

Tutorial 6 - Animació amb ActionScript 3.0

Pas 1 de 19

En aquest tutorial farem una primera aproximació a la programació amb ActionScript 3.0, l'última versió del llenguatge de programació del Flash.

Amb aquest llenguatge de programació podem afegir interactivitat a les nostres pel·lícules, controlar l'aspecte o moviment d'objectes en l'escenari, crear aplicacions complexes, etc.

En primer lloc, aprendrem a controlar els clips que hi ha en l'escenari mitjançant programació. En aquest cas controlarem un clip fet a partir del núvol que hem creat en el tutorial 1.



Pas 2 de 19

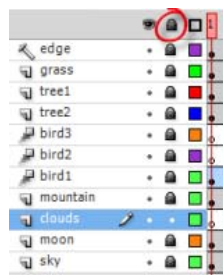
Obrim el document *tutorial1.fla*. Fem clic sobre el núvol amb el **botó dret** del ratolí i seleccionem **Copiar**. Si no podem seleccionar-lo possiblement és perquè tenim bloquejada la capa *cloud* en la línia de temps, així que primer l'haurem de desbloquejar.



Una vegada copiat, ja podem tancar el document sense necessitat de desar els canvis.

Obrim l'arxiu *tutorial5.fla*, que conté l'animació dels ocells, i el dessem com a *tutorial6.fla*.

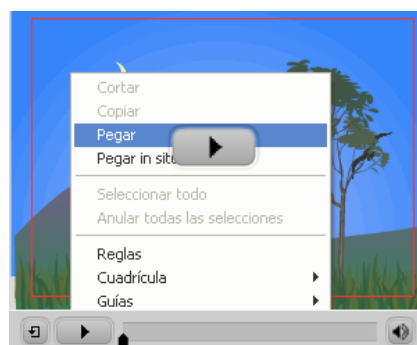
Creem una nova capa per darrere de la muntanya i l'anomenem *clouds*.



Bloquegem la resta de les capes per evitar possibles modificacions accidentals. La manera més ràpida és clicar sobre el cadenat que hi ha sobre totes les capes, i després desbloquejar la capa que ens interessa.

Amb la capa *clouds* activa, seleccionem qualsevol punt de l'escenari amb el **botó dret** del ratolí i seleccionem **Pegar**.

Situem el núvol en la cantonada superior esquerra de l'escenari.



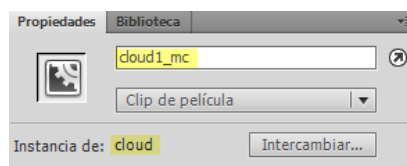
Paso 3 de 19

Amb el núvol seleccionat, premem **F8** o **Modificar > Convertir en símbol**. Li donem el nom *cloud* i com a tipus seleccionem **Clip de pel·lícula**. Situem el punt de registre en l'extrem superior esquerre.



Ara tindrem un clip de pel·lícula anomenat *cloud* en la biblioteca, mentre que en l'escenari tindrem una instància d'aquest clip de pel·lícula, tal com ens indicarà la part superior de l'inspector de **Propiedades**.

Per a poder controlar una instància amb **ActionScript**, el primer que hem de fer és donar-li un nom d'instància en l'inspector de **Propiedades**. Li donarem el nom *cloud1_mc*.



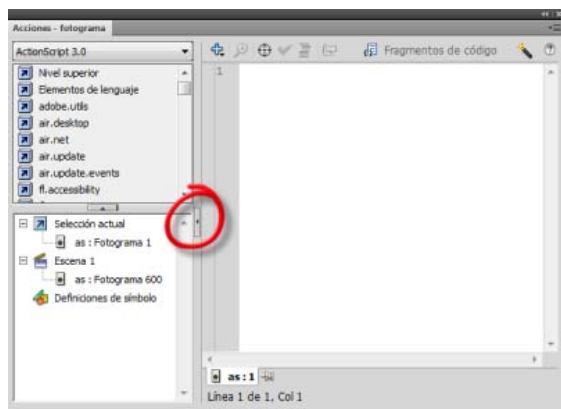
Per convenció els noms d'instància generalment comencen per una lletra minúscula i acaben de la manera següent:

- *_mc* per a clips de pel·lícula (movie clip)
- *_btn* per a botons
- *_txt* per a camps de text

Pas 4 de 19

Escriurem les accions en una **nova capa** que situarem per damunt de totes les altres, i que anomenarem *as* (per **ActionScript**). És convenient tenir totes les accions separades en una capa per a facilitar l'organització del document.

Obrim el panell **Acciones** prement **F9** o **Ventana > Acciones**.



En la columna esquerra d'aquest panell tenim en primer lloc un accés exhaustiu a les diferents classes i funcions disponibles. En el nostre cas no utilitzarem aquesta manera d'introduir codi.

En la part inferior de la mateixa columna tenim informació sobre la selecció actual (en el nostre cas, el fotograma 1 de la capa *as*). Quan tinguem accions distribuïdes en més fotogrames, des d'aquí tindrem un accés còmode i directe que ens permetrà accedir a la programació dels diferents fotogrames.

En la part central tenim l'editor, que és l'àrea sobre la qual escriurem el codi. Sobre aquesta àrea tenim algunes eines per a comprovar el codi, inserir comentaris, accedir a l'ajuda, etc. Recomanem tenir desactivat l'assistent de script.

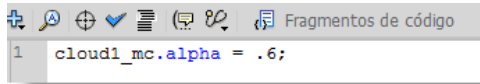
Per a disposar de més espai per a escriure el codi, recomanem reduir la columna esquerra clicant sobre la fletxa petita que separa les dues columnes. Podrem expandir-la de nou quan ho necessitem.

Paso 5 de 19

Comencem assignant un valor de `.6` a la propietat `alpha`. Aquest valor significa una opacitat del 60%. Aquest valor pot oscil·lar entre 0 (transparent) i 1 (opac).

Comencem escrivint el nom de la instància seguit per un punt. Si després d'escriure el punt premem **Ctrl + Barra espaciadora** apareixerà automàticament un desplegable amb diferents mètodes i propietats que podem assignar.

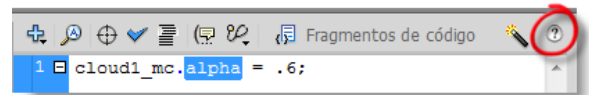
Després del punt s'escriu el mètode que volem executar o la propietat que volem assignar. En aquest cas hem seleccionat la propietat `alpha`. Les paraules clau del llenguatge apareixen en color blau.



Tot seguit, després d'un signe `=`, escrivim el valor per a aquesta propietat. En aquest cas hem escrit `.6` (també podríem haver escrit `0.6`). Els espais abans i després del signe `=` no són necessaris, però milloren la llegibilitat del codi. Acabem cada sentència amb un punt i coma `;`.

Si tenim problemes amb la mida del text del codi, podem obrir les **preferències (Ctrl+U)** i, en la categoria **ActionScript**, canviar el tipus de text, la mida de la lletra o els colors. És preferible mantenir els colors del codi, ja que ens ajuda a distingir paraules clau, comentaris, etc.

Una manera ràpida d'accedir a l'ajuda sobre alguna paraula clau és seleccionar-la directament en l'editor i fer clic sobre la icona de l'ajuda. En la pàgina d'ajuda podrem veure definicions i exemples d'ús de cada paraula clau.



Pas 6 de 19

Si reduïm el panell **Acciones** prement la barra grisa superior del panell per a veure l'escenari, no veiem cap canvi en el núvol. Això es deu al fet que la programació s'executa quan provem o publiquem la pel·lícula (temps d'execució), i no directament en l'escenari d'edició (temps d'edició).

Per a comprovar com ha canviat l'alfa del núvol haurem de provar la pel·lícula amb **Control > Probar pel·lícula (Ctrl+Intro)**.

Si hem assignat un valor per a una propietat en el codi, aquest valor prevaldrà sobre el que hàgim definit en el temps d'edició. Per exemple, en el cas de l'alfa, podem assignar diferents valors en l'inspector de *Propiedades*, però quan provem la pel·lícula sempre veurem el núvol amb el 60% de l'alfa, ja que és el valor que hem introduït en el codi.

De la mateixa manera, si assignem en el codi valors per a altres propietats com, per exemple, les posicions X i Y del clip, aquest apareixerà en les coordenades que especifiquem en la programació, i no en les coordenades en les quals l'hàgim situat en l'escenari.

Per a afegir **interactivitat** a una pel·lícula, el més probable és que necessitem detectar quan es produeix un determinat **esdeveniment** (que s'hagi fet clic sobre un botó, que s'hagi premut una tecla, etc.), i després executar una **acció** sobre la base d'aquest esdeveniment.

Per a això hem d'afegir un *listener*, és a dir, afegir a un objecte la capacitat de detectar si ha passat alguna cosa, i que cridi una funció quan això passi.

La manera general de crear un *listener* és la següent:

```
nameInstance.addEventListener(nameEvent, nameFunction);
```

En el nostre cas farem que el núvol s'arrossegui quan el premem.

Per a això afegirem la línia següent al codi:

```
cloud1_mc.addEventListener(MouseEvent.CLICK, drag);
```

Amb aquest codi hem afegit, a la nostra instància `cloud1_mc`, un detector per a un esdeveniment del ratolí (`MouseEvent`). En concret, l'esdeveniment del ratolí que volem que detecti és si l'usuari l'ha premut (`CLICK`). Si és així, executarà una funció que hem anomenat `drag`, i que definirem més endavant.

Paso 7 de 19

Un esdeveniment de ratolí similar és `MouseEvent.CLICK`. La diferència és que amb `CLICK` es detecta si s'ha premut i s'ha deixat anar el ratolí, és a dir, si s'ha fet clic. Més endavant veurem que en el nostre cas executarem accions diferents per a les accions de prémer i deixar anar el ratolí, i és per això que hem triat l'esdeveniment `MOUSE_DOWN` (i després utilitzarem `MOUSE_UP`).

Ara passarem a definir la nostra funció `drag`. Una funció és un bloc d'instruccions agrupades, i que s'executa quan es crida des d'una altra funció o mètode.

En escriure el codi, les funcions i mètodes es diferencien de les propietats perquè després del nom dels primers hi ha un parèntesi. El parèntesi pot ser buit, o bé pot rebre un o diversos paràmetres separats per comes. Per exemple, `alpha` és una propietat, mentre que `addEventListener` és un mètode amb dos paràmetres (esdeveniment i funció).

Per a crear una funció hem de començar amb la paraula clau `function`. A continuació d'aquesta escrivim el nom que volem donar a la nostra funció. En aquest cas li donem el nom `drag`. Després del nom, escrivim els parèntesis:

```
function drag()
```

Les funcions que s'hagin de cridar des d'un `addEventListener`, com és el nostre cas, reben un **paràmetre**. Aquest paràmetre és un objecte del tipus d'esdeveniment que ha desencadenat la funció, en aquest cas `MouseEvent`.

Entre els parèntesis escriurem el nom de l'objecte (el nom que vulguem), dos punts (`:`) i el tipus d'objecte.

Per tant, de moment, la nostra funció començarà així:

```
function drag(e:MouseEvent)
```

Hem donat al paràmetre el nom `e` per esdeveniment, però n'hi podríem haver donat un altre. Generalment, a aquest paràmetre se sol donar el nom `e` o `event`.

Després del nom de la funció és convenient especificar el tipus de dades que tornarà. En el nostre cas, la funció no tornarà cap resultat, sinó que simplement executarà una acció. En aquests casos, com a tipus de dades s'assigna `void` (buit).

La nostra funció queda com s'indica a continuació:

```
function drag(e:MouseEvent):void
```

Pas 8 de 19

Ara ens queda especificar les accions que volem que executi aquesta funció. Tot el contingut d'una funció s'escriu entre claus, així que per a assegurar-nos que no se'ns oblidin, és convenient escriure primer les claus, i després passar a escriure entre elles les diferents instruccions.

```
function drag(e:MouseEvent):void
{
    //your code here
}
```

Les dues barres que hem afegit en el codi anterior abans de `your code here`, indiquen que és un comentari d'una sola línia. Podem afegir els comentaris que vulguem en el nostre codi, ja que no s'executaran. En comentaris de diverses línies utilitzarem `/*` i `*/` per a delimitar el text que volem que sigui un comentari.

Els comentaris poden ser molt útils no solament per a afegir informació, sinó també per a fer que algunes línies del codi no s'executin temporalment, la qual cosa ens ajudarà a comprovar el funcionament del nostre codi.

Provem d'escriure el codi següent:

```
function drag(e:MouseEvent):void
{
    trace("I'm clicking on the cloud");
}
```

La funció `trace` mostrarà en un panell anomenat **Salida** el que hi hagi dins del parèntesi. Si escrivim entre cometes el contingut del `trace`, en el panell **Salida** es mostrarà exactament la frase que hàgim escrit. Les cometes permeten escriure cadenes de caràcters, i les veurem de color verd en el panell **Acciones**.

Les dades que apareguin en aquest panell no les veurà l'usuari final, però sí quan provem la pel·lícula. Si apareix aquesta frase, vol dir que aquesta línia de codi s'està executant correctament.

Recapitem: hem afegit al núvol un detector de l'esdeveniment de prémer el núvol amb el ratolí, i quan aquest esdeveniment es detecti, s'executarà la funció `drag`, que té la instrucció de mostrar en el panell **Salida** la frase `I'm clicking on the cloud`.

Provem la pel·lícula amb **Control > Probar película** i fem clic sobre el núvol per a comprovar el funcionament del nostre codi.

Paso 9 de 19

Ara que hem comprovat que el nostre codi funciona correctament, substituïm la funció `trace` per l'acció que realment volem que tingui lloc, que és que el núvol s'arrossegui.

```
function drag(e:MouseEvent):void
{
    cloud1_mc.startDrag();
}
```

Si tornem a provar la nostra pel·lícula, podrem veure que en prémer el núvol aquest comença a arrossegar-se juntament amb el ratolí, i no deixa d'arrossegar-se en cap moment.

Perquè l'arrossegament s'aturi, haurem d'introduir una funció nova que permeti que, quan es deixi anar el botó del ratolí, el núvol deixi d'arrossegar-se.

El funcionament serà el mateix que l'anterior. Haurem de crear un altre detector per a l'esdeveniment de deixar anar el ratolí, i assignar-hi una funció que aturi l'arrossegament del núvol.

Per llegibilitat del codi podem ajuntar, d'una banda, les línies amb la creació dels listeners i, d'altra banda, les funcions.

La manera més ràpida per a escriure aquest codi nou és copiar i enganxar el que ja havíem creat, ja que en molts aspectes és similar. Substituïrem `MOUSE_DOWN` per `MOUSE_UP`, i el nom de la funció nova serà `drop` en comptes de `drag`. Finalment, la funció que atura un arrossegament actiu és `stopDrag()`.

El codi que hem creat fins ara en el primer fotograma de la capa *as* és el següent:

```
cloud1_mc.alpha = .6;

cloud1_mc.addEventListener(MouseEvent.CLICK, drag);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);

function drag(e:MouseEvent):void
{
    cloud1_mc.startDrag();
}

function drop(e:MouseEvent):void
{
    cloud1_mc.stopDrag();
}
```

Pas 10 de 19

A més de l'arrossegament interactiu del núvol, afegirem un desplaçament continu del núvol per l'escenari mitjançant una funció que anomenarem `wind`.

Hi ha un tipus d'esdeveniment anomenat `ENTER_FRAME`, que pertany a una categoria genèrica d'esdeveniments anomenada `Event`, que s'executa cada vegada que el cap lector es desplaça un fotograma. Si el cap lector està aturat, però l'objecte a què s'associa és a l'escenari (tant si és visible com no), també s'executarà amb la mateixa freqüència que els fotogrames per segon que tinguem definits.

En el cas de la nostra pel·lícula, una funció que s'associï a un esdeveniment `ENTER_FRAME` s'executarà cada vegada que el cap lector avanci, és a dir, 24 vegades per segon. Canviarem lleugerament la posició del núvol cada 1/24 de segon, per la qual cosa mostrarà una animació fluida.

Afegim aquest listener nou sota els altres dos que havíem creat. Per a això escrivim aquest codi:

```
cloud1_mc.addEventListener(Event.ENTER_FRAME, wind);
```

A falta de definir les instruccions que executarà la funció `wind`, la seva definició quedarà com s'indica a continuació:

```
function wind(e:Event):void
{
}
```

Com podem veure, l'esdeveniment `ENTER_FRAME` pertany a una categoria anomenada `Event`, i no a esdeveniments de ratolí (`MouseEvent`). Aquest `Event` apareixerà tant en la creació del listener com en el paràmetre de la funció.

Perquè el núvol avanci cap a la dreta haurem de variar-ne la posició `x` a poc a poc dins de la funció `wind`. Per exemple, podem especificar que la posició `x` del núvol augmenti d'un en un el seu valor, que traduït a codi seria:

```
cloud1_mc.x = cloud1_mc.x + 1;
```

o bé, més senzill:

```
cloud1_mc.x += 1;
```

que significa sumar-ne el valor més el nombre que hi ha després de `=`, en aquest cas una unitat.

Paso 11 de 19

De moment tindrem la funció `wind` com s'indica a continuació:

```
function wind(e:Event):void
{
    cloud1_mc.x += 1;
}
```

Provem la pel·lícula per comprovar com es desplaça el núvol cap a la dreta, mentre encara es pot arrossegar. Podem provar amb diferents valors fins a trobar una velocitat que ens sembli adequada (per exemple 0.4).

Si mantenim premut el núvol per arrossegar-lo però no deixem anar el ratolí, el núvol continuarà avançant cap a la dreta fora del ratolí. Encara que després deixem anar el ratolí, l'esdeveniment `MOUSE_UP` ja no es produirà, ja que significa aixecar el botó del ratolí damunt del núvol (no fora d'aquest). D'aquesta manera ja no podrem deixar anar l'arrossegament del núvol.

Per a solucionar-ho podem associar a la funció `drop` un altre esdeveniment de ratolí anomenat `ROLL_OUT`, que significa prémer el núvol però que el ratolí s'arrossegui fora d'ell. Podem crear el nou esdeveniment sota l'esdeveniment `MOUSE_UP`.

Per ara els listeners que hem creat seran els següents:

```
cloud1_mc.addEventListener(MouseEvent.CLICK, drag);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);
```

En aquest cas no cal que creem una funció nova, ja que aquest listener cridarà la mateixa funció `drop` que havíem creat abans.

Si provem la pel·lícula veurem que si en algun moment deixem d'arrossegar el núvol, aquest acabarà per desaparèixer per la part dreta de l'escenari. Encara que deixi de visualitzar-se, la instància continuarà executant el codi.

El següent que programarem serà que quan hagi desaparegut per la part dreta, torni a aparèixer per la part esquerra de l'escenari.

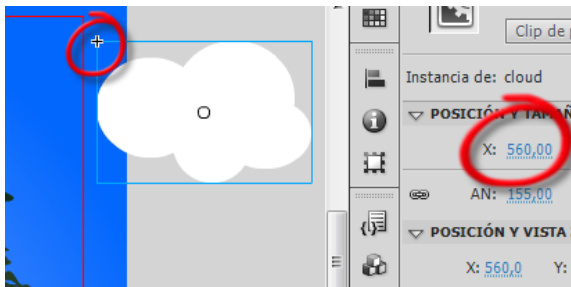
Per fer-ho, introduïrem una sentència condicional dins de la funció `wind`, de tal manera que, si es compleixen uns determinats requisits, el núvol es posicioni en la part esquerra de l'escenari.

Pas 12 de 19

En el nostre cas, definirem que si la posició `x` del núvol assoleix un valor determinat (quan hagi desaparegut per la dreta), que torni a la part esquerra de l'escenari.

Una manera ràpida de fer el càlcul de les posicions d'inici i de final és col·locar el núvol en un extrem i un altre de l'escenari, i anotar el valor d'`x` que apareix en l'inspector de *Propiedades*.

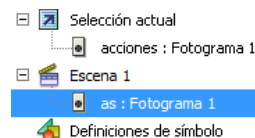
La posició del clip la marca una creu petita. Aquesta posició depèn del punt de registre que hàgim seleccionat en crear el clip.



En aquest cas, seleccionarem com a extrem dret 560 (una mica més que l'amplada de l'escenari), i com a extrem esquerre -160 (una posició en la qual encara no es veu el núvol). Aquests valors dependran de la mida del vostre núvol i del punt de registre.

Podríem fer aquests càlculs utilitzant com a dades l'amplada del clip i l'amplada de l'escenari. El motiu de no fer-ho d'aquesta manera és que en els pròxims passos afegirem un altre núvol al qual programarem profunditat en l'eix `z`, i llavors la posició `x` de l'objecte ja no serà equivalent als píxels de l'escenari, ja que es mourà en un escenari que simularà més profunditat i, per tant, més amplada que els 550 píxels del nostre escenari.

Col·loquem el núvol en ell lloc on vulguem que comenci l'animació i tornem al panell **Acciones (F9)**. Recordem que podem navegar per les diferents accions de la nostra pel·lícula prement la línia corresponent en la part inferior esquerra del panell.



En aquest cas, si no tinguéssim seleccionat el fotograma 1 de la capa accions en la línia de temps, hi podríem anar prement **as: Fotograma 1**.

Paso 13 de 19

La sentència `if`, que és la que utilitzarem, avalua si es compleix una condició, i en cas de complir-se executa les instruccions que indiquem entre les claus. L'estructura és la següent:

```
if (condition)
{
    //statements
}
```

Per tant, en la sentència, haurem d'especificar que si la posició `x` actual del núvol és més gran que 560 (el signe `>` significa més gran que), llavors, la posició hauria de ser `-160`.

La funció `wind` quedarà d'aquesta manera:

```
function wind(e:Event):void
{
    cloud1_mc.x += .4;

    if (cloud1_mc.x > 560)
    {
        cloud1_mc.x = -160;
    }
}
```

Veiem que la funció té les seves pròpies claus, i que dins d'aquesta també hi ha un condicional amb les pròpies claus d'inici i de fi.

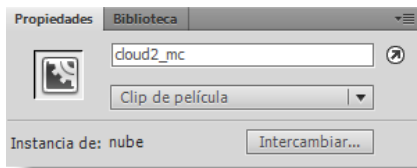
Provem ara la pel·lícula amb **Control > Probar película**. Veiem que, al cap de poc de desaparèixer per l'extrem dret, torna a aparèixer per l'extrem esquerre.

A diferència de la línia de temps principal, que té una extensió de 600 fotogrames, veiem que l'animació del moviment del núvol és independent d'aquesta extensió. En cada repetició del vol dels ocells, el núvol pot ser en un punt diferent.

Per a animar mitjançant programació n'hi ha prou amb un sol fotograma en la línia de temps. Si haguéssim fet aquesta animació en la línia de temps, hauríem necessitat estendre molt més el nombre de fotogrames.

Pas 14 de 19

Arrosseguem una altra instància del clip `cloud` a l'escenari en la mateixa capa `clouds` que el núvol anterior, i en l'inspector de **Propiedades** li donem el nom d'instància `cloud2_mc`.



Al principi de la programació, després de la línia en la qual havíem definit l'alfa de la primera instància, afegim algunes propietats per a la nova instància.

```
cloud1_mc.alpha = .6;
cloud2_mc.alpha = .4; //more transparent
cloud2_mc.scaleY = .7; //lower height (70%)
cloud2_mc.z = 300; //add depth
```

Després afegim els mateixos listeners per a aquesta segona instància. Per a això n'hi ha prou de copiar i enganxar els listeners ja creats i substituir `cloud1_mc` per `cloud2_mc`:

```
cloud1_mc.addEventListener(MouseEvent.CLICK, drag);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);

cloud2_mc.addEventListener(MouseEvent.CLICK, drag);
cloud2_mc.addEventListener(MouseEvent.CLICK, drop);
cloud2_mc.addEventListener(MouseEvent.CLICK, drop);
cloud2_mc.addEventListener(MouseEvent.CLICK, drop);
```

Dins de cada funció també fem referència a la instància `cloud1_mc`. En comptes de repetir les línies amb la programació per a `cloud2_mc` dins de les funcions, el que farem és que la programació faci referència a l'objecte que va iniciar l'esdeveniment.

És a dir, si un listener de la `cloud1_mc` ha cridat la funció `drag`, arrossegarem aquest núvol, però si la funció ha estat iniciada per `cloud2_mc` serà aquest núvol el que arrosseguem.

Paso 15 de 19

Per a això hem d'escriure en primer lloc el nom del paràmetre de l'esdeveniment que hem posat en la funció (e en el nostre cas). Després afegim la propietat `target`. Amb això es farà referència a l'objecte que va disparar l'acció en rebre l'esdeveniment.

Per tant, si substituïm dins de les funcions el nom de la instància per `e.target`, aconseguirem tenir una referència directa als objectes que han enviat la funció.

Per a comprovar-ho en la funció `drag`, a més de substituir `cloud1_mc` per `e.target`, afegirem un `trace` que ens mostri el nom (`name`) de la instància que desencadena l'esdeveniment.

```
function drag(e:MouseEvent):void
{
    e.target.startDrag();
    trace(e.target.name);
}
```

Com que el contingut del `trace` no està entre cometes, la funció no escriu el valor literal que hem escrit entre parèntesis, sinó el seu valor (en aquest cas és un nom). D'aquesta manera, veurem que en prémer cada núvol el panell *Salida* ens mostra el seu nom.

Afegim `e.target` a totes les funcions substituint a `cloud1_mc`.

Ara tenim un problema amb les posicions que havíem determinat en el condicional `if`, ja que en tenir el segon núvol més profunditat hauríem de distanciar els dos extrems.

Una manera de conèixer aproximadament aquests valors és canviar el `trace` de la funció `drag` perquè ens mostri la posició x del clip sobre el qual fem clic:

```
trace(e.target.x);
```

Després desactivem temporalment l'`if` de la funció `wind` convertint-lo en un comentari afegint `/*` abans de l'`if` i `*/` després de la clau de tancament de l'`if`.

```
/*if (e.target.x > 560)
{
    e.target.x = -160;
}*/
```

Pas 16 de 19

Ara provem la pel·lícula i tractem de prémer el segon núvol en posicions molt properes al límit de l'escenari per tots dos costats per fer-nos una idea dels valors d'x adequats.

El panell **Salida** mostrarà el valor x per a aquest núvol, que com podem veure ha variat considerablement en estar programat per a mostrar-se en un pla més profund.

De fet, aquesta diferència entre la propietat x depenent de la profunditat també la podem veure en l'animació, ja que encara que tots dos núvols tenen assignat el mateix increment en el valor x, el més llunyà sembla avançar més a poc a poc que el més proper.

En el nostre cas, com a valor mínim seleccionarem -315 i com a valor màxim 715, però aquests valors dependran de la mida del nostre núvol.

Ara el primer núvol trigarà més a aparèixer. Podríem afegir un condicional i, depenent de si el target és el primer núvol o el segon, prendre uns límits o uns altres. En aquest cas no té importància que un núvol tardi més en aparèixer de nou, així que deixem com a límits els del núvol més llunyà.

Com hem pogut comprovar, l'ús de `trace` i dels comentaris poden ser molt útils a l'hora de programar.

Després d'esborrar el `trace`, ja que ja no el necessitem, les funcions quedaran així:

```
function drag(e:MouseEvent):void
{
    e.target.startDrag();
}

function drop(e:MouseEvent):void
{
    e.target.stopDrag();
}

function wind(e:Event):void
{
    e.target.x += .4;
    if (e.target.x > 715)
    {
        e.target.x = -315;
    }
}
```


Paso 17 de 19

Com a últim detall, farem que es mostri un cursor amb forma de mà quan estiguem sobre el núvol, que sembli un botó i que permeti intuir que el núvol es pot arrossegar.

Per fer-ho, abans dels listeners afegim les propietats següents:

```
cloud1_mc.buttonMode = true;
cloud2_mc.buttonMode = true;
```

Amb això ja veurem que el cursor adopta la forma d'una mà quan estem a sobre d'un núvol, ja que tracta visualment els núvols com si fossin un botó.

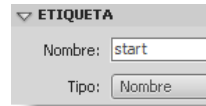
Analitzem què passa amb la nostra pel·lícula. Comença en el fotograma 1 i el player de flash llegeix tota la programació que hem escrit i la va executant. Després la línia de temps continua durant 600 fotogrames, i torna al fotograma 1, on torna a llegir tota la programació.

Per a no sobrecarregar al player amb informació repetitiva, podem fer que el bucle de la línia de temps principal vagi del fotograma 600 al fotograma 2, sense tornar a passar pel fotograma 1, que és on hem escrit tota la nostra programació.

Per a això creem un **fotograma clau** en el fotograma 600 de la capa as, i escrivim la programació següent:

```
gotoAndPlay(2);
```

Això significa que quan el cap lector arribi al fotograma 600, on hi ha escrita la programació, anirà al fotograma 2 i continuarà amb la reproducció.



Si haguéssim creat un fotograma clau i, en l'inspector de *Propiedades*, hi haguéssim assignat el nom *start*, també podríem haver anat a aquest fotograma escrivint:

```
gotoAndPlay("start");
```

Aquest pas no és necessari en aquest cas, però pot resultar útil conèixer aquesta opció per a crear fàcilment menús de navegació.

Pas 18 de 19

Aquest és l'aspecte complet que tindrà la programació completa d'aquest tutorial:

```
cloud1_mc.alpha = .6;
cloud2_mc.alpha = .4;
cloud2_mc.scaleY = .7;
cloud2_mc.z = 300;
cloud1_mc.buttonMode = true;
cloud2_mc.buttonMode = true;

cloud1_mc.addEventListener(MouseEvent.CLICK, drag);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);
cloud1_mc.addEventListener(MouseEvent.CLICK, drop);
cloud1_mc.addEventListener(Event.ENTER_FRAME, wind);

cloud2_mc.addEventListener(MouseEvent.CLICK, drag);
cloud2_mc.addEventListener(MouseEvent.CLICK, drop);
cloud2_mc.addEventListener(MouseEvent.CLICK, drop);
cloud2_mc.addEventListener(Event.ENTER_FRAME, wind);
```

```
function drag(e:MouseEvent):void
{
    e.target.startDrag();
}

function drop(e:MouseEvent):void
{
    e.target.stopDrag();
}

function wind(e:Event):void
{
    e.target.x += .4;
    if (e.target.x > 715)
    {
        e.target.x = -315;
    }
}
```

Pas 19 de 19

Per a complementar els conceptes desenvolupats en aquest tutorial, es recomana fer les activitats següents:

1. Canvia la programació perquè els núvols es desplacin de dreta a esquerra.
2. Afegeix un núvol nou a l'escenari que també es desplaci, però que no es pugui arrossegar.
3. Fes que els núvols es tornin més transparents en desplaçar-se, però que en aparèixer de nou per l'altre costat de l'escenari recuperin el valor alfa original.

