«Talento Tech»

Desarrollo de Videojuegos con Unity

CLASE 5

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 05 | Colisiones

Temario:

- Colisiones
- OnCollision
- OnTrigger
- Tags
- gameObject.name

Colisiones.

Ya vimos lo que es un **Collider** en la clase 3, pero aun no lo vimos en acción. Sabemos que si 2 objetos con collider se aproximan, pueden chocar entre ellos y ahora vamos a abordar cómo utilizar este **contacto** por medio de nuestro código, para poder producir circunstancias según lo que queramos en nuestro juego.

Para esto, utilizaremos las funciones **OnCollision()** y **OnTrigger()**, que nos permitirán crear **mecánicas** como los **PickUps**(Objetos agarrables), **daño** por contacto, **teleports**, **botones,** etc, etc, etc. Vamos a poder empezar a robustecer nuestro juego con varias de las **mecánicas** deseadas.

Las dos funciones tienen distintos usos:

- El **Oncollision** es utilizado para objetos "**fisicos**", aquellos con los cuales queres "**chocar**", como una pared o una puerta.

- El **OnTrigger**, se utiliza para objetos que te gustaria que sean "**traspasables**". Aquellos que si los hacemos sólidos, lo único que harían es entorpecer nuestro movimiento o jugabilidad general. Mayormente esto se reserva para los **PickUps** o algunos NPC o criaturas.

Estados.

Como toda función, estas deben ser llamadas, pero esto no lo haremos nosotros/as. Sino que sucederá cuando se produzca el contacto en sí, inGame. Al usarlas vamos a ver que poseen 3 estados:

- **Enter:** Detecta el primer frame en el que hacen contacto los Colliders

```
private void OnTriggerEnter2D(Collider2D collision)
{

}
Mensaje de Unity | 0 referencias
private void OnCollisionEnter2D(Collision2D collision)
{
```

- **Stay:** Detecta si los collider se mantienen en contacto por un tiempo

```
private void OnTriggerStay2D(Collider2D collision)
{

}
Mensaje de Unity | 0 referencias
private void OnCollisionStay2D(Collision2D collision)
{
```

- **Exit:** Detecta el último frame en el que los Colliders estuvieron en contacto

```
private void OnTriggerExit2D(Collider2D collision)
{

}
Mensaje de Unity | 0 referencias
private void OnCollisionExit2D(Collision2D collision)
{
```

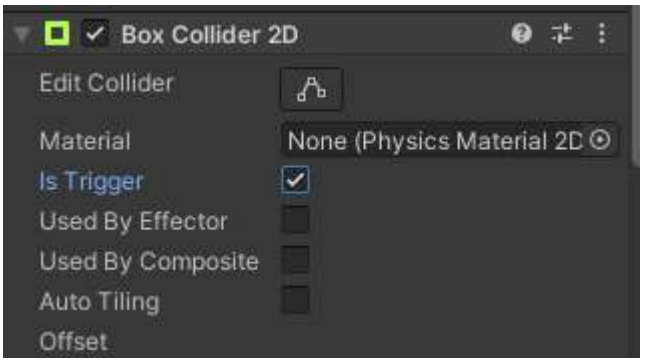
Como podrán ver son muy similares a la hora de crearlas, pero hay que tener en cuenta que el **tipo de parámetro** y el **nombre** de este, **son distintos**.

Configuración para su uso.

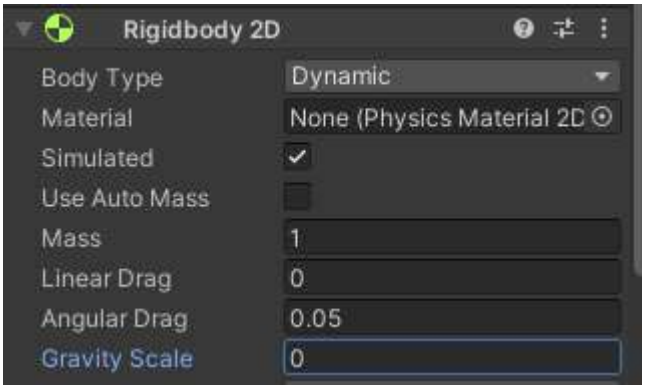
Otra diferencia, es a la hora de setearlos o realizar la configuración para que funcionen correctamente.

Para cada uno se necesita que los 2 objetos posean **Colliders2D**, pero también que al menos 1 de los 2, posea un **RigidBody2D**. Si ninguno posee un **Rigidbody2D**, entonces **no se detectara el contacto.**

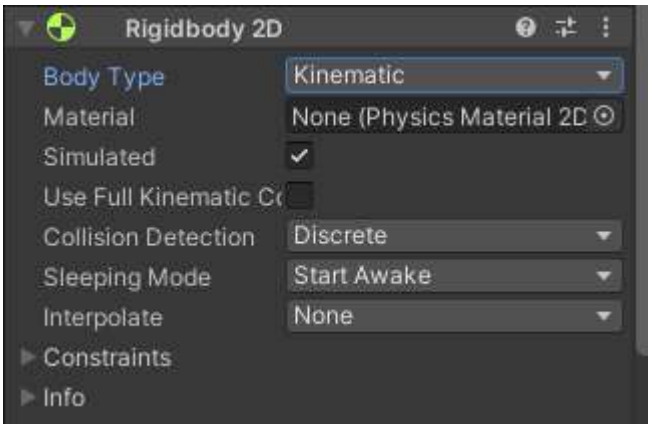
Por otro lado, el **OnTrigger()** también necesita que el checkBox “IsTrigger”, de uno de los dos objetos, esté marcado.



Tengan en cuenta que al marcarlo, el GameObject se volverá intangible, perderá su estado de “sólido” y será atravesable. Esto hará que si tiene **gravedad** caiga infinitamente.
Por eso debemos setear correctamente el **RigidBody2D** reduciendo la gravedad a 0:



O si no nos importa que le afecten las físicas externas, simplemente hacerlo “kinematic”:



Buenos Aires aprende
Agencia de Habilidades para el Futuro

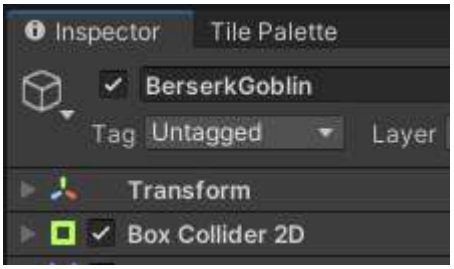
A continuación dejamos un pequeño cuadro comparativo como resumen:

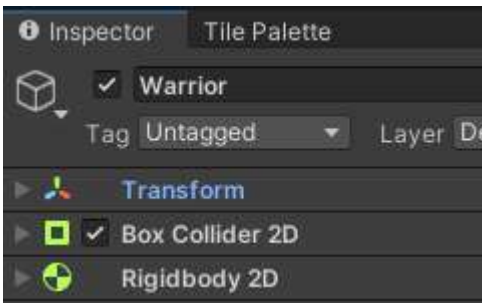
Características	OnCollisionEnter	OnTriggerEnter
Función	Sirve para detectar contacto	=
Interacción	Funciona como interacción física. El objeto "choca" con otro y le impide el paso	La interacción es virtual. Solo detecta la colisión, pero no "choca" físicamente con el objeto
Requisitos	Necesita los 2 objetos con colliders y minimo 1 con RigidBody	=
Deberes	**No debe** tener el "IsTrigger"* chequeado en el componente Collider	Minimo 1 de los objetos **DEBE** tener el "IsTrigger" chequeado en el componente Collider

Ejemplos de uso sencillos:

OnCollision()

Crearemos un simple choque con el Goblin enemigo y que, al hacerlo, nos tire un mensaje por consola. Para hacer esto, recordemos que los dos objetos deben tener colliders2d y al menos uno debe tener un rigidBody2d.





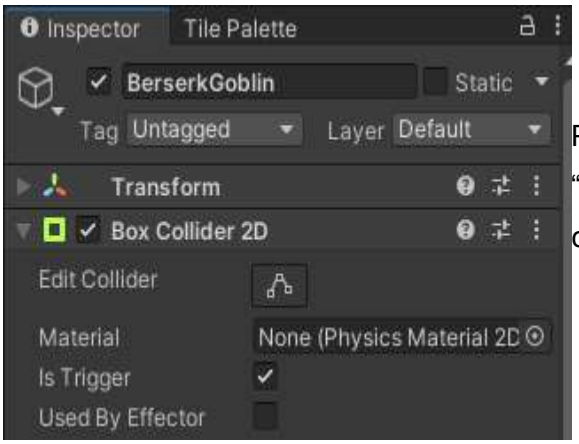
Vamos a ingresar a nuestro código de movimiento, pondremos la función OnCollision**Enter2D**() y procederemos a poner el código.

```
private void OnCollisionEnter2D(Collision2D collision)
 {
     Debug.Log("Toque algo");
 }
```

Para esta opción, asegurense que el objeto que contenga el **RigidBody2D**, lo tenga en **Dynamic.**

Buenos Aires aprende
Agencia de Habilidades para el Futuro

OnTrigger()



Para esta opción necesitaremos tildar el "isTrigger" de uno de los dos objetos. En este caso, lo haremos con el Goblin (el enemigo).

Una vez hecho esto, pasaremos a colocar el código del **IsTriggerEnter2D**

```
  private void OnTriggerEnter2D(Collider2D collision)
{
    Debug.Log("Toque algo");
}
```

Posibles dudas:

- ¿Es necesario tener siempre estas funciones en el mismo código del movimiento? **No**
- ¿En los casos el **rb2d(rigidBody2D)** debe ser Dynamic? **No, solo en el OnColission**.
- ¿La función puede estar en cualquiera de los 2 objetos? **Si**

Ejemplo de uso práctico:

PickUP

Ahora trabajaremos en lo que se le llama “**PickUps**” o **Agarrables**. Para esto crearemos un **objeto 2D** y le pondremos el **sprite** de algún objeto que nos gustaría que nuestro personaje pueda **interactuar**.

En nuestro caso le pondremos el sprite de un Hueso que está dentro del **archivo** que habíamos descargado.

A este, le pondremos un **Collider2D** y al ser un **PickUp**, es decir un objeto agarrable que no debería de evitarnos el paso, le tildaremos el box de **Is Trigger**. Debido a esto, usaremos el **OnTriggerEnter2D()** en vez del OnCollisionEnter2D().

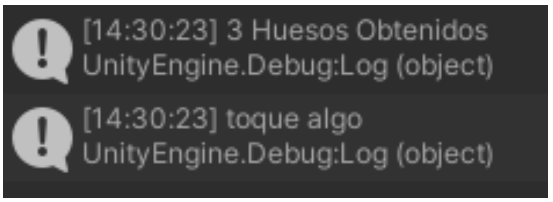
En este caso, crearemos un Script Nuevo llamado “**pickMe**” (Agarrame) que tendrá el siguiente código:

```
  private void OnTriggerEnter2D(Collider2D collision)
{
    Debug.Log("Hueso Obtenido");
}
```

Esto hará que cuando algo colisione con el Hueso, nos de este mensaje. Pero, suponiendo que queramos modificar la cantidad de huesos obtenidos. ¿Cómo podríamos hacerlo?
Por ahora, crearemos una variable llamada "**cantidad**" y la usaremos de esta manera:

```
[SerializeField]
private int cantidad = 3;
private void OnTriggerEnter2D(Collider2D collision){
    Debug.Log(cantidad + "Huesos obtenidos");
}
```

Deberá aparecer un mensaje de este tipo:



SetActive.

Notaran, que al tocar el objeto este no desaparece y podríamos obtener Huesos infinitos. Para solucionar esto hay diversas formas, pero, por ahora usaremos solo una de ellas, el "**SetActive**". Lo que hace esta función es "activar" o "desactivar" al objeto sin eliminarlo realmente de la escena, quedando como "Invisible".

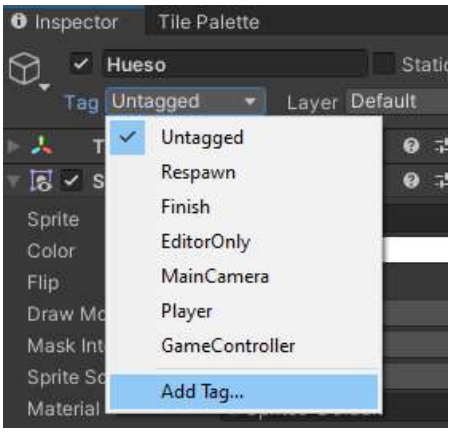
```
  [SerializeField]
private int cantidad = 3;
private void OnTriggerEnter2D(Collider2D collision){
    Debug.Log(cantidad + "Huesos obtenidos");
    gameObject.SetActive(false); }
```

Al usarlo, nos pide un bool como **Argumento,** a lo cual nosotros mandaremos **True** o **False.**

True, si queremos que aparezca o **False** si queremos que desaparezca. Tengan en cuenta que **Desactivar** el objeto es hacer que deje de funcionar, deja de existir **en la escena**. Por lo tanto, SIEMPRE tenemos que colocarlo para que sea la última función llamada, sino el resto nunca se ejecutaría y podríamos encontrarnos con varios Bugs.

Tags y Names.

Ahora bien, ¿Qué pasaría si otro objeto que no es el Player, toca al hueso? Lamentablemente, pasará lo mismo. No hemos colocado nada que permita discernir quién puede "agarrar" el objeto, por lo tanto, cualquiera podría agarrarlo; un enemigo, otro jugador, un perro o hasta un arbusto.
Para estas situaciones empezaremos a utilizar algo que se llama "**Tags**"(etiquetas) o el mismo **Name**(nombre) del objeto:



Tag: Es un método para clasificar GameObjects dependiendo de su uso. Cada objeto, si vamos al inspector, arriba de todo y debajo del nombre, tendrán la opción de agregarle un **tag**:

Al hacer click podremos desplegar una lista de los distintos tags por default y nos dará la opción de crear uno nuevo:

Name: Esta opción nos permite recurrir al **nombre** específico del **GameObject**, sirve solamente si queremos preguntar o usar ese objeto en específico. Recuerden que aunque un objeto sea copia de otro NO tendrán el mismo nombre, es como **ID**.

Ejemplo de uso práctico:

Tag.

Para empezar, crearemos un **if** con la función **CompareTag**, que nos permitirá preguntar por algún tag específico. Esto lo haremos para averiguar por el objeto al que he chocado.

```
 private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.CompareTag("Player")) {
        Debug.Log(cantidad + "Huesos obtenidos");
        gameObject.SetActive(false);
    }
}
```

Como verán, para acceder a los datos del objeto al que estoy chocando, usaremos el **parámetro** de la **función**. De ahí, colocaremos nuestra función **CompareTag**.
En este caso, pasaremos como Argumento el String “player”. Siendo un tag ya creado por Unity que le aplicaremos a nuestro personaje:

Pueden probar el juego sin colocar el **tag** y luego cambiarlo. Verán que a la primera, no sucederá nada, pero al ponerlo y realizar nuevamente el choque, llamara a las instrucciones

dentro del **if**. Recuerden que esta **función es un Enter**, por lo tanto, SOLO detecta la **colisión** en el primer frame de contacto.

Ejemplo del Name

Vamos a hacer lo mismo, pero esta vez relacionando el nombre. El código se debería ver así:

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.name == ("Warrior")) {
        Debug.Log(cantidad + "Huesos obtenidos");
        gameObject.SetActive(false);
    }
}
```

Nuevamente, usamos la variable del parámetro para acceder al **name** del objeto con el que interactúo. Si es mi “Warrior” entonces realizara las instrucciones dentro del **if**.

Accediendo al otro objeto

¿Qué pasaría ahora si yo quisiera colocar una variable de “huesosObtenidos” en mi Warrior y que vayan aumentando a medida que agarre huesos?

Primero crearemos la variable:

```
public int huesosObtenidos;
```

Recuerden hacerla “public” para poder acceder a ella más tarde.

Dentro del código del hueso pondremos lo siguiente:

```
private void OnTriggerEnter2D(Collider2D collision)
{

    if (collision.name == ("Warrior")) {
        Heroe p = collision.GetComponent<Heroe>();
        p.huesosObtenidos += cantidad;
        Debug.Log(cantidad + "Huesos obtenidos");
        Debug.Log("Tienes " + p.huesosObtenidos);
        gameObject.SetActive(false);
    }
}
```

Veamos un par de cosas.

1. Creamos una variable local del tipo de dato que queremos usar. En este caso se creó de “Heroe” siendo el nombre de la clase en donde se desarrolló el movimiento de mi personaje y donde se creó la variable “HuesosObtenidos.

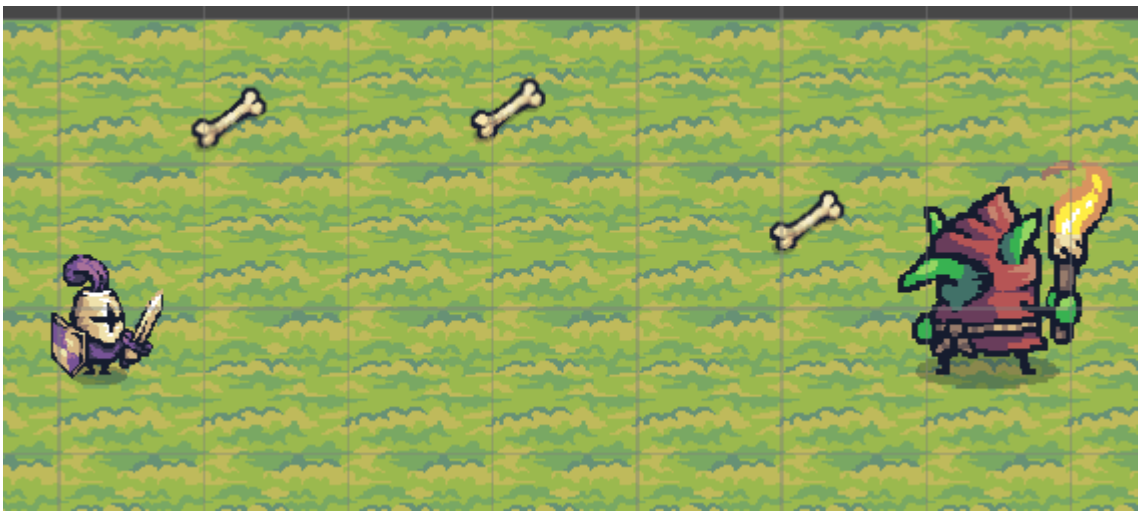
2. Utilizamos el GetComponent para asignarle el valor, siempre partiendo de nuestro Parámetro siendo “collision”. Lo que hará el GetComponent es: si el objeto que chocamos posee el Script pedido, entonces lo guardará en nuestra variable.

3. Accedemos a la variable “**huesosObtenidos**” de nuestra variable “**p**” (recuerden que hace referencia a nuestro Warrior) y le sumamos el valor de “**cantidad**”.

4. Por último aparecen los mensajes en consola y el **SetActive(False).**

De esta manera ya tenemos a nuestro PickUp, que irá sumando Huesos a nuestro personaje cuando es agarrado.

Prueben **Clonar o copiar (Ctrl+c / Ctrl+v)** los PickUps para colocar varios distribuidos en el mapa.



Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejercicios prácticos:

Utiliza las funciones Ontrigger u Oncollision para manejar las interacciones de tus gameObject creando mecánicas de “Daño”, “PickUps” o Diálogo.

1. Daño: Al tocar un objeto, debo perder vida (Mi variable vida se tiene que restar).
2. Pickup: Al tocar un objeto, este debe desaparecer y se tendrá que sumar una variable que refleje la “obtención” de ese objeto. Por ej: Si agarro una poción, me aumentara la vida o la cantidad de pociones.
3. Diálogo: Al tocar algún objeto o personaje, en la consola saldrá un mensaje simulando un diálogo. Ej: Si toco una piedra rara dirá “Esta es una piedra rara, capaz tenga algo dentro”.

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad