

Tutorial 9 - Creació d'un joc (I)

Pas 1 de 26

En aquest tutorial crearem la primera part d'un joc en el qual hem de portar una nau fins a un planeta, amb l'ajuda de les fletxes del teclat.

Començarem creant els elements gràfics necessaris per al joc, utilitzant eines com l'oval i el rectangle simple, i el pinzell ruixador.

Referent a la programació, aprendrem a controlar un objecte amb el teclat, i també a detectar-ne la posició o saber si xoca amb algun altre objecte.

També aprendrem a crear botons que ens mostrin informació quan hi situem el punter a sobre.

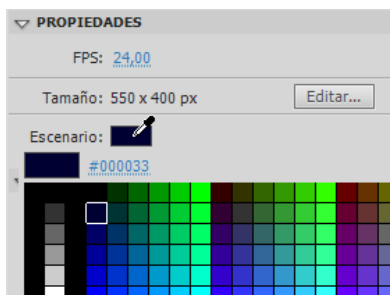
En el tutorial següent afegirem complexitat al nostre joc afegint obstacles amb moviments aleatoris.



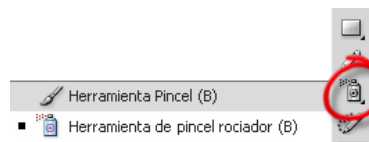
Pas 2 de 26

Creem un nou **ActionScript 3.0** des de l'àrea **Crear nuevo** de la pantalla benvinguda o bé seleccionant **Archivo > Nuevo**. Desem l'arxiu com a *tutorial9 fla*.

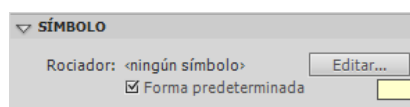
Canviem el **color del fons** en l'inspector de *Propiedades*. Hi assignem un color blau fosc (#000033).



En primer lloc, crearem els elements gràfics del nostre joc, començant pel fons d'estrelles. Per a crear aquest fons utilitzarem l'eina **Pinzel rociador** que s'agrupa amb l'eina *Pinzel*.



Aquesta eina emet de manera predeterminada un esprai de punts de partícules amb el color que seleccionem. Amb l'eina *Pinzel rociador* seleccionada, en l'inspector de *Propiedades* canviem el **color** de la partícula a un groc clar (#FFFFCC).



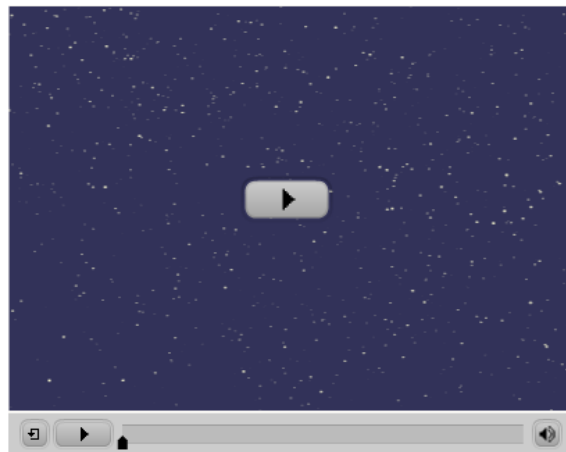
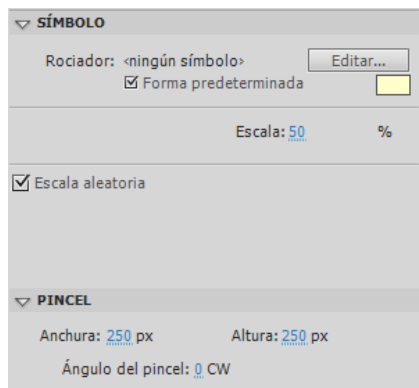
Amb el botó *Editar* podríem seleccionar com a partícula emesa un símbol que tinguéssim en la biblioteca. En aquest cas mantindrem la forma predeterminada.

Paso 3 de 26

Continuant l'edició de les propietats del pinzell ruixador, canviem l'**Escala** a 50%. L'escala es refereix a la mida de les partícules emeses.

Seleccionem la casella **Escala aleatoria**. Això fa que les partícules siguin de diferents mides, encara que la mida màxima serà la que hem indicat prèviament en la casella **Escala**.

Per acabar, en l'àrea **Pincel**, seleccionem una **amplada** i una **altura** de 250 px. Això farà que les partícules estiguin més distanciades entre elles.



Premem i l'arrosseguem sobre l'escenari amb el **pinzell ruixador** fins a aconseguir un fons estrellat que ens agradi.

Anomenem aquesta capa **stars**. Després bloquegem la capa per evitar canvis accidentals.

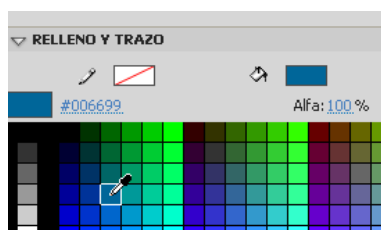
Pas 4 de 26

El següent pas serà la creació del planeta. El nostre planeta estarà format per la part interior i els anells, és a dir, contindrà més d'una capa. Perquè totes les capes formin part del mateix objecte, el més còmode és crear-les directament dins d'un mateix símbol.

Seleccionem **Insertar > Nuevo símbolo**, li donem el nom *planet* i com a **Tipo** seleccionem **Clip de película**.

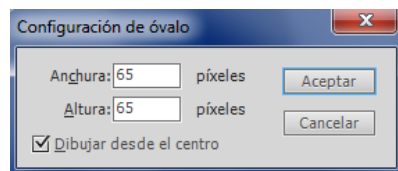
Dins del símbol *planet*, dibuixem en primer lloc la part interior del planeta. Per a això seleccionem l'eina **Óvalo**.

En l'inspector de **Propiedades** **eliminem el color del traç** i seleccionem com a **color de farciment** el color #006699.



Amb l'eina **Óvalo** seleccionada, premem la tecla **Alt** i, sense deixar-la anar, fem clic sobre l'escenari.

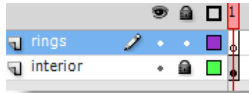
S'obrirà la pantalla **Configuración de óvalo**, que permet definir directament la mida de l'oval. En aquest cas li assignarem una **amplada** i una **altura** de 65 px.



Seleccionem l'oval que hem creat i el centrem dins del seu propi escenari amb ajuda del panell **Alinear**.

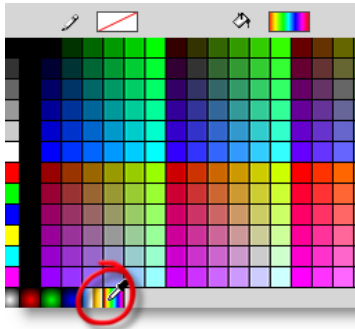
Paso 5 de 26

Anomenem *interior* a la capa en la qual hem creat l'oval anterior, i afegim una nova capa a la qual anomenarem *rings*.



Aquesta vegada seleccionem l'eina **Óvalo simple** per poder modificar algunes característiques de l'oval després de dibuixar-lo.

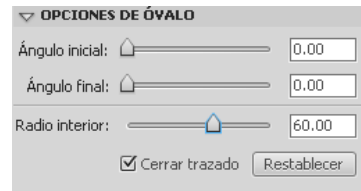
Aquesta vegada tampoc no seleccionarem un traç en l'inspector de *Propiedades*. Com a **color de farciment** seleccionarem el **degradat multicolor** que apareix en la part inferior de la paleta.



Com en el pas anterior, premem la tecla **Alt** i, sense deixar-la anar, fem clic sobre qualsevol punt de l'escenari. De moment no és necessari que l'oval dels anells estigui centrat.

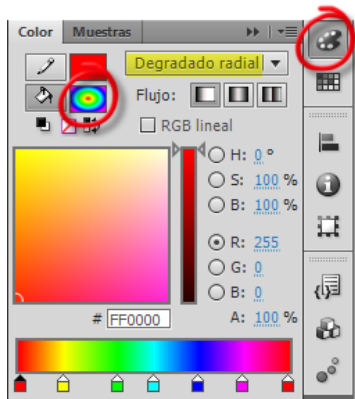
En la pantalla **Configuración de óvalo** assignem una **anchura** i **altura** de **150 px**. Acceptem per crear l'oval.

A **Opciones de óvalo** establim un **Radio interior** de **60**. Recomanem provar les opcions d'angle inicial i final per a conèixer les possibilitats que ofereix, encara que finalment deixarem els angles a 0.

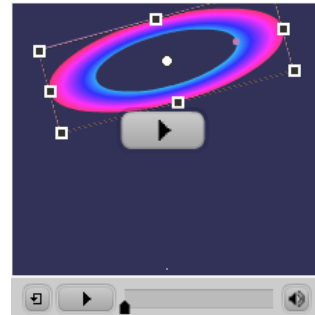


Pas 6 de 26

El farciment multicolor que hem utilitzat és per defecte un degradat lineal. Per convertir el degradat en radial, obrim el panell **Color** i, amb l'oval dels anells seleccionat, canviem el **Tipo de color** a **Degradado radial**.

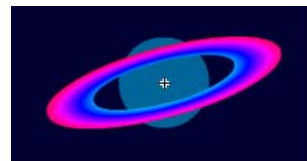
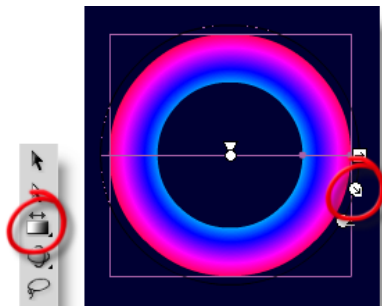


Amb l'eina **Transformación libre** modifiquem la forma dels anells, tal com es mostra en el vídeo.



Centrem els anells en l'escenari amb ajuda del panell **Alinear**. La part interior del planeta haurà de sortir lleugerament tant per sobre com per sota dels anells. Si no és així, farem les rectificacions pertinents en els anells amb l'eina **Transformación libre**.

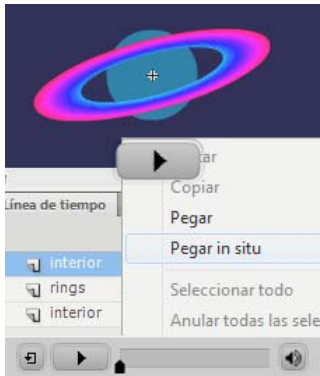
Tenim diverses formes de canviar els colors dels anells. Una d'elles és utilitzar l'eina **Transformación de degradado** i canviar l'amplitud del degradat perquè es mostrin diferents colors.



Paso 7 de 26

La part interior del planeta queda completament per darrere dels anells. Tanmateix, perquè sembli que aquesta part està dins dels anells, una part del planeta s'hauria de veure per davant d'ells, i una altra part per darrere.

Per a aconseguir aquest efecte, duplicarem la part interior del planeta en una nova capa superior, que podem anomenar *interior*.



Copiem l'oval de la part interior del planeta fent-hi clic a sobre amb el **botó dret** del ratolí i seleccionant **Copiar**.

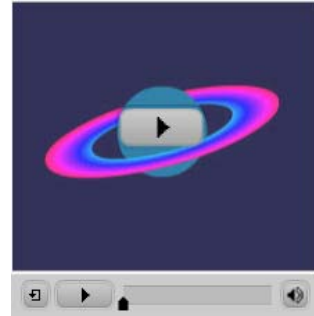
Bloquegem les dues capes inferiors i, amb la capa superior activa, fem clic amb el **botó dret** sobre l'escenari i seleccionem **Pegar in situ**.

Aquesta funció copiarà el dibuix de la part interior del planeta en la mateixa posició que tenia en la capa inferior.

Si ara esborrem de la capa superior una part del planeta, podrem veure el que hi ha sota aquesta part, és a dir, els anells en primer lloc i, al darrere, la part interior del planeta en la capa inferior.

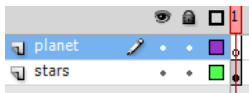
Provem d'eliminar la part superior o inferior de la part interior del planeta en la capa superior, segons la perspectiva que vulguem donar al nostre planeta (vist des de dalt o vist des de baix).

Observem el vídeo per entendre millor el que passa quan s'esborra una zona del planeta en la capa superior. Triem la perspectiva que més ens agradi.



Pas 8 de 26

Tornant a l'escena principal, creem una nova capa anomenada *planet* per sobre de la capa *stars*.



Amb la capa *planet* activa, arrosseguem des de la biblioteca una instància del clip *planet*.

Situem el clip en la part superior dreta de l'escenari, i li donem el nom d'instància *planet_mc*.



Creem una altra capa anomenada *spacecraft* en la qual situarem la petita nau que controlarà l'usuari.

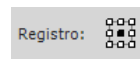
Aquesta nau tindrà una sola capa, així que podem crear-la a la capa *spacecraft* de l'escena principal.

Dibuixem una nau d'aspecte similar al d'aquesta imatge. Per a fer-ho podem utilitzar eines com el rectangle i la línia.



La imatge està molt ampliada. L'altura total de la nau ha de ser aproximadament de 25 px, i l'amplada d'uns 10 px.

Una vegada dibuixada, seleccionem **Modificar > Convertir en símbol** o premem **F8**. Li donem al clip el nom *spacecraft*, i com a **Registro** li assignem el **punt central**. Això farà que el clip quedi centrat en el seu propi escenari.



Situem la nau en la part inferior esquerra de l'escenari. En l'inspector de *Propiedades* li assignem el nom *spacecraft_mc*.

Amb aquests dos clips ja podem començar la programació del nostre joc. Creem una nova capa a la qual anomenarem *as*, i amb el primer fotograma d'aquesta capa seleccionat, premem **Ventana > Acciones** o **F9**.

Paso 9 de 26

Volem que la nau giri i es desplaci quan premem les fletxes del teclat, així que el primer pas serà convertir la nau en un detector d'esdeveniments del teclat.

Programem que, quan es detecti una pulsació de tecla, s'executi una funció a la qual anomenarem `arrows`. La manera d'afegir un esdeveniment de teclat és similar al que hem vist fins ara.

```
spacecraft_mc.addEventListener(KeyboardEvent.KEY_DOWN, arrows);

function arrows(e:KeyboardEvent):void
{
    trace("He pressed a key");
}
```

Amb aquesta programació diem a la nau que, quan detecti que s'ha premut una tecla, executi una funció anomenada `arrows`. Aquesta funció, que rep un esdeveniment de teclat, té al seu torn l'ordre d'escriure en el panell *Salida* la frase `I pressed a key`.

No obstant això, si provem la pel·lícula i premem les fletxes del teclat no passarà res.

Això es deu al fet que, perquè funcioni un detector d'esdeveniments de teclat, l'objecte encarregat de detectar-los ha de rebre el focus de la pel·lícula, és a dir, ha d'estar seleccionat. Per a seleccionar-lo, a l'inici de la programació afegim el codi següent:

```
stage.focus = spacecraft_mc;
```

Ara que la nau té el focus, detectarà les pulsacions del nostre teclat, i després de cada pulsació escriurà la frase que havíem posat en el `trace`. Quan estem en fase de proves, és possible que aquesta detecció no funcioni amb les tecles que tinguem en Flash com a mètodes abreujats de teclat. És per això que en la fase de proves també ens limitarem a les fletxes del teclat.

Com que ara la nau té el focus, podem veure que apareix un rectangle al voltant de la nau per a indicar que té el focus. Perquè no aparegui aquest rectangle, abans de donar el focus a la nau afegirem una sentència que especifiqui que, encara que la nostra nau tingui el focus, no mostri un rectangle. Les dues primeres línies de la nostra programació quedaran així:

```
spacecraft_mc.focusRect = false;
stage.focus = spacecraft_mc;
```

Pas 10 de 26

Ara que hem provat que el `trace` funciona, i que la nau no mostra el rectangle del focus, procedirem a substituir el `trace` per les sentències que volem executar.

Volem avaluar quina fletxa s'ha premut, i executar accions diferents segons si s'ha premut una fletxa o una altra. Per a aquest tipus de problemes és molt útil una sentència anomenada `switch`.

Per comprendre aquesta sentència, suposem el cas d'un test amb respostes a, b i c:

```
switch (answer)
{
    case a :
        trace("the user has chosen a");
        break;

    case b :
        trace("the user has chosen b");
        break;

    case c :
        trace("the user has chosen c");
        break;

    default :
        trace("ha ocurrido un error");
}
```

El que la sentència `switch` faria en aquest cas és avaluar `answer` (variable que hi ha entre parèntesis).

En cas que la resposta fos per exemple `b`, s'anitzaran els casos fins a trobar-ne un que sigui correcte (`case b`), i llavors s'executaran les sentències que hi hagi dins d'aquest `case` (en aquest cas un `trace` que informa que s'ha respost `b`), i ja no es revisaran els altres casos (`break`).

Cada `case` ha d'acabar sempre amb una sentència `break` perquè s'ometi la resta de la sentència `switch` quan trobem un `case` correcte. Així evitem que comprovi la resta dels casos.

Si no hi ha cap `case` correcte, llavors s'executarà el que s'indiqui a `default`. Aquí les úniques respostes possibles són `a`, `b` o `c`, per la qual cosa si `answer` és qualsevol altra cosa, hem programat un `trace` que indiqui que hi ha hagut un error. Aquí no faria falta un `break`.

Sempre és convenient incloure `default`, encara que no prevegem que hagi de passar res fora dels `case`.

És important no oblidar que aquesta sentència també té les seves pròpies claus a l'inici i al final.

Paso 11 de 26

Avaluem quina tecla ha detectat el *listener* que s'ha premut (`e.keyCode`). Segons la tecla premuda, farem girar el clip cap a un costat o cap a un altre. Si s'ha premut qualsevol altra tecla, no farem res, llevat sortir de la sentència.

La nostra sentència, que és dins de la funció `arrows` (substituint `trace`), quedarà com s'indica a continuació:

```
switch (e.keyCode)
{
    case Keyboard.RIGHT :
        e.target.rotation = 90;
        break;

    case Keyboard.LEFT :
        e.target.rotation = -90;
        break;

    case Keyboard.UP :
        e.target.rotation = 0;
        break;

    case Keyboard.DOWN :
        e.target.rotation = 180;
        break;

    default :
        break;
}
```

La rotació es mesura en graus (360 graus seria un gir complet). El clip girarà sobre el seu punt de registre, que en aquest cas és en el centre del clip.

Provem la pel·lícula (**Ctrl+Intro**). La nau ja s'orienta cap a un costat o cap a un altre depenent de la fletxa que hàgim premut.

El pas següent serà afegir sentències perquè la nau, a més de girar, es desplaci. Per a aconseguir-ho podem utilitzar sentències similars a les que hem utilitzat per a desplaçar el núvol en el tutorial 6.

Per exemple, podem afegir les sentències següents en cada `case`, segons correspongui:

```
e.target.x += 3; //3px right
e.target.x -= 3; //3px left
e.target.y -= 3; //3px up
e.target.y += 3; //3px down
```

El primer `case` quedaria, per tant, de la manera següent:

```
case Keyboard.RIGHT :
    e.target.rotation = 90;
    e.target.x += 3;
    break;
```

Pas 12 de 26

Completem els quatre `case` mb les sentències corresponents i tornem a provar la pel·lícula. És convenient provar la pel·lícula amb cada petit pas que fem, per a poder comprovar què fa exactament cada part del codi que anem afegint.

La nau ja es mou cap als quatre costats, depenent de la fletxa que premem. Tanmateix, el moviment no és gaire fluid.

Si volem que, en prémer una fletxa, el moviment cap a aquest costat es mantingui de manera contínua fins que en premem una altra, llavors necessitarem afegir un `ENTER_FRAME`.

Afegim a la nau un detector de l'esdeveniment `ENTER_FRAME`, que cridi una funció que anomenarem `moveSpacecraft`:

```
spacecraft_mc.addEventListener(Event.ENTER_FRAME, moveSpacecraft);
```

La funció `moveSpacecraft` ha de detectar l'últim moviment que s'ha activat, i continuar en la mateixa direcció. Per exemple, en cas que el moviment sigui cap a la dreta, llavors ha de continuar avançant cap a la dreta.

Crearem una variable anomenada `course`, que emmagatzemi un text amb la direcció del moviment.

Les variables es creen amb l'estructura següent (assignar un valor inicial és opcional):

```
var identifier:DataType = value;
```

Per a la llegibilitat del codi, és convenient declarar les variables que utilitzarem en la part superior del codi. Per tant, afegirem aquesta primera línia al nostre codi:

```
var course:String = "";
```

El tipus de variable `String` representa cadenes de caràcters. Les cadenes de caràcters sempre van entre cometes.

Com a valor inicial hem creat una cadena buida (sense cap contingut).

Paso 13 de 26

Com que ara volem que els desplaçaments de la nau els faci una altra funció que rep un `ENTER_FRAME` (la funció `moveSpacecraft`), en la funció `arrows` només desarem la dada sobre la rotació i cap a on s'ha de dirigir el moviment.

Per tant, substituïm les sentències `case` del desplaçament en la funció `arrows` de la manera següent:

```
case Keyboard.RIGHT :
    e.target.rotation = 90;
    course = "right";
    break;
```

En els altres casos, assignem a la variable `course` els valors "left", "up" i "down" respectivament.

La funció `moveSpacecraft`, que crearem a continuació, mourà de manera contínua el clip cap a un costat o un altre depenent de l'últim valor assignat a la variable `course`, ja que s'executarà contínuament quan la cridi un esdeveniment `ENTER_FRAME`.

La funció `moveSpacecraft` avaluarà el contingut de la variable `course`, i a partir d'això mourà la nau.

```
function moveSpacecraft(e:Event):void
{
    switch (course)
    {
        case "right" :
            e.target.x += 3;
            break;

        case "left" :
            e.target.x -= 3;
            break;

        case "up" :
            e.target.y -= 3;
            break;

        case "down" :
            e.target.y += 3;
            break;

        default :
            break;
    }
}
```

Pas 14 de 26

El valor 3 és la quantitat de píxels que es desplaça la nau en cada `ENTER_FRAME`. Per tant, és la velocitat de la nau. En comptes d'escriure aquest valor, podríem crear una variable anomenada `speed`, que especifiqui el nombre de píxels en els quals varia la posició de la nau en cada moment.

Podem crear aquesta variable en la part superior de la programació, després de la declaració de la variable `course`:

```
var course:String = "";
var speed:Number = 3;
```

Dins de la funció `moveSpacecraft`, substituïm el nombre de píxels per la variable `speed` en els quatre `case`, que prendrà el valor que li hem assignat quan creem la variable:

```
case "right" :
    e.target.x += speed;
    break;
```

D'aquesta manera, si canviem el valor de la variable `speed`, canviarà la velocitat dels desplaçaments en les quatre direccions. Podem provar diferents valors fins que ens sembli una velocitat adequada.

El pas següent serà detectar si la nau arriba al planeta, que serà la meta del joc. Per a fer-ho, mentre la nau es desplaça hem d'anar comprovant si xoca amb el planeta.

Dins de la funció `moveSpacecraft`, per sota de la clau de tancament de la sentència `switch`, afegirem aquest codi:

```
if (e.target.hitTestObject(planet_mc))
{
    trace("Targed achieved. I win.");
}
```

És un condicional que comprova si la nau (`e.target`) xoca (`hitTestObject`) amb el planeta (`planet_mc`). Si xoca, llavors es mostrarà la frase `Targed achieved. I win.` en el panell *Salida*.

Afegirem un altre condicional, aquesta vegada per detectar si la nau surt fora dels límits de l'escenari, que mesura 550 x 400. Per tant, haurém de comprovar si la propietat `x` (posició horitzontal) de la nau té un valor inferior a 0 o superior a 550, i si la propietat `y` (posició vertical) té un valor per sota de 0 o per sobre de 400. En qualsevol dels quatre casos, la nau estarà posicionada fora de l'escenari.

Paso 15 de 26

Aquest nou condicional, escrit també dins de la funció `moveSpacecraft`, quedarà de la manera següent (les barres verticals `||`, que s'escriuen amb `AltGr + 1`, equivalen a l'OR lògic):

```
if (e.target.x < 0 || e.target.x > 550 || e.target.y < 0 ||
    e.target.y > 400)
{
    trace("I'm outside the stage. I lost.");
}
```

Provem la pel·lícula per comprovar si funciona correctament. La nau es desplaçarà per l'escenari depenent de les fletxes que premem en el teclat. Si arribem al planeta, apareixerà la frase que hem guanyat en el panell *Salida*, i si som fora dels límits de l'escenari, apareixerà la frase que hem perdut.

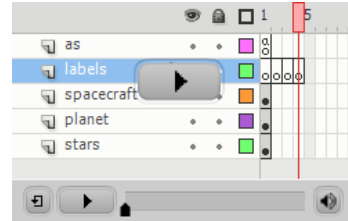
Ara el joc comença directament, però seria adequat disposar d'una pantalla prèvia amb les instruccions del joc, i que en arribar al planeta (guanyar) o sortir de l'escenari (perdre) es mostrés una pantalla diferent i el joc acabés, donant opció a tornar a jugar.

Per tant, a continuació crearem quatre fotogrames diferents:

- Un fotograma inicial amb dos botons, un amb les instruccions del joc i un altre per a començar a jugar.
- Un segon fotograma en el qual es desenvoluparà el joc.
- Un tercer fotograma per a indicar que hem guanyat.
- Un últim fotograma per a indicar que hem perdut.

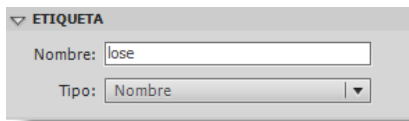
Aquests dos últims fotogrames tindran un botó per a tornar jugar.

Inserim una nova capa amb el nom *labels*. Inserim **fotogrames clau** en els fotogrames 2, 3 i 4. Podem crear aquests fotogrames clau fàcilment si fem clic sobre cada fotograma i premem la tecla **F6**.



Pas 16 de 26

Premem sobre cada fotograma de la capa *labels*, i assignem a cada un un **nom d'etiqueta** en l'inspector de **Propiedades**. Al fotograma número 1 l'anomenarem *start*, al 2 *game*, al 3 *win*, i al 4 *lose*.

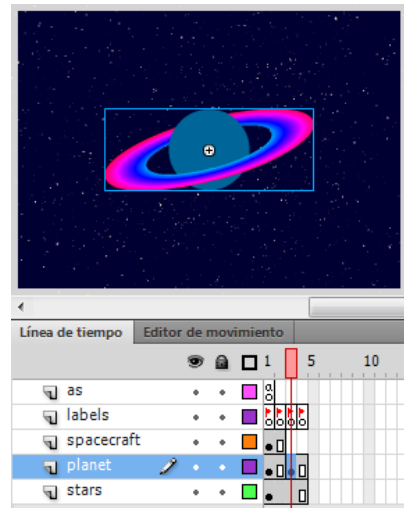
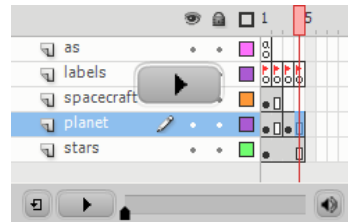


Els fotogrames que tinguin una etiqueta mostraran una bandera vermella petita en la línia de temps.

La nau i el planeta hauran de mantenir-se visibles i sense canvis en els fotogrames *start* i *game*, mentre que les estrelles es veuran com a fons en tots 4 fotogrames.

En els fotogrames *win* i *lose* utilitzarem de nou una imatge del planeta, però aquesta vegada el situarem a una mida més gran i en el centre de la pantalla.

Per tant, inserim els fotogrames corresponents prement **F5**, i després inserirem un fotograma clau amb **F6** en el tercer fotograma de la capa *planet*, on farem un canvi en la mida i posició del planeta que es mantindrà en els fotogrames 3 i 4.



Paso 17 de 26

Canviem la posició i mida del planeta en el fotograma 3, i el posicionem en el centre de l'escenari i amb una mida més gran.

Creem una **nova capa** anomenada *text*.

En el fotograma 3 d'aquesta nova capa creem un **fotograma clau** (F6).

Amb l'eina **Texto** creem en aquest fotograma un camp de text clàssic i estàtic amb el text *you won!!!*.

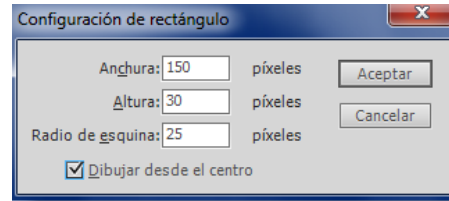
En el fotograma 4 de la capa *text* creem un nou fotograma clau (F6), i canviem el text a *you lost!!!*



Creem una **nova capa** anomenada *buttons*.

Per crear un rectangle que després serà el fons del botó, seleccionem l'eina **Rectángulo**, **sense traç** i amb color de **farciment** vermell.

Amb la tecla **Alt** premuda fem clic sobre l'escenari. En la configuració del rectangle, establim una **amplada** de 150 px, una **altura** de 30 px, i un **radi de cantonada** de 25 px, perquè apareguin les cantonades arrodonides.

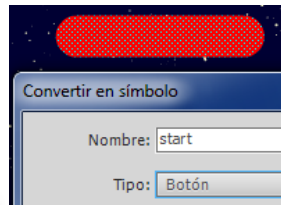
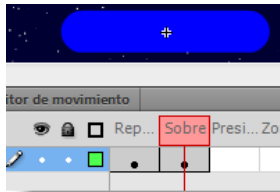


Pas 18 de 26

Seleccionem el rectangle vermell que hem creat, i premem **F8** per convertir-lo en **símbol**.

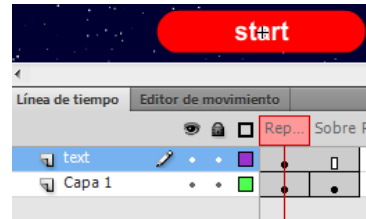
Li donem el nom *start*, i seleccionem que sigui del tipus **Botón**.

Fem doble clic sobre el botó en



Afegim una nova capa sobre el fons del botó, en la qual situarem el text del botó. Amb l'eina **Texto** creem un camp de text clàssic estàtic, i escrivim *start* en color blanc. Centrem el camp de text en el rectangle de fons.

Aquest text es mantindrà tant en el fotograma *Reposo* como en el fotograma *Sobre*.



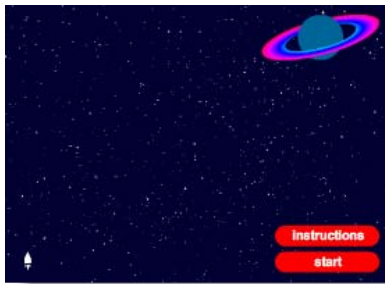
l'escenari per editar-lo.

Inserim un **fotograma clau** prement **F6** en l'estat *Sobre*, i canviem el **color** del rectangle a blau en l'inspector de *Propiedades*.

Tornem a l'escena principal, i situem el botó en la part inferior dreta de l'escenari. En l'inspector de *Propiedades* li assignem el **nom d'instància** *start_btn*.

Paso 19 de 26

Creem un botó nou anomenat *instructions* amb les mateixes característiques que l'anterior, però aquesta vegada amb el text *instructions*. El situem en l'escenari sobre el botó *start*.



Farem que quan situem el punter sobre el botó *instructions*, apareguin directament les instruccions del joc. La manera més senzilla d'aconseguir aquest efecte és afegint el contingut que volem mostrar al fotograma *Sobre* del botó.

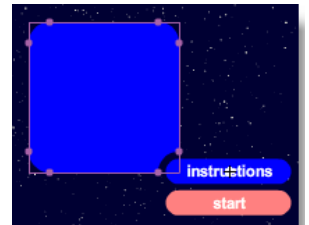
Fem doble clic sobre el botó *instructions* per editar-lo i afegir alguns canvis al seu fotograma *Sobre*.

Afegim una **nova capa** en la línia de temps del botó, i inserim un **fotograma clau** en l'estat *Sobre*.

Seleccionem l'eina **Rectángulo simple**, per poder efectuar modificacions en les característiques del rectangle després de dibuixar-lo.



En el fotograma *Sobre* d'aquesta nova capa que hem creat, dibuixem un rectangle amb el mateix color blau que havíem utilitzat per al botó.

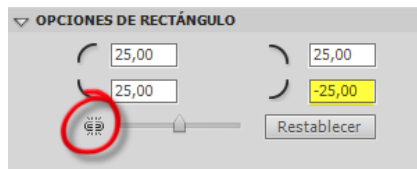


Pas 20 de 26

Amb el rectangle acabat de crear seleccionat en l'escenari, canviem el valor de les cantonades a el l'inspector de *Propiedades*.

Per a poder assignar un valor diferent a una cantonada, en primer lloc hem de desactivar el bloqueig del radi de les cantonades.

Després assignem un valor de 25 a totes les cantonades excepte a la cantonada inferior dreta, a la qual assignarem un valor de -25. D'aquesta manera aconseguirem una cantonada amb el radi invers.



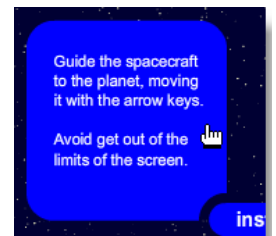
Creem una altra **capa** per sobre d'aquest rectangle per a escriure-hi el text de les instruccions. Inserim un **fotograma clau** en l'estat *Sobre* d'aquesta capa.

El **text**, en color blanc per ressaltar sobre el requadre blau, podria ser similar al següent: *Guide the spacecraft to the planet, moving it with the arrow keys. Avoid get out of the limits of the screen.*

Tornant a l'**escena principal**, seleccionem **Control > Habilitar botones simples** per poder comprovar l'efecte de situar el punter sobre un botó sense necessitat de provar la pel·lícula.

Podem comprovar que si situem el punter sobre l'àrea en què es mostren les instruccions, el botó també s'activa.

Tanmateix, volem que només es mostrin les instruccions si ens situem sobre el rectangle vermell del botó *instructions* en estat de repòs.

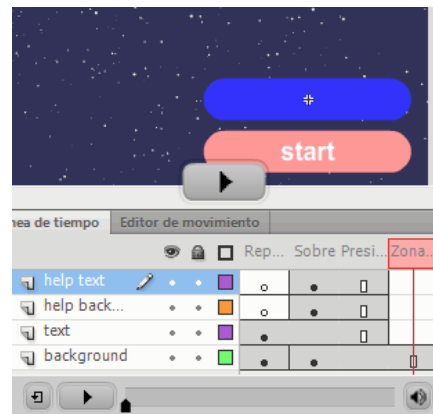
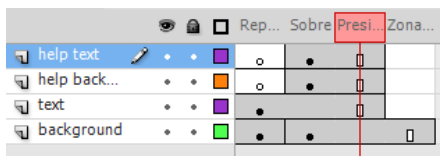


Paso 21 de 26

Seleccionem de nou **Control > Habilitar botones simples** per desactivar els botons, i així poder fer **doble clic** sobre el botó *instructions* per a editar-lo.

Inserim fotogrames (F5) de tal manera que en la *zona activa* es mostri el fons del botó, és a dir, el rectangle arrodonit que servia de base al botó. Aquesta serà l'àrea de clic del botó.

Inserim fotogrames en la resta de les capes perquè també es mostrin en l'estat *Presionado*, i així evitar que en aquest estat només es mostri el fotograma del fons del botó.



Si ara tornem a habilitar els botons simples per fer la prova, veurem que el botó es comporta de la manera esperada.

Aquesta manera d'utilitzar l'estat *Sobre* de un botón, també anomenat *rollover*, pot resultar molt útil per a crear senzilles aplicacions educatives, ja que podem mostrar diferents tipus d'informació en situar-nos sobre una àrea concreta.

Pas 22 de 26

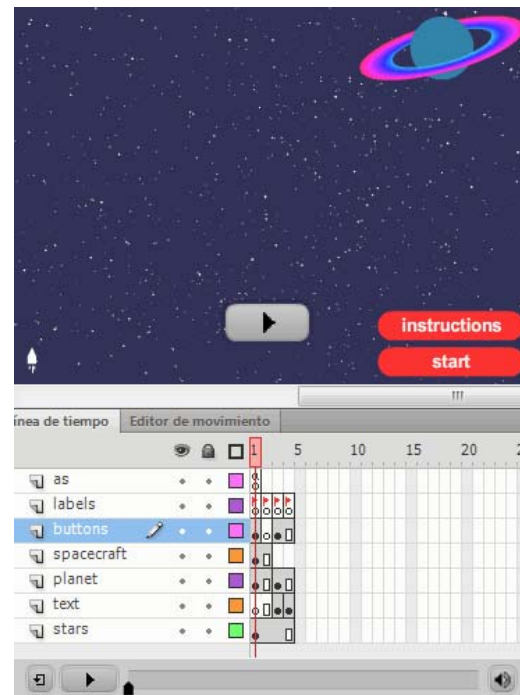
Tornem a l'escena principal. Els botons *instructions* i *start* només s'han de veure en el primer fotograma, ja que durant el joc no han de ser visibles.

Creem el botó que ens falta, que anomenarem *replay*. Serà un botó igual que *start*, llevat que en el text escrivim *play again*. Aquest botó haurà de ser visible en els fotogrames *win* i *lose* (3 i 4).

Col·loquem aquest tercer botó en el tercer fotograma de la capa *buttons*, en la mateixa posició que tenia el botó *start* en el primer fotograma i li donem el nom d'instància *replay_btn*. Al botó *instructions* no li vam donar nom d'instància perquè no portarà associada cap programació.

Inserim un fotograma en blanc en el segon fotograma (en el qual te lloc el joc) perquè en aquest fotograma no es mostri cap botó.

Repassant el contingut que hem creat, en el primer fotograma (*start*) es veuran, a més de la nau i el planeta, els botons *instructions* i *start*. En el segon fotograma (*game*) només veurem la nau i el planeta. En el tercer fotograma (*win*) i en el quart fotograma (*lose*), veurem el botó *replay*, i també una imatge ampliada del planeta en el centre de l'escenari, amb diferents textos segons es guanya o es perd. Les estrelles seran el fons comú dels quatre fotogrames.



Paso 23 de 26

Ara que tenim els botons creats, necessitem programar-ne el funcionament. Tornem al fotograma 1 de la capa *as* i premem **F9** per obrir el panell **Acciones**.

En primer lloc, en iniciar la nostra pel·lícula volem que s'aturi al primer fotograma, motiu pel qual inclourem la instrucció `stop()`. Podem col·locar aquesta instrucció, per exemple, en la primera línia de la nostra programació.

Ara el moviment de la nau no s'ha de permetre directament, sinó que s'ha d'activar quan premem el botó *start_btn*. D'una altra banda, el botó *start_btn* també ens ha de portar al fotograma 2, etiquetat com a *game*, en què no es mostra cap botó.

Per tant, afegim un listener al botó *start_btn*, de tal manera que executi una funció que anomenarem `playGame` quan hi fem clic.

```
start_btn.addEventListener(MouseEvent.CLICK, playGame);
```

Les sentències que havíem creat per seleccionar (posar el focus) i afegir els listeners a la nau han de ser ara dins de la funció `playGame`.

La funció `playGame` quedarà, per tant, de la manera següent:

```
function playGame(e:MouseEvent):void
{
    gotoAndStop("game");
    spacecraft_mc.focusRect = false;
    stage.focus = spacecraft_mc;
    spacecraft_mc.addEventListener(KeyboardEvent.KEY_DOWN, arrows);
    spacecraft_mc.addEventListener(Event.ENTER_FRAME,
moveSpacecraft);
}
```

Per al botó *instructions* no hi ha programació, ja que la seva funció és mostrar l'ajuda quan situem el punter sobre el botó, i això ho hem aconseguit mostrant l'ajuda en l'estat *Sobre* del botó.

Ara canviem les sentències `trace` de la funció `moveSpacecraft` per les sentències següents segons correspongui:

```
gotoAndStop("win");
gotoAndStop("lose");
```

Si no haguéssim posat etiquetes en els fotogrames, les instruccions serien `gotoAndStop(3)` en un cas i `gotoAndStop(4)` en l'altre cas.

Pas 24 de 26

Tot i que encara que ens falta per programar el botó per tornar a jugar, si ara provem la pel·lícula i arribem al planeta, comprovarem que després de mostrar el fotograma que indica que hem guanyat, al cap de poca estona es mostrarà el fotograma que hem perdut.

Això es deu al fet que, malgrat que no veiem la nau en els fotogrames 3 i 4, no hem eliminat els listeners, per la qual cosa es continuen calculant posicions per a la nau. És a dir, que després d'arribar al planeta, la posició de la nau sortirà per un extrem de l'escenari, i això farà que vegem el fotograma que indica que hem perdut.

Per a solucionar aquest problema, abans del `gotoAndStop` que envia el cap lector als fotogrames *win* o *lose*, hem d'eliminar els listeners que havíem creat per a la nau.

```
spacecraft_mc.removeEventListener(KeyboardEvent.KEY_DOWN, arrows);
spacecraft_mc.removeEventListener(Event.ENTER_FRAME, moveSpacecraft);
```

Després de la sentència `gotoAndStop`, que enviarà el cap lector als fotogrames del final, podem afegir el listener per habilitar el botó *replay*, ja que és llavors quan el botó apareixerà en escena:

```
replay_btn.addEventListener(MouseEvent.CLICK, replay);
```

Per tant, els passos que es faran, tant si guanyem com si perdem, seran en primer lloc esborrar els listeners de la nau, després anar al fotograma *win* o *lose*, segons el cas, i finalment afegir un listener al botó *replay*.

Llevat del nom del fotograma al qual ens dirigim, les instruccions que donem són les mateixes en tots dos casos (guanyar o perdre). Per a no repetir la programació en dos llocs diferents (en els dos condicionals `if` dins de `moveSpacecraft`), el més adequat en aquests casos és crear una funció independent que contingui aquestes sentències.

Per a solucionar el fet que calgui anar en un fotograma diferent en cada cas, podem afegir un paràmetre a la funció, de tal manera que rebi el nom del fotograma on ha d'anar.

Així podríem incloure en la crida a la funció, que anomenarem `gameOver`, el nom del fotograma de la manera següent (el mateix però amb *lose* en l'altre cas):

```
if (e.target.hitTestObject(planet_mc))
{
    gameOver("win");
}
```

Paso 25 de 26

La funció, que rep un paràmetre amb el nom del fotograma, tindrà aquesta definició:

```
function gameOver(frameLabel:String):void
{
    spacecraft_mc.removeEventListener(KeyboardEvent.KEY_DOWN,
arrows);
    spacecraft_mc.removeEventListener(Event.ENTER_FRAME,
moveSpacecraft);
    gotoAndStop(frameLabel);
    replay_btn.addEventListener(MouseEvent.CLICK, replay);
}
```

Como veiem, el paràmetre que rep la funció l'hem anomenat `frameLabel`, i a la sentència `gotoAndStop` li indiquem que vagi a un fotograma amb el valor del paràmetre rebut (en el nostre cas, els valors rebuts són "win" o "lose").

La funció `replay`, que es cridarà quan premem el botó `replay_btn`, només indicarà que tornem al fotograma inicial:

```
function replay(e:MouseEvent):void
{
    gotoAndStop("start");
}
```

Resumint la programació creada en la primera part del joc, tenim en primer lloc un `stop` i la creació de les variables `course` i `speed`.

Després afegim un listener al botó `start_btn`.

Per acabar, tenim creades cinc funcions, que es cridaran en diferents moments. Les funcions que hem creat són:

- `playGame`, activa el moviment de la nau.
- `arrows`, fa girar la nau i indica la direcció del moviment.
- `moveSpacecraft`, desplaça la nau i avalua si ha arribat al planeta o ha sortit fora de l'escenari.
- `gameOver`, esborra els listeners, va a la pantalla final i prepara el botó per tonar a jugar.
- `replay`, va al fotograma inicial del joc.

En el tutorial següent afegirem complexitat al joc afegint obstacles que es mouran de manera aleatòria per l'escenari.

Pas 26 de 26

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

1. Crea en un nou document un dibuix d'un cos humà utilitzant botons, de tal manera que, quan situem el punter sobre cada àrea del cos, l'àrea ressalti i es mostri un requadre amb informació sobre aquesta.
2. Crea un joc nou amb diferents nivells, en el qual puguem moure un objecte amb el teclat i en arribar a un objectiu passem a una altra pantalla. En cada nova pantalla augmentem la velocitat.

