

**MP_0489 Programación
multimedia y dispositivos móviles
UF4. Análisis de motores de juegos**

4.6. Partículas

Índice

☰	Objetivos	3
☰	Uso del sistema de partículas	4
☰	Ejemplo	10
☰	Resumen	20

Objetivos

Con esta unidad perseguimos los siguientes objetivos:

- 1 Conocer el sistema de partículas en Unity.
- 2 Descubrir algunas características del sistema de partículas.
- 3 Realizar un ejemplo con el sistema de partículas.

¡Ánimo y adelante!

Uso del sistema de partículas

Las partículas son pequeñas imágenes simples o mallas que se muestran en gran número.

Funcionan gracias a un sistema en el que **cada partícula representa una pequeña porción de un fluido o entidad amorfa**. El efecto de todas las partículas juntas crea la impresión de la entidad completa.



El sistema de partículas de Unity es robusto y está lleno de características.

Los sistemas de partículas permiten que podamos agregar un "dinamismo" adicional a lo que estamos viendo con una pequeña cantidad de partículas. Afortunadamente, Unity hace que la creación de estos sistemas sea bastante simple, con un **sistema modular de partículas incorporado llamado Shuriken**.

Además, los sistemas de partículas son componentes que podemos agregar a cualquier **GameObject** en la escena.

Cómo funciona el sistema de partículas en Unity

Cada partícula tiene un tiempo de vida predeterminado, típicamente de unos cuantos segundos, durante el cual puede conllevar varios cambios.

Su vida comienza cuando es generada por el sistema. El sistema emite partículas en posiciones aleatorias dentro una región de espacio formada como una esfera, hemisferio, cono, caja o cualquier malla arbitraria.

La partícula se muestra hasta que su tiempo se agota, momento en el que se elimina del sistema. El ratio de emisión del sistema indica, aproximadamente, cuántas partículas se emiten por segundo, aunque los tiempos exactos de la emisión son asignados ligeramente al azar.

Propiedades del sistema de partículas

Unity implementa el sistema de partículas con un componente, por lo que colocar un sistema en una escena es cuestión de agregar un *GameObject*, o bien agregar el componente a un objeto existente.

Debido a que el componente es bastante complicado, el *Inspector* está dividido en un número de módulos en el que cada uno contiene un grupo de propiedades relacionadas. Algunas de las más importantes son:

- **Duration.** Número de segundos que el sistema emite las partículas.
- **Looping.** El sistema va a emitir las partículas sin parar, sin tener en cuenta el valor del campo *Duration*.
- **Prewarm.** Cuando queremos que las partículas ya estén en la escena al principio. En caso contrario, el jugador las verá nacer.

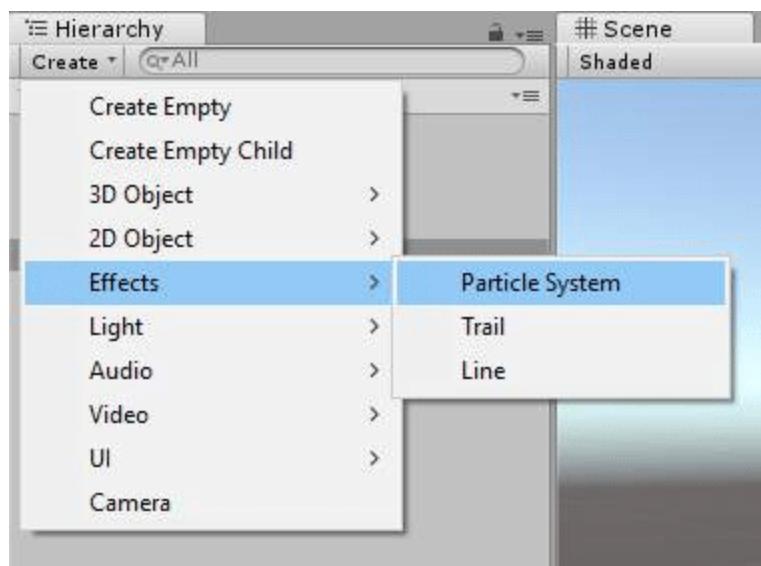
- ***Start Delay.*** Espera con el valor en segundos antes de empezar a emitir las partículas.
- ***Start Lifetime.*** Indica cuántos segundos quedará visible la partícula después de ser emitida.
- ***Start Speed.*** Es la velocidad con la que se lanzará la partícula.
- ***Start Size.*** Tamaño de la partícula.
- ***Start Color.*** Color de la partícula.
- ***Gravity Modifier.*** Aplica fuerza de la gravedad en la partícula, haciéndola caer.
- ***Play On Awake.*** Inicia el sistema de partículas tan pronto como se inicializa en el juego.
- ***Max Particles.*** Número máximo de partículas visibles en la pantalla. Cuando este número es alcanzado, el sistema de partículas para de producirlas, hasta que acabe el tiempo de vida de las otras.

Antes de pasar a un ejemplo más complejo, veamos de una manera sencilla **cómo añadir partículas a un objeto, utilizando el proyecto de la lección anterior:**

Partículas

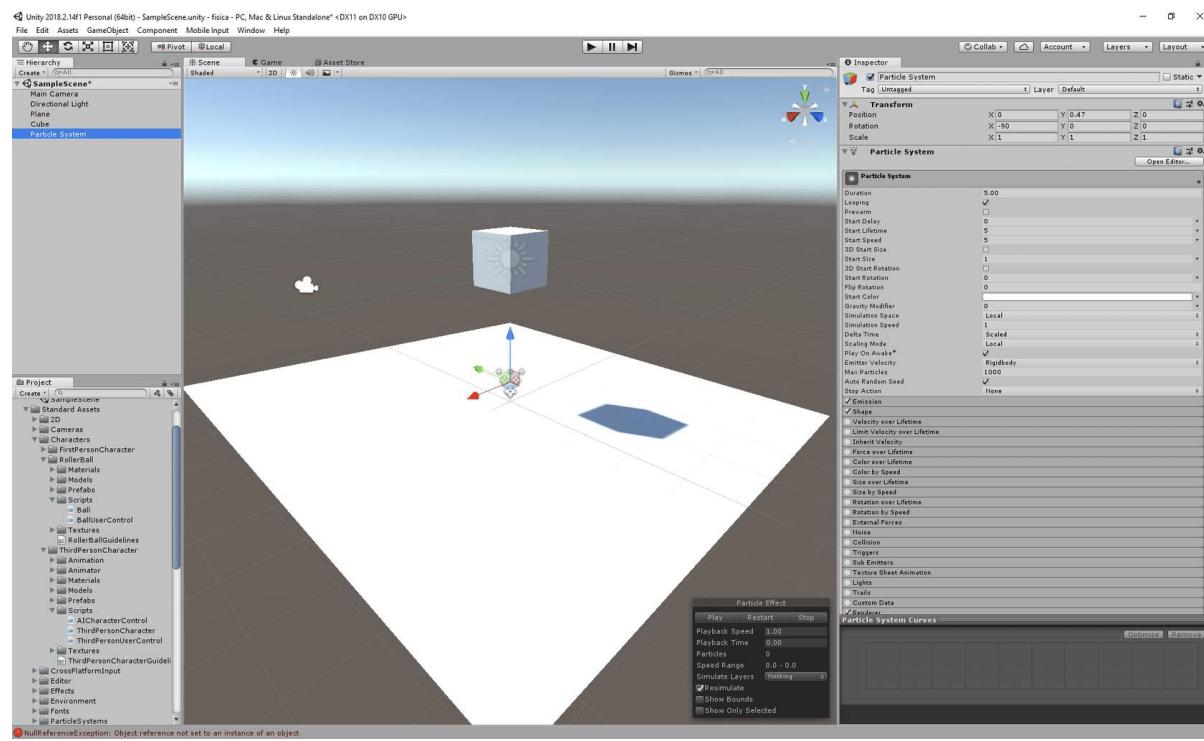
Añadir partículas a nuestro proyecto de Unity.

Paso 1



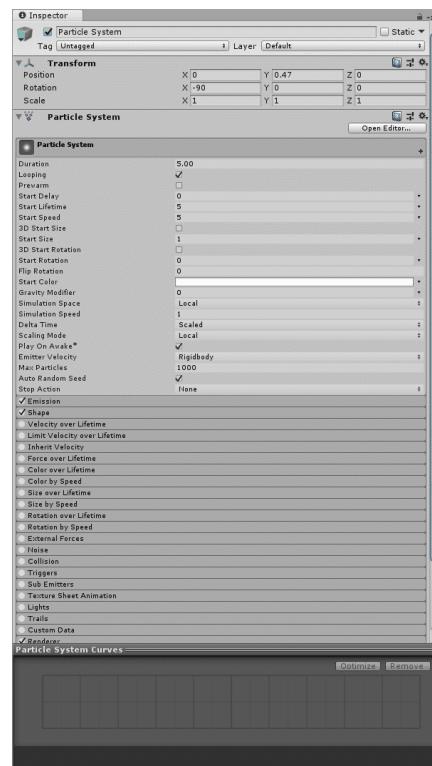
Añadimos el sistema de partículas desde el panel de *Jerarquía*.

Paso 2



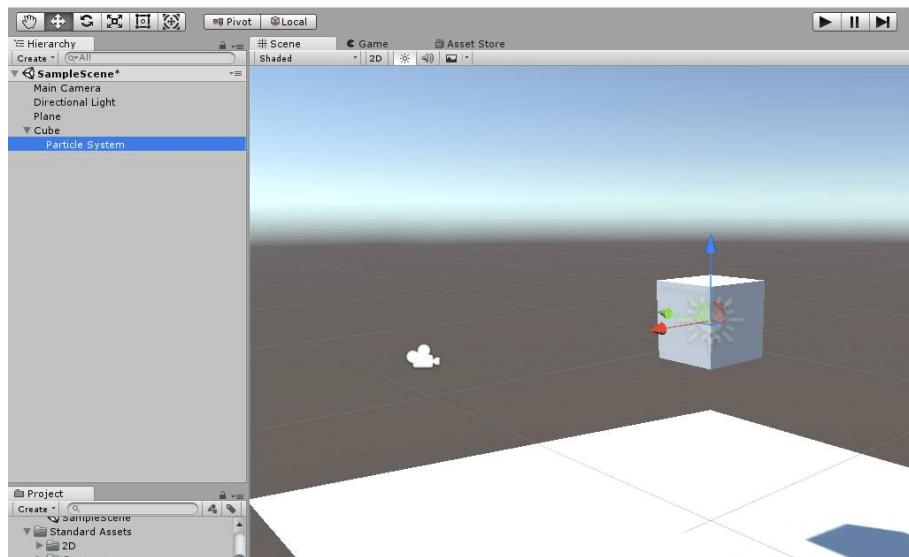
Vemos cómo aparece en nuestra escena.

Paso 3



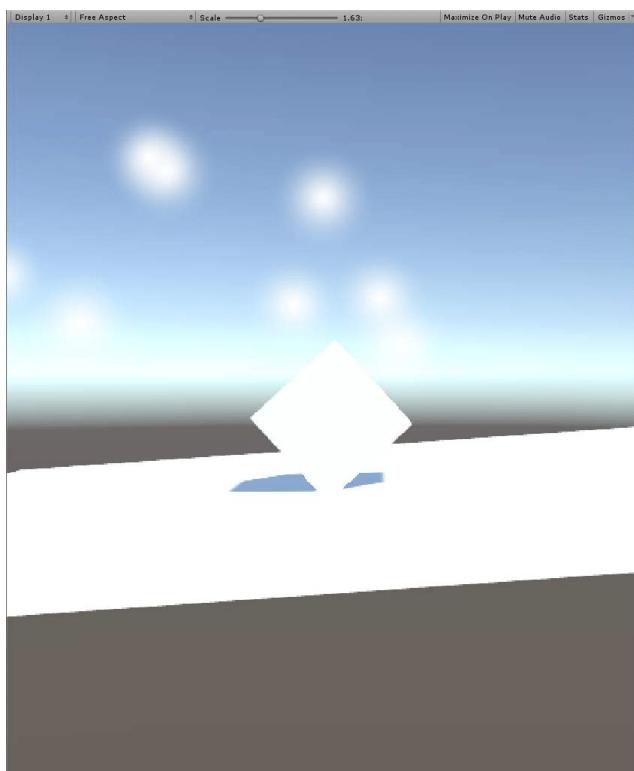
En el panel de *Inspector* podemos observar la cantidad de características distintas que nos vamos a encontrar.

Paso 4



Ahora, en el panel de *Jerarquía*, arrastramos el sistema de partículas "dentro" del cubo y lo situamos en la escena a la misma altura.

Paso 5



Al poner en marcha el juego vemos cómo a la vez que el cubo se mueve, lo acompaña el sistema de partículas.

¡Hecho!

Ya conoces la base para añadir partículas a tu juego.

Ejemplo

La mejor manera de conocer el uso de partículas es mediante un ejemplo. En este caso, crearemos una explosión.

Lograr un efecto de explosión es bastante simple en Unity. Para comenzar, vamos a **descargar el proyecto sobre el que trabajaremos**.



Crear la bomba

1

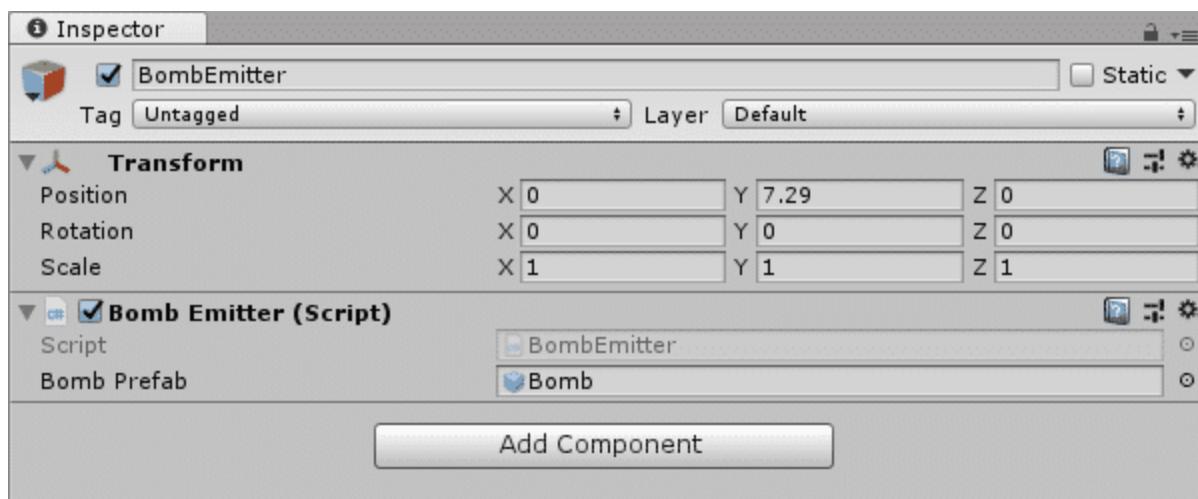
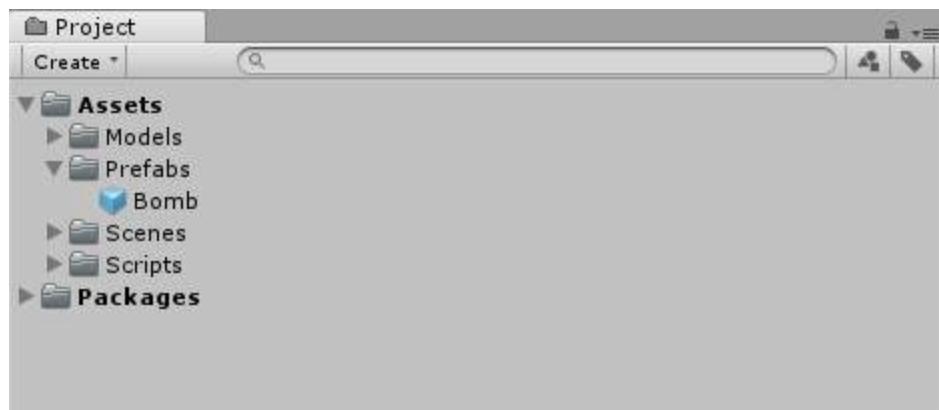
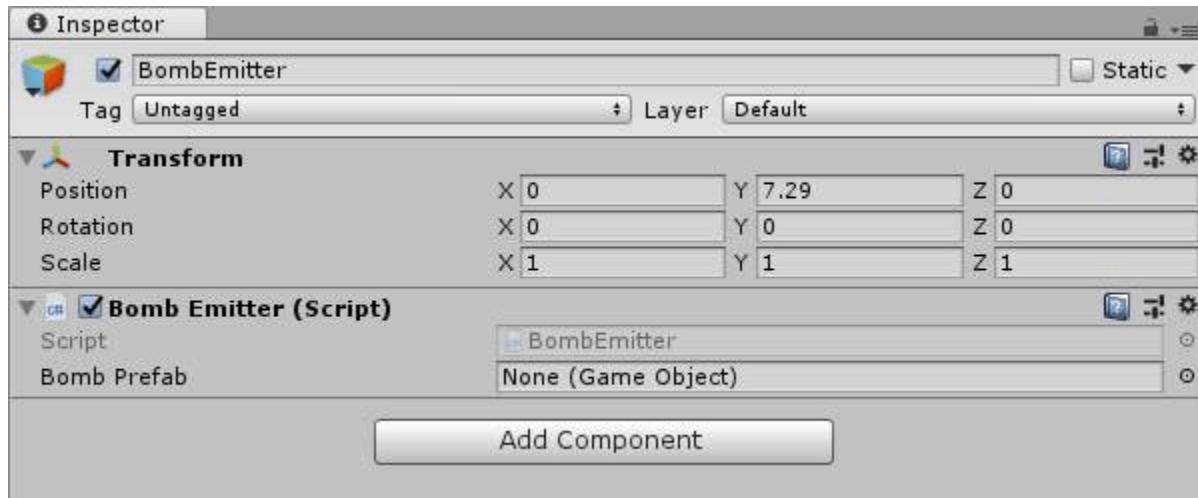
Abrimos la escena de la bomba desde la ventana del proyecto y la reproduccimos:



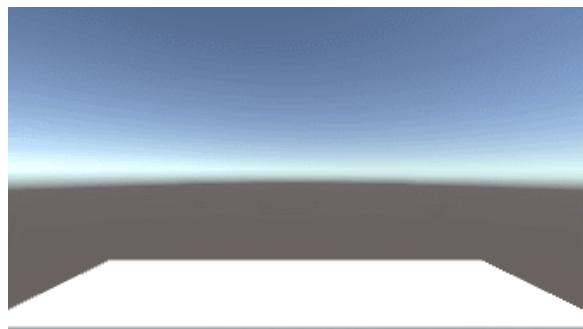
Observamos que hay una base en la escena, pero aparte de eso, nada más.

2

Para generar bombas, seleccionamos *BombEmitter* en el panel de *Jerarquía* y arrastramos el *Prefab* de la bomba, que se encuentra en el panel de *Proyectos*, al hueco del *Bomb Prefab* en el panel de *Inspector* del *Bomb Emitter*.



Si miramos ahora nuestra escena, veremos que el **Bomb Emitter** crea una nueva bomba cada dos segundos.



Agregar rotación

Para darle un toque más real, vamos a agregar un poco de fuerza de rotación a la bomba cuando caiga.

1

Para ello, abrimos el *script* de *Bomb* dentro de la carpeta *Scripts* en el panel de *Proyectos* y añadimos el siguiente código a *Start()*:

```
void Inicio ()
{
    float randomX = UnityEngine.Random.Range (10f, 100f);
    float randomY = UnityEngine.Random.Range (10f, 100f);
    float randomZ = UnityEngine.Random.Range (10f, 100f);

    Rigidbody bomba = GetComponent <Rigidbody> ();
    bomba.AddTorque (randomX, randomY, randomZ);
}
```

2

Las primeras tres líneas generan valores flotantes aleatorios entre 10 y 100 para los ejes x, y y z. Luego, obtenemos una referencia al componente *Rigidbody* de la bomba y le aplicamos una torsión. Esto hace que la bomba gire en una dirección aleatoria.

3

En el panel de *Jerarquía*, pulsamos *Create*, seleccionamos *Create empty* y cambiamos el nombre a *ExplosionParticles*.

4

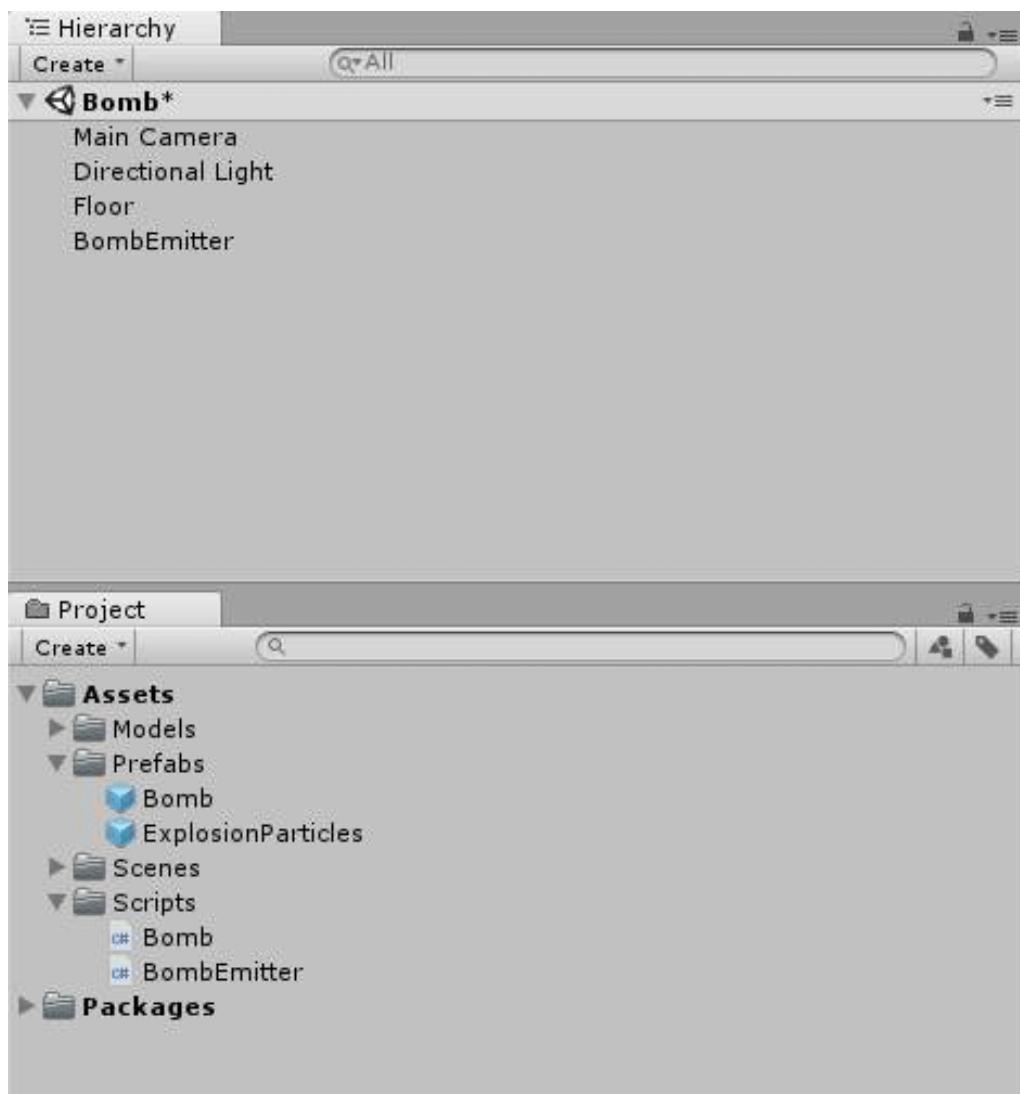
A continuación, agregamos un componente de sistema de partículas al *GameObject*.

5

Arrastramos el *GameObject* de *ExplosionParticles* del panel de *Jerarquía* en la carpeta *Prefabs*, dentro del panel de *Proyectos*.

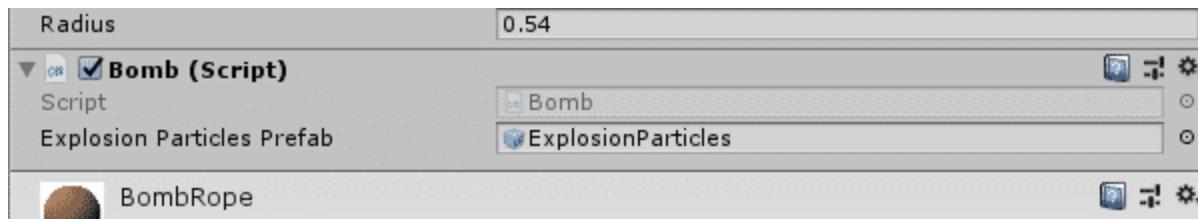
6

Una vez hecho esto, eliminamos *ExplosionParticles* del panel de *Jerarquía*.



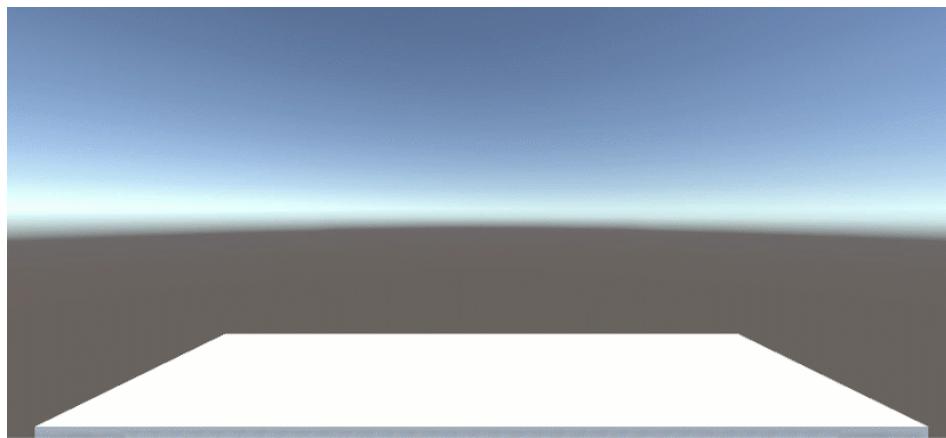
7

A continuación, seleccionamos el *Prefab Bomb* dentro de la carpeta *Prefabs* y arrastramos el *Prefab de ExplosionParticles* al hueco "*ExplosionParticles Prefab*" bomba, así:



Ahora, un nuevo *GameObject* de *Explosion Particles* se generará cuando una bomba toque el suelo.

Veamos la escena y cómo se ve la explosión. Si aparecen manchas rosas, no te preocupes, porque vamos a cambiar la textura.



Cambiar la textura

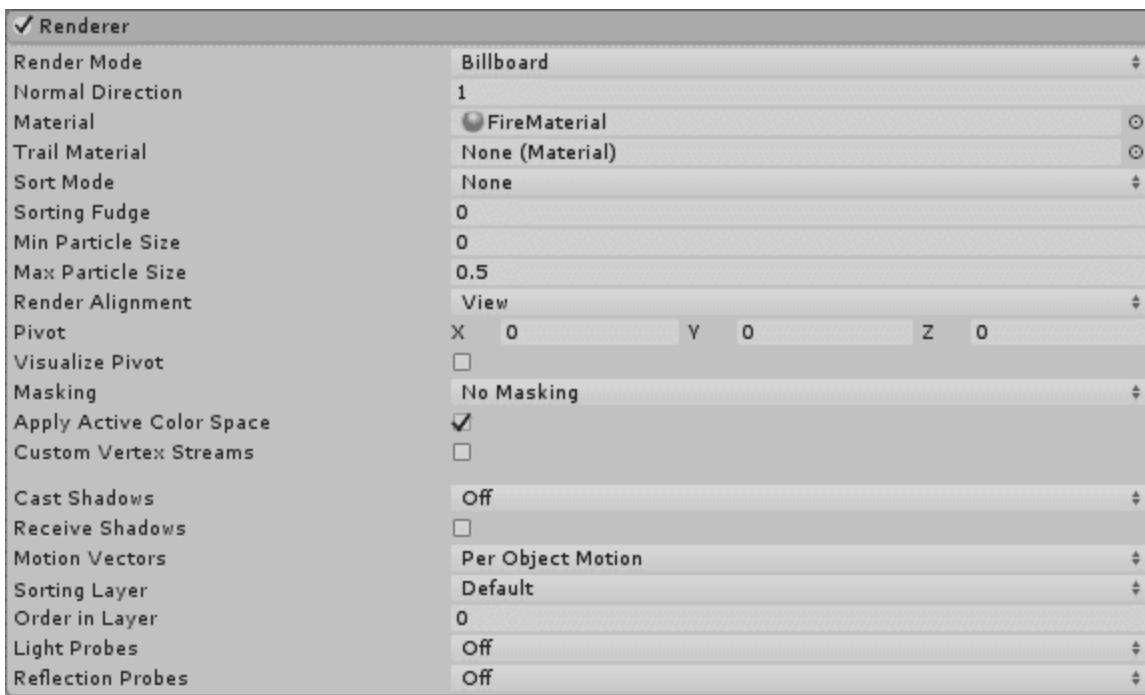
Seguiremos con nuestro ejemplo. En esta ocasión vamos a modificar la textura de la escena.

1

Seleccionamos el *Prefab de ExplosionParticles* en el panel de *Proyecto*, luego expandimos el módulo *Renderer* en el panel de *Inspector*.

2

Arrastramos *FireMaterial* desde la carpeta de *Materials* en el panel de *Proyecto*, hasta un hueco en *Material* en el panel de *Inspector*.



3

Para completar el efecto, **modificaremos las siguientes características** en el módulo principal, en el panel de *Inspector* del *Prefab* de *ExplosionParticles*:

- ***Duration:*** 0.70.
- Desactivamos ***Looping***.
- ***Start Lifetime:*** 0.7.
- ***Start Speed:*** 10.
- ***Start Size:*** 2.
- ***Gravity Modifier:*** 1, para que las partículas caigan más suavemente al final.

4

Para mejorar la explosión, alteraremos las propiedades del módulo *Emisión*.

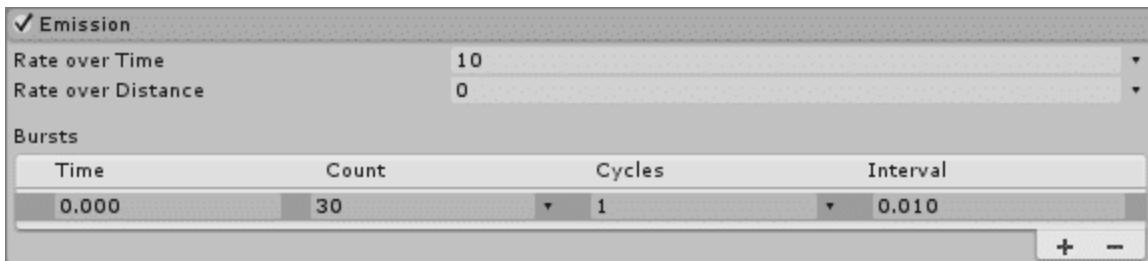
Expandimos el módulo *Emisión*. *Rate* es el número de partículas generadas por segundo. Para esta explosión no queremos un flujo constante de partículas, sino una explosión repentina.

5

Establecemos la *Rate over Time* a 0. Veremos una lista de *Bursts* (ráfagas), que está vacía de forma predeterminada.



Una ráfaga es una colección de partículas emitidas todas a la vez en un momento determinado en el tiempo.



6

Pulsamos el botón + en la parte inferior derecha para agregar una nueva ráfaga.

7

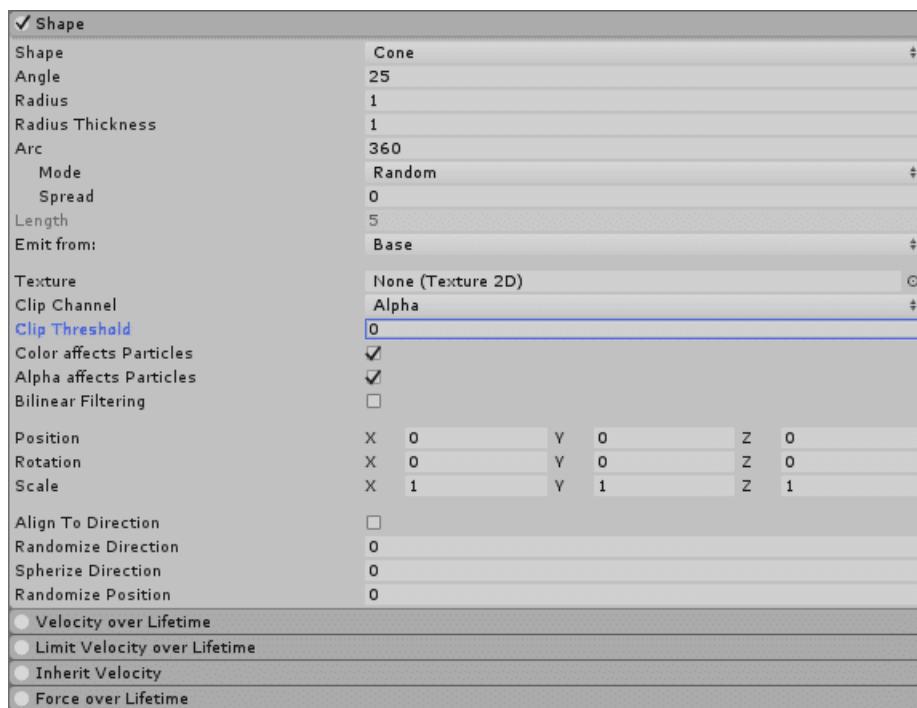
Dejamos *Time* en 0 y establecemos *Count* en 150. Estos ajustes harán que el sistema de partículas emita 150 partículas a la vez al inicio del sistema.

Moldear la forma

A continuación, vamos a moldear la explosión para darle un toque final.

1

Para comenzar, expandimos el Módulo de Forma:

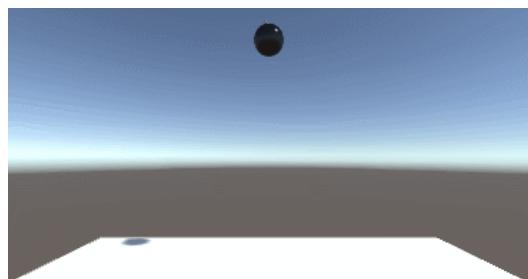


2

Pulsamos en el cuadro desplegable que dice *Cone* para ver **todas las opciones disponibles**. Tenemos muchos efectos diferentes del mismo sistema, simplemente cambiando la forma.

3

Para crear una explosión realista, elegimos la forma en *Sphere*.



Modificar el color

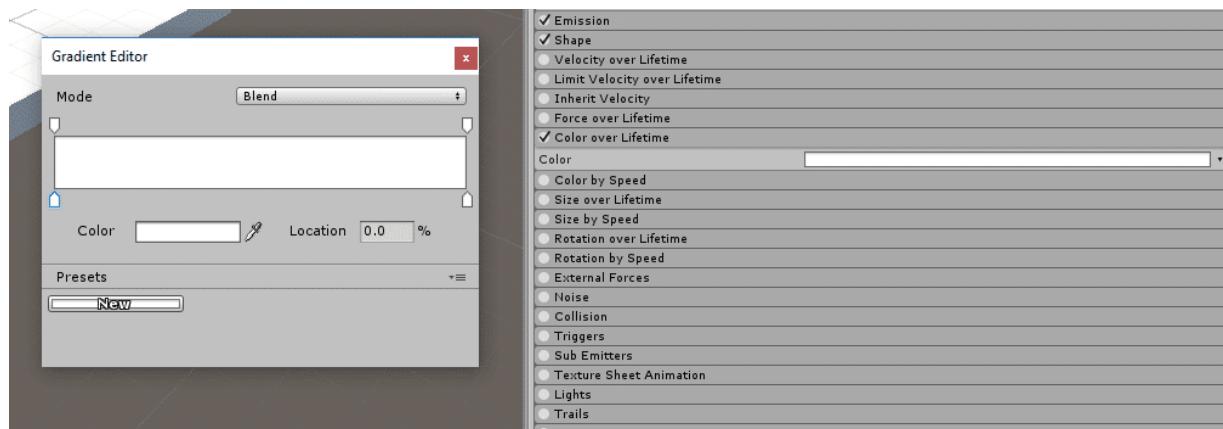
Tenemos un pequeño problema: las partículas simplemente desaparecen. Es un efecto discordante y no parece natural. En lugar de desaparecer, las partículas deberían desvanecerse. Vamos a cambiar el color para lograrlo.

1

Con el sistema de partículas abierto en el *Inspector*, pulsamos en la casilla de verificación que se encuentra al lado del módulo *Color over Lifetime* para habilitarlo y expandirlo. Aparece la palabra *Color* y un bloque blanco al lado.

2

Pulsamos en el bloque blanco y se abre el editor de degradado.



El cambio de color durante la vida útil de las partículas se representa como una **barra de degradado**. El color de inicio está en el extremo izquierdo y las partículas pasarán al color del lado derecho.

Las cuatro flechas blancas en los bordes se conocen como **marcadores**.

- Podemos hacer clic entre dos marcadores existentes para **agregar uno nuevo**.
- Para **eliminar un marcador**, lo arrastramos fuera de la barra.

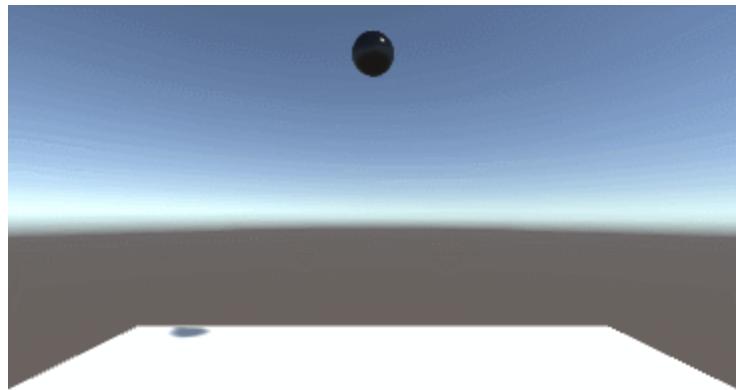
Los marcadores superiores manejan el *alfa* u opacidad del color, mientras que los marcadores inferiores administran los valores de color RGB (rojo, verde, azul).

3

Seleccionamos el marcador *alfa* más a la derecha. La parte inferior del editor de degradado nos muestra el valor *alfa* actual.

4

Movemos esta parte del editor de degradado hasta 0. Las partículas se desvanecerán gradualmente. ¡Ahora sí lo tenemos!



Resumen

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

- Durante el desarrollo de esta unidad hemos aprendido **cómo funcionan los sistemas de partículas en Unity**, así como su utilidad en el desarrollo de nuestros juegos.
- Hemos tenido un primer contacto con algunas características de las partículas.
- Hemos aprendido cómo añadir partículas a un *GameObject*.
- Y para finalizar, hemos creado una explosión como ejemplo para **familiarizarnos con el uso del sistema de partículas** y de todas las características que nos permite modificar.

