación un Flash Player "muy" actual. Pensad que los usuarios no suelen ir a la última. Por este motivo, no solemos recomendar la elección de la última versión de Flash Player a la hora de publicar el programa.

4.3. Actividades

Actividad 1

A lo largo de los apuntes, hemos visto tres tipos diferentes de símbolos: los clips de película, los botones y los gráficos. Utilizad Google para encontrar las diferencias entre estos y cuándo es recomendable utilizarlos.

5. Action

5.1. Introducción

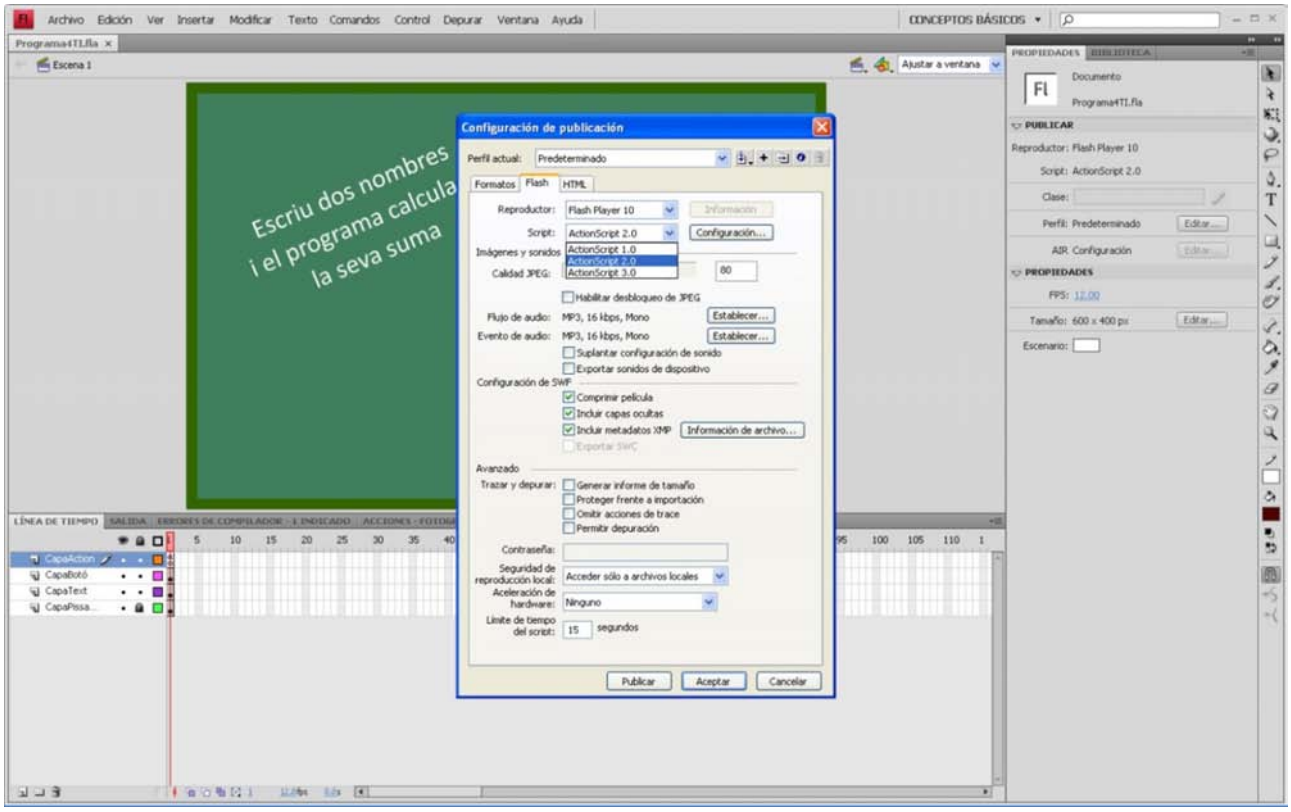
ActionScript es el lenguaje de programación de la plataforma Adobe Flash.

Actualmente, la mayoría de los programadores ya trabajan en la versión 3, pero no está de más aprender un poco de la 2 por si debemos modificar algún programa antiguo.

Solo hay una opción para aprender programación, y es practicando y practicando. Nuestra intención para este apartado consiste en presentar unas primeras nociones con todas las funciones que se pueden utilizar en este primer curso de Matemáticas.

5.2. Unas ideas de Action

Si abrís el programa4TI y vais *a Archivo -> Configuración de publicación* y hacéis clic sobre el menú en la ventana Flash, veréis que el programa ha sido creado en actionScript 2.



En el código acciones del botón del programa4TI, encontramos las expresiones Number() y String(). Podéis observar en el resumen de código siguiente que Number() es una función con la capacidad de convertir un texto que contiene un número en una variable que el ordenador ya entiende que es un número con el que podrá trabajar de manera matemática. Si no lo convertimos, hay la opción de que el programa, en lugar de sumar, concatene las expresiones pensando que son palabras.

Resumen de propiedades y métodos de variables de texto y numéricas

Imaginemos que tenemos tres variables *a*, *vTexto* y *b*, en las que *a* y *vTexto* son de tipo texto y *b* es de tipo numérico.

```
var a:String
var vText:String
var b:Number
```

Nota

Observad que las variables pueden tener nombre de letras o palabras y combinar mayúsculas y minúsculas.

Conversión de números en frases y de frases en números

Si tenemos *b* = 234 y queremos que *a* sea la frase *a* = 234, podemos hacer:

```
a=String(b)
```

A partir de ahora, en *a* tenemos la frase 234 y podemos hacer métodos relacionados con cadenas de texto.

Si sabemos que *a* = 123, es decir, es una frase que se puede interpretar como un número, podemos hacer:

```
b=Number(a)
```

Entonces, en *b* tenemos un número y podemos hacer métodos relacionados con números.

Métodos de las variables numéricas

Generalmente, para trabajar con números y variables numéricas utilizaremos una construcción denominada clase Math. Esta construcción contiene métodos y constantes que representan funciones y valores matemáticos comunes. Podemos imaginar esta construcción como un saco donde hay guardados números especiales y funciones que hacen cosas. Para utilizarlas, solo tenemos que decir que tomamos el saco y después sacamos de dentro el número o la función que necesitemos para nuestro trabajo. Por ejemplo:

```
b=Math.min(4,5) //En b guardamos el mínimo de 4 o 5
b=Math.max(4,5) //En b guardamos el máximo de 4 o 5
b=Math.pow(2,3) //En b guardamos 2 elevado a 3 = 8 (= 2^3)
b=Math.floor(2.345) //En b guardamos el valor redondeado a la baja que es 2
b=Math.random() // En b guardamos un número entre 0 y 1 aleatorio
b=Math.abs(-3) //En b guardamos el valor absoluto del número
b=4+2*2-4/2 // En b guardamos el valor 4+4-2=6
b=5 %2 // En b guardamos el resto de la división de 5 por 2, que es 1 (relacionado con módulo)
b=Math.Pi // En b guardamos el valor de Pi. En realidad guardamos una aproximación, ya que Pi
          // es un número con infinitos decimales imposible de ser determinado totalmente.
```

Nota para cuando tengamos que trabajar con decimales

En Flash, los números decimales se tienen que poner con puntos y no con comas.

Métodos de las variables cadenas de texto

En las variables de tipo texto, podemos aplicar directamente funciones especiales desarrolladas de manera expresa por los creadores de Flash para operar con este tipo de clase. Por ejemplo, *b = a.length* guarda en *b* la longitud o el número de caracteres de la frase guardada en *a*. De este modo, si *a* contiene la palabra *vaca*, en *b* obtendremos el valor numérico 4.

Veamos otros tipos de funciones especiales:

```
a=vText.substr(0,1) En a guardamos la primera letra de la frase guardada en texto
a=vText.substr(1,1) //En a guardamos la segunda letra de la frase guardada en texto
a=vText.substr(2,1) //En a guardamos la tercera letra de la frase guardada en texto
a=vText.substr(3,1) //En a guardamos la cuarta letra de la frase guardada en texto
a=vText.substr(4,1) //En a guardamos la quinta letra de la frase guardada en texto
...
a=vText.substr(0,2) //En a guardamos la primera y segunda letra de la frase guardada en texto
a=vText.substr(0,3) //En a guardamos la primera, segunda y tercera  letra de la frase guardada
                    en texto
...
a=vText.substr(1,2) //En a guardamos la segunda y tercera  letra de la frase guardada en texto
a=vText.substr(2,2) //En a guardamos la tercera y cuarta  letra de la frase guardada en texto
a=vText.charAt(0) //En a guardamos la primera letra de la frase guardada en texto
a=vText.charAt(1) //En a guardamos la segunda letra de la frase guardada en texto
a=vText.charAt(2) //En a guardamos la tercera letra de la frase guardada en texto
a=vText.charAt(3) //En a guardamos la cuarta letra de la frase guardada en texto
a=vText.charCodeAt(0) //En a guardamos el código en decimal ascii de la primera letra de la frase
                      guardada en texto
a=vText.charCodeAt(1) //En a guardamos el código en decimal ascii de la segunda letra de la frase
a=vtext.charCodeAt(2) //En a guardamos el código en decimal ascii de la tercera letra de la frase
a=vText.charCodeAt(3) //En a guardamos el código en decimal ascii de la cuarta letra de la frase
a=text.toUpperCase()  //En a guardamos la frase guardada en texto en mayúsculas
a=vText. toLowerCase ()  //En a guardamos la frase guardada en texto en minúsculas
```

Consideraciones de sumar o concatenar variables

a) Imaginemos tres variables de texto *a*, *b*, *c*

Si en *a* tenemos a Pepe y en *b* tenemos a Ana

En c = *a* + *b* (o también *a* & *b*) tendremos la frase PepeAna

En c = *a* + " " + *b* tendremos la frase Pepe Ana

En c = *a* + "quiere a" + *b* tendremos la frase Pepe quiere a Ana

Si en *a* tenemos 5 y en *b* tenemos 4

En c = *a* + *b* tendremos 54

b) Imaginemos tres variables numéricas *a*,*b*, *c*

Si en *a* tenemos 5 y en *b* tenemos 4

En c = *a* + *b* tendremos el número 9

Observad que la suma de números, si están en texto, no es lo mismo que la suma de números si están en variables numéricas. Esto obliga a pensar en qué tipo de variables estamos trabajando. Por este motivo, siempre recomendamos declarar las variables al principio del programa, a pesar de que no es obligatorio.

Conviene saber que si no declaramos las variables y en medio del código de programación hacemos la asignación *a* = 10, automáticamente Flash entiende que *a* es una variable numérica y que tiene guardado el número 10.

La asignación *a* = "10" permite a Flash entender que *a* es una variable de texto y que tiene por valor la expresión o frase 10. Observad que en una variable de texto las comillas desaparecen y, por lo tanto, solo sirven para indicar que *a* es una variable de texto, en lugar de numérica. Por ejemplo, si hacemos *a* = "La luz está encendida", en *a* tenemos guardada la frase *La luz está encendida* que empieza por *L*. Es decir, a.Text.charCodeAt (0) es *L* y no la comilla ".

Recordemos, pues, que para definir una variable de texto o introducir una palabra o frase en una variable de texto se tienen que poner las comillas de manera obligatoria en la expresión aunque, tal y como hemos explicado, la variable no las guarda y, por lo tanto, no las podemos recuperar.

5.3. Cuando un programa "habla"

Para que un programa muestre al usuario un valor final calculado mediante programación, suele haber generalmente[3] un botón que indicará al programa que ejecute ciertas instrucciones y una línea de este código que pida al programa mostrar una información en alguna caja de texto que se visualiza en la pantalla.

[3] Generalmente pero no siempre, puesto que, por ejemplo, podemos hacer que un programa muestre una acción al pasar por encima de un objeto, al arrastrar una imagen, al mantener presionado el botón derecho del ratón, etc.

En el código acciones del botón del programa4TI, la expresión siguiente: btCalcular.onRelease = function () con onRelease detrás de btCalcular indica que efectuaremos las acciones del paréntesis cuando el usuario haga clic sobre el botón con el nombre btCalcular. Con la expresión btCalcular.onMouseUP = function (), las acciones se ejecutarían cuando pasáramos el puntero del ratón por encima del botón btCalcular.

En la expresión: t3.texto = ... (recordad que t3 era el nombre de una caja de texto), en t3 aparece exactamente el texto que hemos querido que el programa calculara.

Nota

Podemos también cambiar el color de la caja de texto t3 modificando un poco el código de la ventana acciones:

```
btCalcular.onRelease = function ()
{
   t3.text= String(Number(t1.text)+Number(t2.text))//String(Number(t1.text)+Number(t2.text))
   t3.textColor = 0x009900;
}
```

Pese al buen funcionamiento del programa, si un usuario introdujera una letra en vez de un número en una de las cajas de introducción, el programa no tendría que ejecutar las instrucciones de la suma, sino que debería avisar al usuario de que no ha introducido correctamente la información solicitada. Proteger un programa de la "inconsciencia" del usuario es uno de los puntos más difíciles que un programador debe intentar. Y decimos "intentar" puesto que os aseguramos que por mucho que nos esforcemos, nunca estaremos a la altura del singular ingenio de algún usuario.

5.4. Actividades

Actividad 1

Buscad chistes de informáticos por la Red y prestad especial atención a las leyes de Murphy para la informática.

Actividad 2

Cread un nuevo archivo en Flash (AS 3.0) formado por un texto estático que pida introducir un número, un texto de introducción denominado txtEntrada, un texto dinámico denominado txtResultado y un botón denominado botonCalcular.

Añadid el código siguiente a la acción de hacer clic en el botón:

```
btCalcular.addEventListener(MouseEvent.CLICK, funcionEscribirSiguiente);
function funcionEscribirSiguiente (tipusEvent:MouseEvent):void
{
              var a:Number
a = Number(_root.txtEntrada.text);
if (isNaN(a))
{
        _root.txtResultado.text="Mal, no has introducido un número";
}else {
                                    _root. txtResultado.text= String(a+1);
}
}
```

Buscad en la Web qué hace la expresión isNan() (por ejemplo, introduciendo en Google las palabras *isNan Flash* o directamente en la página de Adobe http://help.adobe.com/es_es/as2lcr/flash_10.0/help.html?content=00000576.html).

¿Qué creéis que hace el programa?

Podéis observar que si lo habéis creado AS 3.0, estáis publicando en ActionScript 3.

Observad las diferencias a la hora de hacer funcionar un botón entre AS 2.0 y AS 3.0.

Actividad 3

Abrid la carpeta **Ejemplos if, while y for**. Encontraréis todo tipo de ejemplos de programas que piden introducir números y ofrecen un resultado. Leed el código en Action y buscad las diferencias de un tipo de programa a otros.

Actividad 4

Abrid la carpeta **Paisaje marítimo**. Encontraréis un programa que hace un dibujo de un paisaje marinero sin utilizar nada más que código actionScript. Observad la construcción de las funciones que ayudan a hacer más reutilizable el código.

Actividad 5

Abrid la carpeta **Cielo Nocturno**. Aquí encontraréis un programa que duplica una película denominada Estrella y la dispone en el escenario de manera aleatoria. Observad el código y buscad la expresión random(). Averiguad cómo podéis utilizar esta función.

Actividad 6

Abrid la carpeta **Burbujas**. Aquí encontraréis un programa que escala unas burbujas y las va elevando de manera ondulada. Observad el código y buscad la expresión gotoAndPlay(). Buscad en los foros de internet qué hace y cómo se utiliza esta función. Buscad también las expresiones gotoAndStop(), Stop() y Play().

Actividad 7

Averiguad el uso de las expresiones _root y _parent.

Actividad 8

Cambiad las configuraciones de publicación de todos los programas del tutorial en archivos con script ActionScript 3.0. Si es necesario, modificad la programación para que los programas funcionen correctamente en AS3.