



**MP_0489. Programación
multimedia y dispositivos móviles
UF5. Desarrollo de juegos 2D y 3D**

**5.4. Proyecto en 2D:
animaciones**

Índice

☰	Objetivos	3
☰	Crear animaciones	4
☰	Crear transiciones	8
☰	Crear controles	12
☰	Resumen	19

Objetivos

Con esta unidad perseguimos los siguientes objetivos:

1

Aprender a hacer animaciones en Unity.

2

Utilizar esas animaciones para distintas situaciones del juego.

3

Controlar la posición de nuestro jugador.

¡Ánimo y adelante!

Crear animaciones

A lo largo de esta unidad seguiremos practicando efectos como la animación con nuestro ejemplo del astronauta.

En esta ocasión, nuestro nuevo objetivo será **lograr que el astronauta corra.**

Crear animaciones

Para empezar a trabajar con animaciones, debemos abrir la ventana *Animación*, si es que aún no la tenemos abierta.

1

Una vez ahí, seleccionamos *Ventana ▶ Animación*, para abrir la vista *Animación*.

2

Antes de crear la primera animación, **crearemos una carpeta de animaciones en el panel de Proyecto** y nos aseguraremos de que esté seleccionada.

3

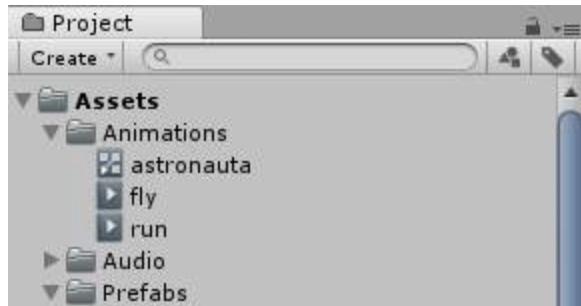
A continuación, **seleccionamos el GameObject del astronauta en el panel de Jerarquía**, ya que las nuevas animaciones se agregarán al objeto seleccionado más recientemente.

4

En la ventana *Animación* nos piden que creamos un animador y un clip de animación para comenzar. Hacemos clic en *Create* y nombramos la primera animación como *run*.

5

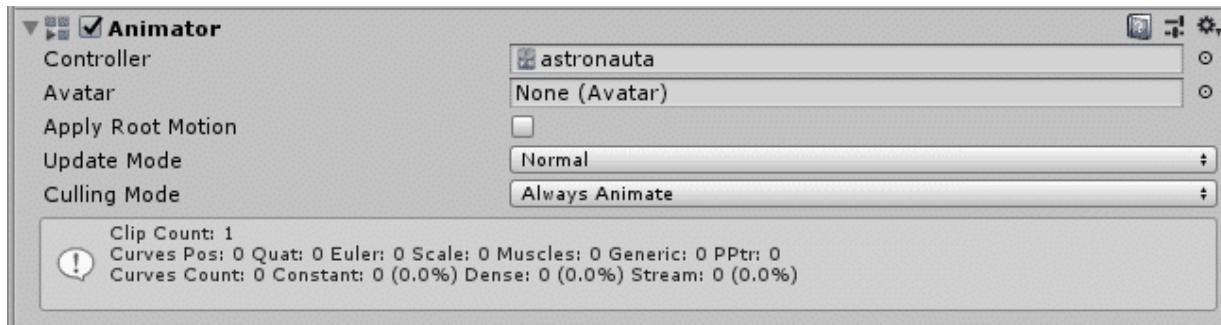
Creamos un segundo clip, llamado *fly*, seleccionando *Create new clip* en el menú desplegable de la esquina superior izquierda.



Vemos tres nuevos archivos creados en el panel de *Proyecto*. Además de las dos animaciones de *fly* y *run*, también hay un archivo animador, llamado *astronauta*.

6

Seleccionamos el *astronauta* en el panel de *Jerarquía*. En el panel de *Inspector*, verás que se nos agregó automáticamente un componente *Animator*.



Añadir cuadros de animación

Primero, vamos a agregar marcos a la animación *run*. Nos aseguramos de que tanto la vista *Animación* como la vista *Proyecto* estén visibles.

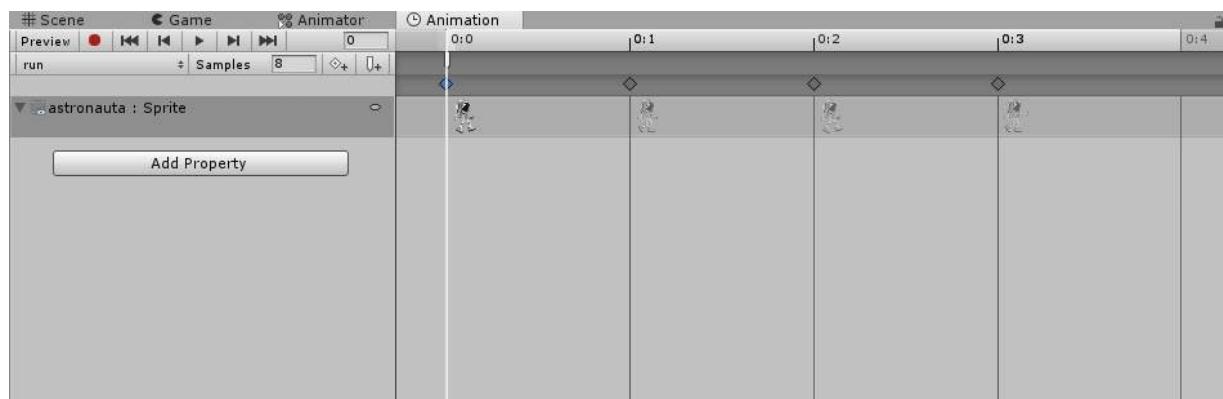
1

En el panel de *Animación*, seleccionamos la animación *run*.

2

Seleccionamos todos los cuadros de animación de ejecución: *astronauta_run_0*, *astronauta_run_1*, *astronauta_run_2*, *astronauta_run_3*. Arrastramos los cuadros a la línea de tiempo de la vista *Animación*.

Así debe verse la línea de tiempo tras agregar los marcos:



Añadir el cuadro de animación *Fly*

Lo creas o no, la animación *fly* consiste en un solo cuadro.

1

Seleccionamos la animación *fly* en el panel de *Animación*.

2

En el panel de *Proyecto*, buscamos el *sprite astronauta_fly* y lo arrastramos a la *línea de tiempo*, tal como hicimos con la animación *run*, salvo que esta vez solo necesitamos agregar un *sprite*.



¿Por qué creamos una animación con un solo cuadro? Porque esto hace que sea mucho más fácil cambiar entre los estados de astronauta corriendo y volando, mediante las transiciones de *Animator*.

Ajustar el animador y ejecutar la configuración de animación

Ahora vemos que el astronauta no deja de correr a una velocidad endiablada.

Dado que la animación *run* se agregó primero, el componente *Animator* la estableció como la animación predeterminada. Por tanto, la animación comienza a reproducirse tan pronto como se ejecuta la escena.

1

Para corregir la velocidad de la animación, seleccionamos la animación de ejecución en el panel de *Animación* y establecemos la propiedad *Muestras* en 8 en lugar de 60.

2

Seleccionamos el *GameObject* del astronauta en el panel de *Jerarquía* y buscamos el componente *Animator* en el panel de *Inspector*.

3

Seleccionamos el cuadro desplegable *Update mode* y cambiamos *Normal* a *Animate Physics*.



Como el juego utiliza la física, es buena idea mantener las animaciones sincronizadas con la física.

Crear transiciones

El astronauta sigue caminando, incluso mientras está en el aire. Para arreglar esto, necesitamos crear algunas transiciones de animación.

Veamos cómo se hace.

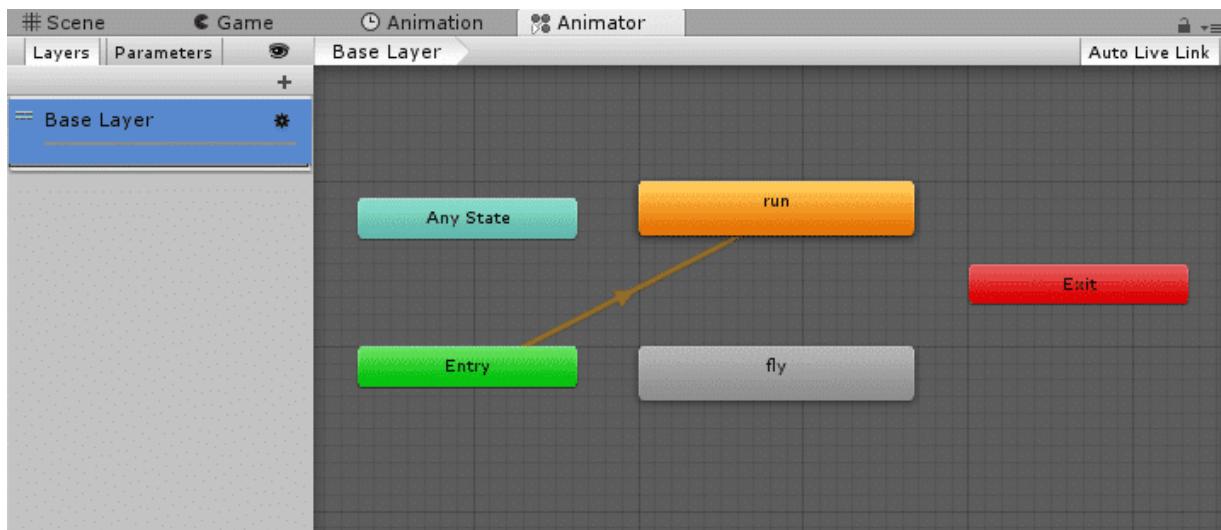
Creación de transiciones de animación

Para usar las transiciones del animador, necesitaremos una ventana más de Unity.

1

En el menú superior, elegimos **Ventana ▶ Animador** para agregar la vista del animador y nos aseguramos de tener seleccionado el *GameObject* del astronauta en el panel de *Jerarquía*.

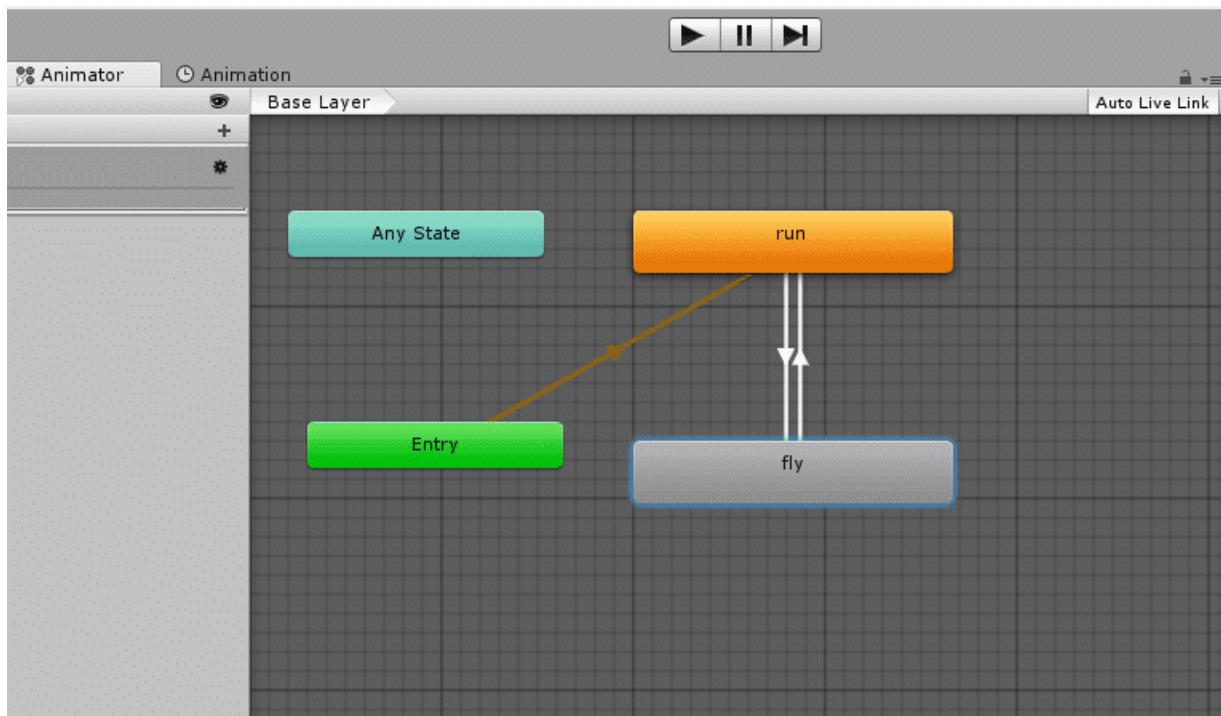
Actualmente tenemos dos animaciones allí: *run* y *fly*. La animación *run* es naranja, lo que significa que es la animación predeterminada.



Sin embargo, no hay transición entre las animaciones de *run* y *fly*. Esto significa que el astronauta está atascado para siempre en el estado de animación *run*. Para solucionar este problema, debemos agregar dos transiciones: una de *run* a *fly*, y otra al contrario.

- 2 Para agregar una transición de *run* a *fly*, hacemos clic con el botón derecho en la animación *run* y seleccionamos *Make Transition*.
- 3 Luego, colocamos el cursor sobre la animación *fly* y hacemos clic con el botón izquierdo sobre ella.
- 4 De manera similar, para agregar una transición de *fly* a *run* hacemos clic derecho en la animación *fly*. Seleccionamos *Make Transition*, pero esta vez ponemos el cursor sobre la animación *run* y hacemos clic con el botón izquierdo.

Hemos creado dos transiciones incondicionales, lo que significa que cuando ejecutamos la escena, el astronauta primero ejecutará su estado *run*, pero después de reproducir la animación *run* una vez, cambiará al estado *fly*. Una vez que se completa el estado *fly*, volverá a *run* y así sucesivamente.



Agregar un parámetro de transición

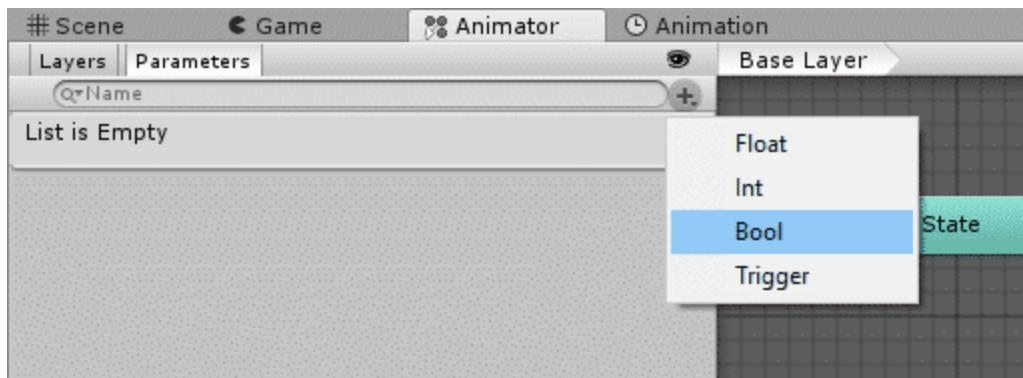
Para romper este círculo vicioso, **debemos agregar una condición que controle cuándo la animación *fly* debería pasar a la animación *run*, y viceversa.**

1

Abrimos la vista *Animator* y en el panel *Parameters* de la esquina superior izquierda, que actualmente está vacío, hacemos clic en el botón **+** para agregar un parámetro.

2

En el menú desplegable, seleccionamos *Bool*.



3

Nombramos el nuevo parámetro *isGrounded*.

4

Seleccionamos la transición *run* a *fly* para abrir las propiedades de transición en el panel de *Inspector*. En la sección *Conditions*, hacemos clic en el signo + para agregar *isGrounded* y establecer su valor en *false*.

Mientras esté así, evitará cualquier retraso o transición entre los estados de animación.

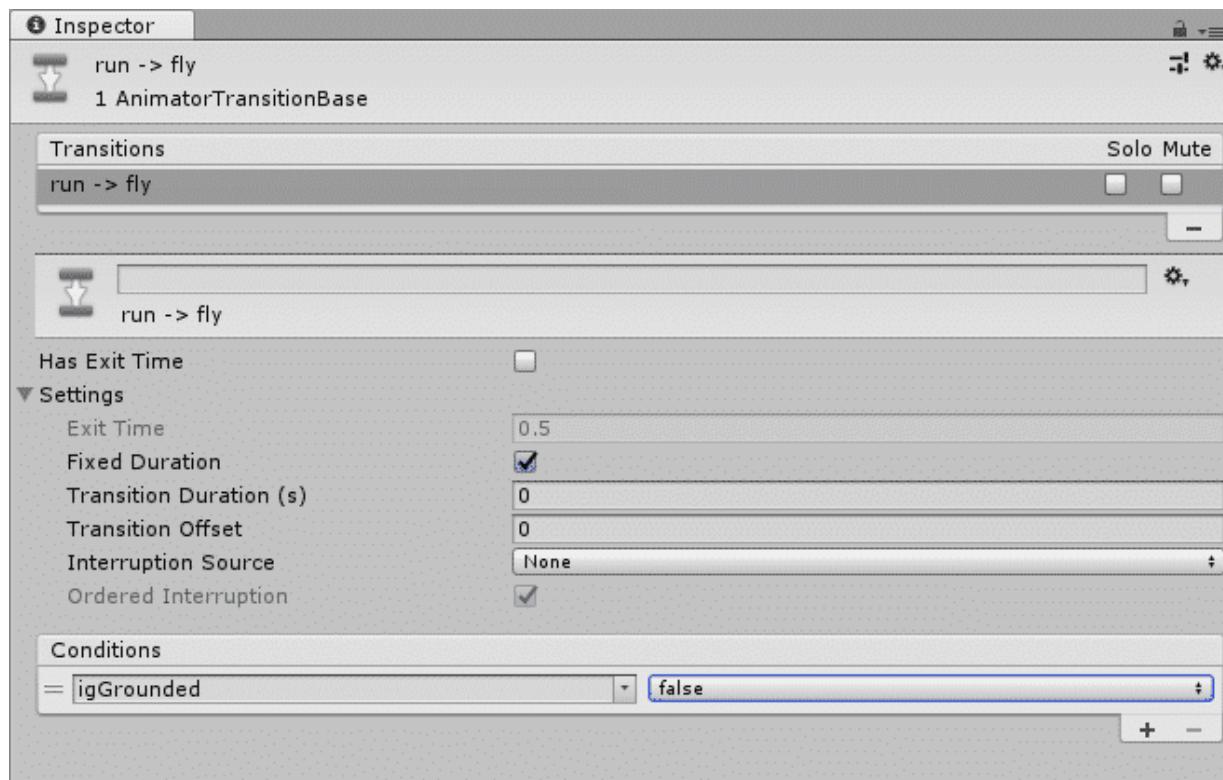
5

Desmarcamos *Has Exit Time* y hacemos clic en la flecha de divulgación para expandir la configuración de la transición. Establecemos la *Transition duration* en 0.

6

Hacemos lo mismo con la transición de *fly* a *run*, pero esta vez establecemos el valor *isGrounded* en *true*.

De esta manera, el estado del astronauta cambiará a *fly* cuando *isGrounded* sea *false*, y a *run* cuando *isGrounded* sea *true*.



Crear controles

Por último, vamos a comprobar si el astronauta está tocando suelo. Y hay muchas formas de verificar si el objeto del juego está conectado al suelo.

El siguiente método proporciona una **representación visual del punto donde verificamos el terreno**, y puede ser muy útil cuando tenemos muchos controles diferentes, como el control del suelo, el control del cielo u otros.

Obtener una representación visual

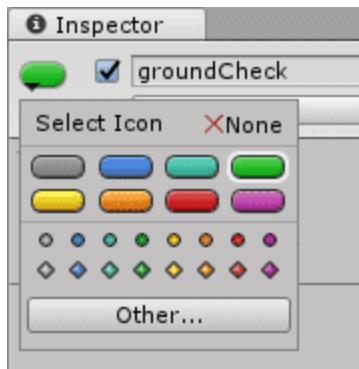
Lo que da a este método una representación visual es un *GameObject* vacío, agregado como un hijo del personaje del jugador.

1

Seleccionamos este *GameObject* en el panel de *Jerarquía* y **le cambiamos el nombre a *groundCheck***. Establecemos su posición en (0, -0.7, 0).

2

Para hacerlo visible en la escena, hacemos clic en el botón de *Select Icon* en el panel de *Inspector* y establecemos su ícono en el óvalo verde. Podemos elegir cualquier color.



El script *AstronautaController* utilizará la posición de este *GameObject* vacío para verificar si está en el suelo.

Usar capas para definir qué es tierra

Antes de que podamos comprobar que el astronauta está en el suelo, **debemos definir qué es el suelo**. Utilizaremos la clase *LayerMask* en el script, pero para usarla, primero debemos establecer el suelo en la capa correcta.

1

Abrimos la carpeta *Prefabs* en el panel de *Proyecto* y expandimos la *Prefab luna1*. Seleccionamos el suelo dentro del *Prefab*.

2

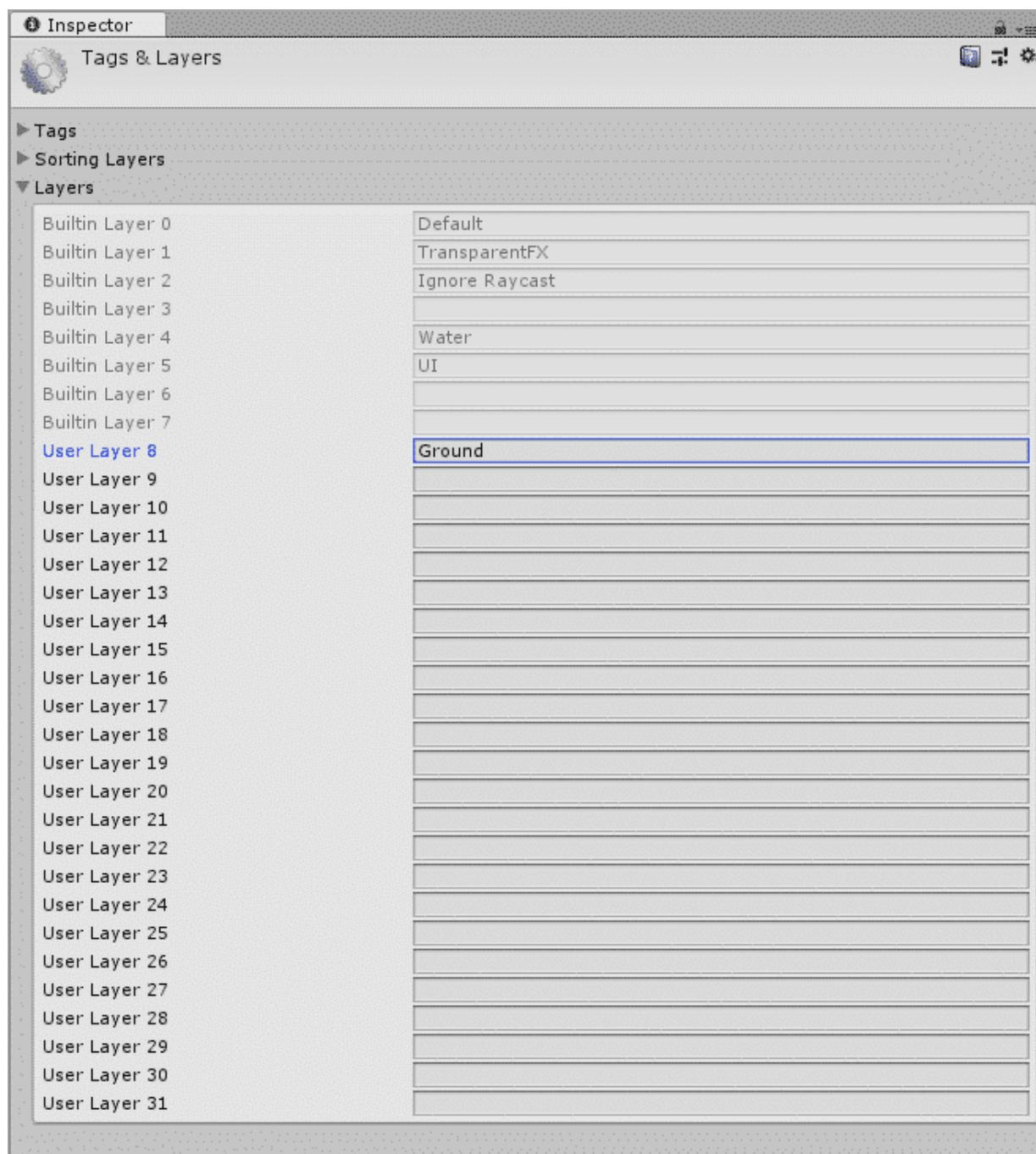
En el panel de *Inspector*, hacemos clic en el menú desplegable *Layer* y elegimos la opción *Add layer*. Esto abrirá el editor *Tags & Layers* en el panel de *Inspector*.

3

Buscamos el primer elemento editable, *User Layer 8*, y escribimos *Ground* en él. Todas las capas anteriores están reservadas por Unity.

4

A continuación, seleccionamos el suelo dentro de la *Prefab luna1* una vez más y establecemos su *Layer* a *Ground*.



Comprobar si el astronauta está tocando suelo

Para hacer que el astronauta cambie automáticamente de estado, deberemos actualizar el script *AstronautaController* para verificar si el astronauta está actualmente tocando suelo y, luego, avisar al *Animator*.

1

Abrimos el script *AstronautaController* y agregamos las siguientes variables de instancia:

```
public Transform groundCheckTransform;  
private bool isGrounded;  
public LayerMask groundCheckLayerMask;  
private Animator astronautaAnimator;
```

2

- La variable *groundCheckTransform* almacenará una referencia al *groundCheck Empty GameObject* que creamos anteriormente.
- La variable *isGrounded* indica si el astronauta está tocando suelo, mientras que *groundCheckLayerMask* almacena una *LayerMask* que define qué es el suelo.
- Finalmente, la variable *astronautaAnimator* contiene una referencia al componente *Animator*.

2

Para almacenar en caché el componente *Animator*, agregamos la siguiente línea de código a *Start*:

```
astronautaAnimator = GetComponent<Animator>();
```

3

Ahora agregamos *UpdateGroundedStatus*:

```
void UpdateGroundedStatus()  
{  
    isGrounded = Physics2D.OverlapCircle(groundCheckTransform.position, 0.1f,  
    groundCheckLayerMask);  
    astronautaAnimator.SetBool("isGrounded", isGrounded);  
}
```

Este método verifica si el astronauta está tocando suelo y establece el parámetro *Animator* de la siguiente manera:

- Para verificar si el astronauta está tocando suelo, creamos un círculo de radio 0.1 en la posición del objeto *GroundCheck* que agregamos a la escena. Si este círculo se superpone a cualquier objeto que tenga una capa especificada en *groundCheckLayerMask*, entonces el astronauta está tocando suelo.
- Este código realmente establece el parámetro *isGrounded* del *Animator* que luego activará la animación.

4

Finalmente, agregamos una llamada a *UpdateGroundedStatus* al final del método *FixedUpdate*.

```
UpdateGroundedStatus();
```

Esto llama al método con cada actualización, asegurando que comprobamos si se toca suelo de manera consistente.

Configuración de los parámetros de script de *AstronautaController* para *Ground Check*

Solo queda un pequeño paso para que el astronauta cambie automáticamente entre volar y correr.

1

Abrimos Unity y seleccionamos el astronauta *GameObject* en el panel de *Jerarquía*.

2

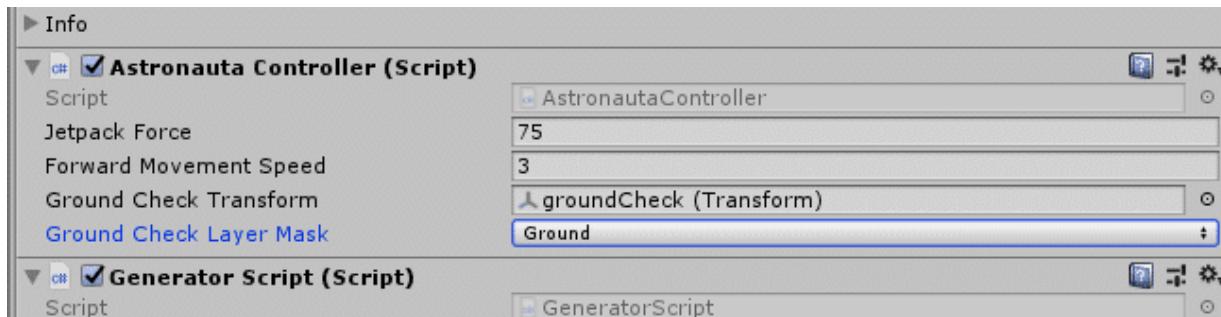
Buscamos el componente de *Script* del controlador de astronauta. Veremos dos nuevos parámetros en el panel de *Inspector*.

3

Hacemos clic en el menú desplegable *Ground check layer mask* y seleccionamos *Ground*.

4

Arrastramos el *groundCheck* desde el panel de *Jerarquía* a la propiedad *Ground Check Transform*.



Activar y desactivar las llamas Jetpack

1

Abrimos el script *AstronautaController* y agregamos la siguiente variable *jetpack* para almacenar una referencia al *Particle System*.

```
public ParticleSystem jetpack;
```

2

Luego, agregamos el siguiente método *AdjustJetpack*:

```
void AdjustJetpack(bool jetpackActive)
{
    var jetpackEmission = jetpack.emission;
    jetpackEmission.enabled = !isGrounded;
    if (jetpackActive)
    {
        jetpackEmission.rateOverTime = 300.0f;
    }
    else
    {
        jetpackEmission.rateOverTime = 75.0f;
    }
}
```

Este método desactiva la emisión del *jetpack* cuando el astronauta está tocando el suelo.

También disminuye la tasa de emisión cuando el astronauta se está cayendo, ya que el *jetpack* aún podría estar activo, pero no a plena potencia.

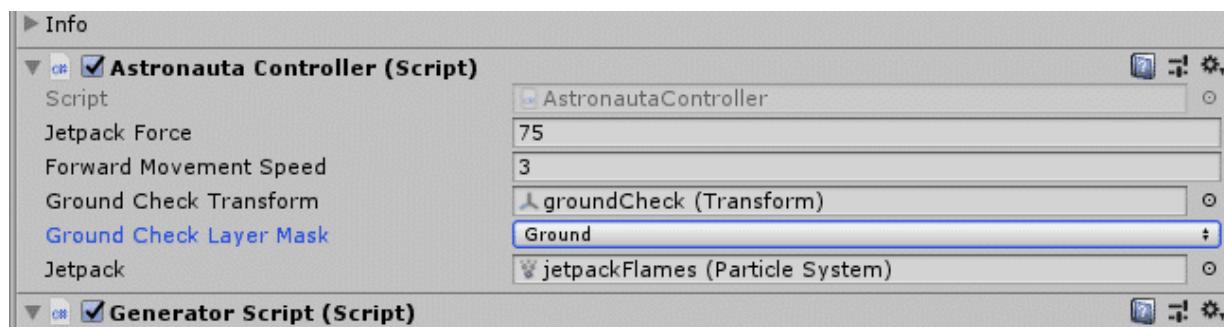
3

Agregamos una llamada a este método al final de *FixedUpdate*:

```
AdjustJetpack(jetpackActive);
```

4

Ahora, volvemos a Unity y arrastramos los *jetpackFlames* del astronauta desde el panel de *Jerarquía* a la propiedad *Jetpack* del componente *AstronautaController*.



Ahora el *jetpack* tiene tres estados diferentes:

1. Está deshabilitado cuando el astronauta está tocando el suelo.
2. Está con una fuerza total cuando el astronauta sube.
3. Y se ejecuta a una tasa de emisión reducida cuando el astronauta está bajando.

Resumen

Hemos terminado la lección, repasemos los puntos más importantes que hemos tratado.

- A lo largo de esta unidad hemos aprendido a **crear animaciones**, utilizando nuestro ejemplo del astronauta.
- Hemos creado, además, **transiciones de las animaciones** para que resulten más reales.
- Y, para finalizar, hemos aprendido a **controlar la situación del astronauta cuando toca el suelo**.



PROEDUCA