* En el método OnTriggerEnter2D del shot controller hemos utilizado GetComponent para comprobar si el objeto colisionado era un enemigo antes de llamar a sus métodos. ¿Conoces alguna forma más sencilla de hacer esta comprobación? ¿Cuál? ¿Y por qué es preferible hacerlo como lo hemos hecho nosotros?
* Hay formas más sencillas, como, por ejemplo, buscar por su nombre (Enemy) o por el tag al que pertenezca. Nosotros hemos aplicado duck typing, que consiste en hacer una búsqueda del componente que tenga un gameObject determinado, asegurando 100% que se trata de un enemigo. (falta explicar y comporbar)
* En el caso de DeathZone no hemos hecho esa comprobación. ¿Por qué no era necesario hacerlo?
* En este caso, la DeathZone sólo puede recibir información de trigger de los enemigos, ya sea porque está marcado así en la matriz de colisión y porque no va a recibir colisión de otro tipo de objeto de la escena (revisar).
* Hemos utilizado multitud de componentes para implementar comportamientos muy sencillos que podrían haberse incluido en un único componente. ¿Por qué es mejor hacerlo así?
* (Poner ejemplos y explicar)
* Hablemos de prefabs anidados:
* Si cambias el color de p\_Enemy, ¿Qué ocurre con los prefabs p\_Squad\_01, p\_Squad\_02...? Y si cambias el color de un enemigo en p\_Squad\_01, ¿Qué ocurre con p\_Enemy?
* En el primer caso, los prefabs p\_squad\_01 y sus relativos, al estar compuestos de p\_Enemies, cambiarían de color al cambiar p\_Enemy. En el caso contrario, no le ocurriría nada a p\_Enemy, ya que en el prefab del escuadrón está compuesto sólo de instancias del p\_Enemy principal, que, al no ser modificado, no tendría ningún cambio.
* ¿Qué método has utilizado para implementar el movimiento de Player, p\_Shot y p\_Squads? ¿Por qué?
* (No sé a que se refiere)
* ¿Qué método has utilizado para que el resto de GameObjects pudieran acceder al GameManager? ¿Cuántas veces en la ejecución de tu juego se busca el GameManager? ¿Es una forma correcta de implementarlo?
* (Haría falta comprobarlo cuando esté en pleno funcionamiento)