Colliders y Triggers

GUILD

TUTOR: C. Freddy Sacaca López

Gmail: [freddyjefemaster@gmail.com](mailto:freddyjefemaster@gmail.com)

(Se recomienda ver las diapositivas)

1. **COLLIDERS Y TRIGGERS**
   1. Collider (scripting)

Es para verificar si existe una colisión entre objetos. Unity ya creo estas funciones es decir no es necesario crearlas desde un principio para más información consultar el manual de Unity. Pero aquí les traigo algunas funciones principales y básicas.

* OnCollisionEnter

Se llama a este método cuando este tiene el componente Rigidbody y además el Game Object a colisionar tiene un collider.

La clase de la Collisión contiene la información sobre los puntos de reunión, etc de velocidad de impacto.(Ver Script “Detectar colisión”).

Esta función va a llamarse solo una ves cuando inicia la colisión

* OnCollisionStay

Este método se ejecuta en cada movimiento que se realiza en el objeto con collider es decir, si es que no seguimos moviendo en el área del collider de dicho Game Object (Ver Script “Verificar área Colisión”).

* OnCollisionExit

Este método se ejecuta cuando una colisión deja de existir, es decir cuando el objeto salió de dicha área de colisión (Ver Script “Salida Colisión”).

* 1. Triggers

Es la colisión con un objeto que no tiene volumen que no se ve en el escenario es decir entra al espacio de una escena sin crear una colisión

Este tiene la misma lógica en sus métodos que los métodos del Collision

* OnTriggerEnter

Cuando entra al Área indicada en el escenario Ej.:

using UnityEngine;

using System.Collections;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html) {

void OnTriggerExit([Collider](https://docs.unity3d.com/es/current/ScriptReference/Collider.html) other) {

// Destroy everything that leaves the trigger

Destroy(other.gameObject);

}}

* OnTriggerStay

Indica si el Game Object sigue en el área indicada, en cada movimiento Ej.:

using UnityEngine;

using System.Collections;

public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html) {  
  
 // Applies an upwards force to all rigidbodies that enter the trigger.

void OnTriggerStay([Collider](https://docs.unity3d.com/es/current/ScriptReference/Collider.html) other) {

if (other.attachedRigidbody)

other.attachedRigidbody.AddForce([Vector3.up](https://docs.unity3d.com/es/current/ScriptReference/Vector3-up.html) \* 10);

}

}

* OnTriggerExit

Indica si el Game Object ha salido del Área indicada. Ej.:

using UnityEngine;

using System.Collections;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/es/current/ScriptReference/MonoBehaviour.html) {

void OnTriggerExit([Collider](https://docs.unity3d.com/es/current/ScriptReference/Collider.html) other) {

// Destroy everything that leaves the trigger

Destroy(other.gameObject);

}

}

* 1. Rigidbody

Los **Rigidbodies** le permite a sus **GameObjects** actuar bajo el control de la física. El Rigidbody puede recibir fuerza y torque para hacer que sus objetos se muevan en una manera realista. Cualquier GameObject debe contener un Rigidbody para ser influenciado por gravedad, actué debajo fuerzas agregadas vía scripting, o interactuar con otros objetos a través del motor de física NVIDIA Physx.

## Propiedades

| **Propiedad:** | **Función:** |
| --- | --- |
| **Mass** | La masa del objeto (en Kilogramos por defecto). |
| **Drag** | Qué tanta resistencia al aire afecta el objeto cuando se mueva con fuerzas. 0 significa sin resistencia al aire, e infinidad hace que el objeto pare de moverse inmediatamente. |
| **Angular Drag** | Qué tanta resistencia al aire afecta el objeto cuando gire desde un torque. 0 significa sin ninguna resistencia al aire. Tenga en cuenta que usted no puede hacer que objeto pare de girar simplemente ajustando su Angular Drag a infinidad. |
| **Use Gravity** | Si está activado, el objeto es afectado por gravedad. |
| **Is Kinematic** | Si está activado, el objeto no será manejado por el motor de física, y puede solo ser manipulado por su **Transform**. Esto es bueno para plataformas que se muevan o si usted quiere animar un Rigidbody que tiene un **HingeJoint** adjunto. |
| **Interpolate** | Intente una de las opciones si usted está viendo que los movimientos de su Rigidbody estén sacudidos. |
| - **None** | Ninguna interpolación es aplicada. |
| - **Interpolate** | El transform es suavizado básandose en el transform del cuadro previo. |
| - **Extrapolate** | El transform es suavizado básandose en el transform estimado para el siguiente cuadro. |
| **Collision Detection** | Utilizado para prevenir que objetos que se estén moviendo rápido pasen a través de otros objetos sin detectar colisiones. |
| - **Discrete** | Utiliza la detección de colisiones discreta (Discreet collision detection) contra todos los otros colliders en la escena. Otros colliders van a utilizar la detección de colisiones discreta cuando pruebe por colisiones contra ella. Utilizado para colisiones normales (Este es el valor por defecto). |
| - **Continuous** | Utiliza Discrete collision detection contra dynamic colliders (con un rigidbody) y continuous collision detection contra static MeshColliders (sin un rigidbody). Rigidbodies configurados a Continuous Dynamic utilizarán continuous collision detection cuando pruebe por colisiones contra este rigidbody. Otros rigidbodies utilizarán Discreet Collision detection. Utilizado para objetos los cuales el Continuous Dynamic detection necesita colisionar con. (Esto tiene un gran impacto en el rendimiento de física, déjelo configurado a Discrete, si usted no quiere problemas con colisiones con objetos rápidos) |
| - **Continuous Dynamic** | Utiliza la detección de colisión continua contra objetos configurados Continuous y Continuous Dynamic Collision. También utilizará la detección de colisión continua contra MeshColliders estáticos (sin un rigidbody). Para los demás otros colliders éste utiliza discreete collision detection. Utilizado para objetos que se estén moviendo rápidos. |
| **Constraints** | Restricciones en el movimiento del Rigidbody :- |
| - **Freeze Position** | Detiene al Rigidbody que se mueva en los ejes X, Y y Z del mundo selectivamente. |
| - **Freeze Rotation** | Detiene al Rigidbody en girar alrededor del eje X, Y y Z del mundo selectivamente. |

## Detalles

Los Rigidbodies le permite a sus GameObjects actuar bajo el control del motor de física. Esto abre la puerta a colisiones realistas, una variedad de tipos de articulaciones, y otros comportamientos buenos. Manipular sus Gameobjects al agregar fuerza a un Rigidbody crea una sensación diferente y de aspecto que en vez de ajustar el Transform **Component** directamente. Generalmente, usted no debería manipular el Rigidbody y el Transform del mismo GameObject - solamente uno o el otro.

La diferencia más grande entre manipular el Transform vs el Rigidbody es el uso de fuerzas. Los Rigidbodies pueden recibir fuerzas y torque, pero el Transforms no puede. Los Transform pueden ser trasladados y girados, pero esto no es lo mismo que utilizar física. Usted notará la diferencia clave cuando lo intente usted mismo. Agregando fuerzas/torque al Rigidbody puede cambiar la posición y la rotación del componente Transform del objeto. Esta es la razón por la cual usted debería solamente utilizar una o la otra. Cambiar el Transform mientras utilice física puede causar problemas con colisiones y otros cálculos.

Los Rigidbodies deben ser explícitamente agregados a su GameObject antes de que sean afectados por el motor de física. Usted puede agregar un Rigidbody a su objeto seleccionado desde **Components->Physics->Rigidbody** en la barra de menú. Ahora su objeto está listo con física; este caerá bajo la gravedad y puede recibir fuerzas via scripting, pero usted pueda que necesite agregar un **Collider** o Joint para obtener que se comporte como usted quiere.

### Crianza

Cuando un objeto está debajo del control de física, este se mueve semiindependientemente de la manera en que el transform de sus padres de mueva. Si usted mueve cualquier padre, estos jalarán el hijo Rigidbody a lo largo con ellos. Sin embargo, el Rigidbodies también se caerán gracias a la gravedad y la reacción a eventos de colisión.

### Scripting

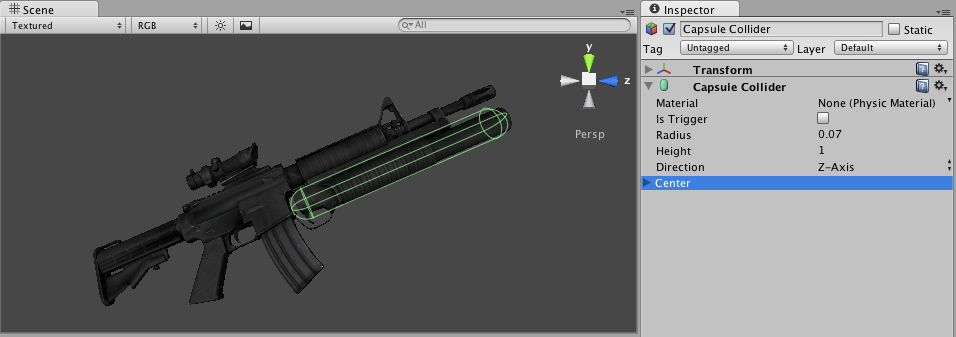
Para controlar sus Rigidbodies, usted va a principalmente necesitar utilizar scripts para agregar fuerzas o torque. Usted hace esto llamando [**AddForce()**](https://docs.unity3d.com/es/current/ScriptReference/Rigidbody.AddForce.html) y [**AddTorque()**](https://docs.unity3d.com/es/current/ScriptReference/Rigidbody.AddTorque.html)en el rigidbody del objeto. Recuerde que usted no debería directamente alterar el transform del objeto cuando usted está utilizando física.

### Animación

Para algunas situaciones, principalmente creando efectos de ragdoll, es necesario cambiar el control del objeto entre animaciones y física. Para este propósito los Rigidbodies pueden ser marcados [**isKinematic**](https://docs.unity3d.com/es/current/ScriptReference/Rigidbody-isKinematic.html). Mientras que el Rigidbody esté marcado **isKinematic**, este no será afectado por colisiones, fuerzas, o cualquier otra parte del sistema de física. Esto significa que usted va a necesitar controlar los objetos manipulando el componente [Transform](https://docs.unity3d.com/es/current/Manual/class-Transform.html) directamente. Los Rigidbodies Kinematic van a afectar otros objetos, pero ellos mismos no serán afectados por física. Por ejemplo, las Articulaciones (Joints) que estén adjuntos a objetos Kinematic van a restringir otros Rigidbodies adjuntos a ellos y Kinematic Rigidbodies van a afectar otros Rigidbodies a través de colisiones.

### Colliders

Los Colliders son otro tipo de componente que debe ser agregado a lo largo del Rigidbody con el fin de permitir que otras colisiones ocurran. Si dos Rigidbodies se chocan entre ellas, el motor de física no va a calcular una colisión al menos que ambos objetos también tengan un Collider adjunto. Los Rigidbodies sin Colliders simplemente pasarán a través de cada uno durante otras simulaciones de física.

Los Colliders definen los limites físicos de un Rigidbody

Agregue un Collider con el menú **Component->Physics**. Mire la página de referencia del Componente de cualquier Collider individual para información más especifica:

* [Box Collider](https://docs.unity3d.com/es/current/Manual/class-BoxCollider.html) - una figura primitiva de un cubo
* [Sphere Collider](https://docs.unity3d.com/es/current/Manual/class-SphereCollider.html) - una figura primitiva de una esfera
* [Capsule Collider](https://docs.unity3d.com/es/current/Manual/class-CapsuleCollider.html) - una figura primitiva de una capsula
* [Mesh Collider](https://docs.unity3d.com/es/current/Manual/class-MeshCollider.html) - crea un collider del mesh del objeto, no puede colisionar con otro Mesh Collider
* [Wheel Collider](https://docs.unity3d.com/es/current/Manual/class-WheelCollider.html) - específicamente para crear carros o otros vehículos en movimiento
* [Terrain Collider](https://docs.unity3d.com/es/current/Manual/class-TerrainCollider.html) - Manejan colisiones con el sistema de terrain (terrenos) de Unity

### Compound Colliders (Colliders Compuestos)

Los Compound Colliders (colliders compuestos) son combinaciones de colliders primitivos, colectivamente actuando como un collider sencillo. Estos son prácticos cuando usted tiene un mesh complejo para utilizar en colisiones pero no pueden utilizar un Mesh Collider. Para crear un Compound Collider, cree objetos hijo de su objeto colisionado, luego agregue un componente collider a cada objeto hijo. Esto le permite a usted posicionar, girar, y escalar cada Collider de manera fácil e independiente uno de otro. Usted puede construir su compound collider de un número de colliders primitivos y/o convex mesh colliders.

[Una configuración del mundo real de un Compound Collider](https://docs.unity3d.com/es/current/uploads/Main/CompoundCollider.png)

En la foto de arriba, el GameObject Gun Model tiene un Rigidbody adjunto, y múltiplos Colliders primitivos como GameObjects hijos. Cuando el padre Rigidbody es movido alrededor por fuerzas, los colliders hijos se mueve con él. Los Colliders primitivos van a colisionar con el entorno del Mesh Collider, y el Rigidbody del padre va a alterar la manera en que se mueva basándose en las fuerzas siendo aplicadas a él y cómo sus colliders hijos interactuar con otros Colliders en la escena.

Cuando los Mesh Colliders no pueden normalmente colisionar con cada uno. Si un Mesh Collider es marcado como **Convex**, entonces puede colisionar con otro Mesh Collider. La solución típica es utilizar Colliders primitivos para cualquier objeto que se mueva, y los Mesh Colliders para objetos de fondo estáticos.

* 1. **Ejemplos**

Los ejemplos se les proporcionaran en Código

* 1. **Practica**:

**Recoger Coins**

* Deberá recoger algo así como monedas y estas deben de desaparecer cuando colisionan con el jugador
* Bajo la Misma Lógica crear Objetos que cuando mi jugador se acerque a dicha Área estos desaparezcan (Usar triggers)