DESARROLLO DE VIDEOJUEGOS Y REALIDAD VIRTUAL CON UNITY 3D

Tema 6: Físicas

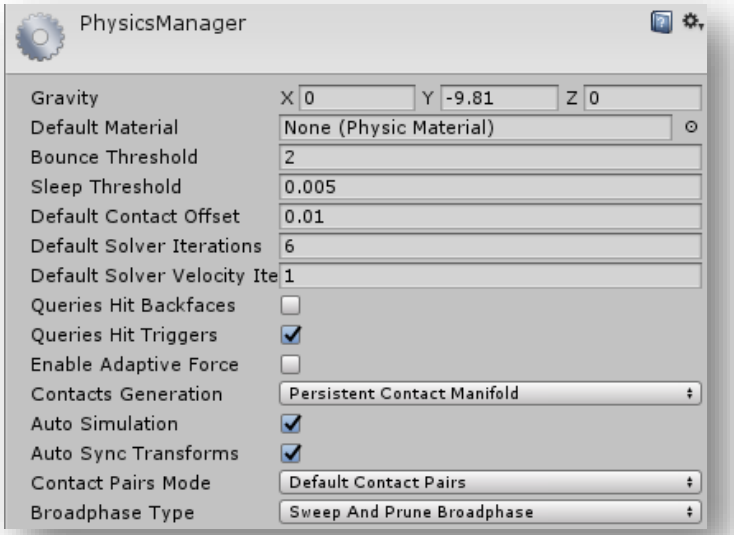
TEMA 6. FÍSICAS

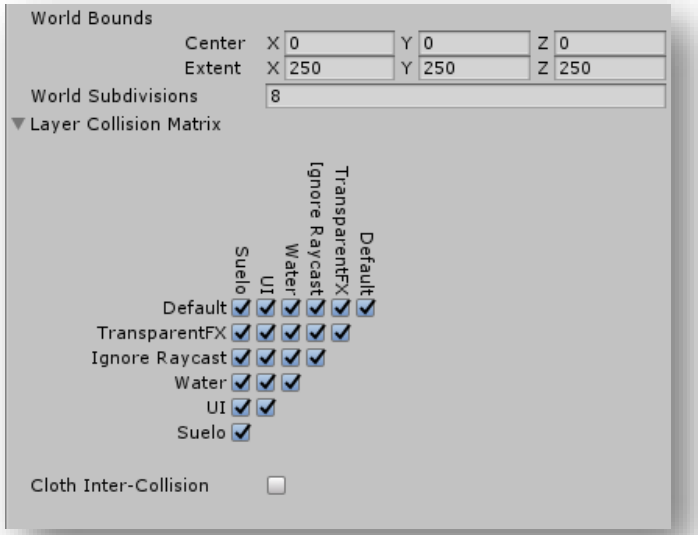
- Introducción al motor físico
  - El problema de la física en los videojuegos
  - PPU (Physical Procesing Units)

  - Motores físicos incluidos en Unity de forma nativa:
    - NVIDIA PhysX (antes AGEIA PhysX) https://www.nvidia.es/object/nvidia-physx-es.html
    - Box2D https://box2d.org/

TEMA 6. FÍSICAS

- Introducción al motor físico
  - La configuración de las propiedades físicas en Unity se pueden modificar a través de la opción Edit->Project Settings->Physics y Physics 2D.





https://docs.unity3d.com/Manual/class-PhysicsManager.html
https://docs.unity3d.com/es/current/Manual/class-PhysicsManager.html

TEMA 6. FÍSICAS

- Cuerpos rígidos y uso de fuerzas
  - Rigidbody
    - Componente necesario para vincular un GameObject con PhysX.
    - Propiedades:
      - Mass: la masa expresada en kilogramos. Determina como afectan las fuerzas aplicadas al objeto y como afecta el impacto de dicho objetos en otros objetos del escenario.
      - Drag: rozamiento contra el aire.
        - Es una fuerza en sentido contrario al movimiento del objeto y dependerá de su velocidad.
        - No afecta masa ni forma.
        - Ejemplo: caída libre pluma vs piedra.
      - Angular Drag: rozamiento contra el aire sobre la rotación del objeto.
      - Use Gravity: determina si debe afectar la gravedad o no.
        - La gravedad se puede modificar en la configuración del proyecto.

TEMA 6. FÍSICAS

- Cuerpos rígidos y uso de fuerzas
  - Rigidbody
    - Propiedades:
      - Is Kinematic. Determina si el objeto es dinámico (Is Kinematic desactivado) o cinemático (Is Kinematic activado).
        - Un objeto cinemático es controlado desde un script o animaciones.
        - Si el objeto es cinemático ignora cualquier fuerza que se le asigne, ya sea a través de un script o del contacto con otro objeto.
      - Interpolate. Permite sincronizar el movimiento del objeto (FixedUpdate) con el movimiento de la cámara (Update). Soluciona problemas de temblor a nivel gráfico *(jitter)*.
        - None: sin sincronización.
        - Interpolate: Mueve el objeto con un frame de retraso sobre su posición física para sincronizar la cámara.
        - Extrapolate. En función de la velocidad del objeto, predice donde estará en el momento en el que lo dibuje la cámara.

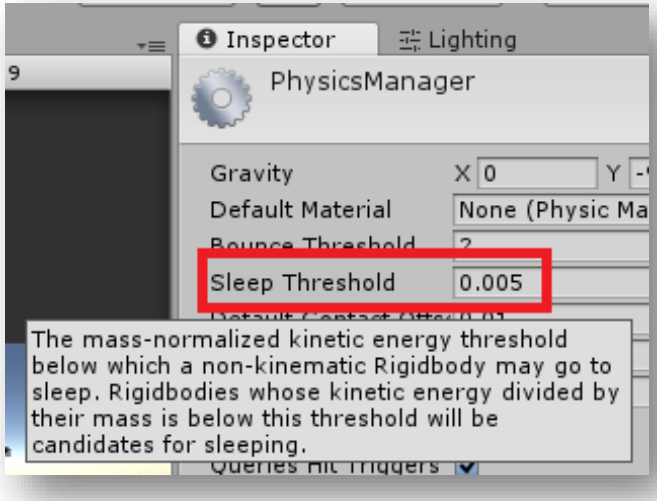
TEMA 6. FÍSICAS

- Cuerpos rígidos y uso de fuerzas
  - Rigidbody
    - Propiedades:
      - Collision Detection. Determina la precisión de la detección de colisiones.
        - General (por defecto)
        - Continous Dynamic (objetos que se mueven rápidamente)
        - Continous (objetos contra los que colisionan objetos "Continous Dynamic".
        - Constraints: restricciones de movimiento y rotación.

TEMA 6. FÍSICAS

- Cuerpos rígidos y uso de fuerzas
  - Consideraciones
    - Una vez un objeto tiene Rigidbody, deberemos utilizar preferentemente los métodos proporcionados por el mismo para realizar algún tipo de movimiento, ya que provocará movimientos más realistas**. No se debe utilizar directamente el componente Transform** ya que entra en conflicto con el motor de física.
    - **Emparentar** varios objetos con RigidBody puede provocar errores en los cálculos físicos, al afectar la física de los distintos objetos sobre el resto.
    - La **escala** del objeto afectará a cómo se aplican las fuerzas: una escala correcta producirá un efecto más realista.
    - Para aplicar **animaciones** a objetos con Rigidbody se debe desactivar el modo kinemático. Si se desea recuperar el control de Transform del objeto mediante código, se debe activar cuando se necesite.

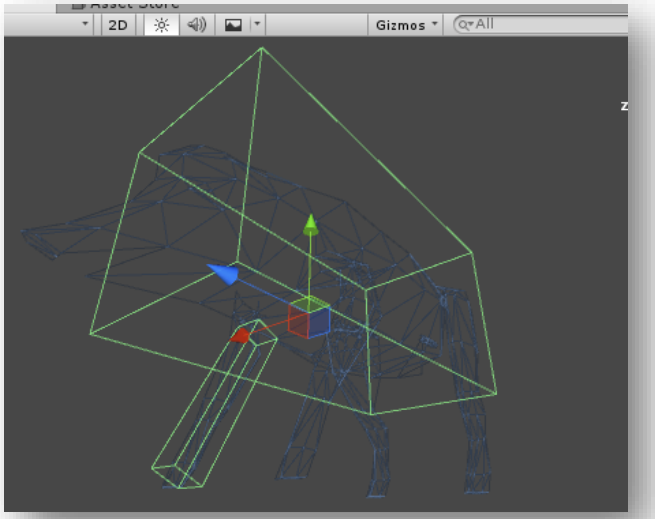
TEMA 6. FÍSICAS



- Cuerpos rígidos y uso de fuerzas
  - Estados del componente Rigidbody:
    - Sleep y Wakeup.
      - Sleep: No participa de los cálculos físicos.
      - El motor físico pone los objetos con el componente Rigidbody en estado Sleep cuando éstos no se mueven.
      - El parámetro que define cuando un objeto está "quieto" para el motor físico es Sleep Threshold (Edit -> Project Settings -> Physics).
      - Ante un cambio en las colisiones (esta se produce o deja de producirse) Unity dormirá o desperará el rigidbody de manera automática.
      - Se puede invocar a los métodos Sleep y WakeUp del componente Rigidbody desde los scripts.

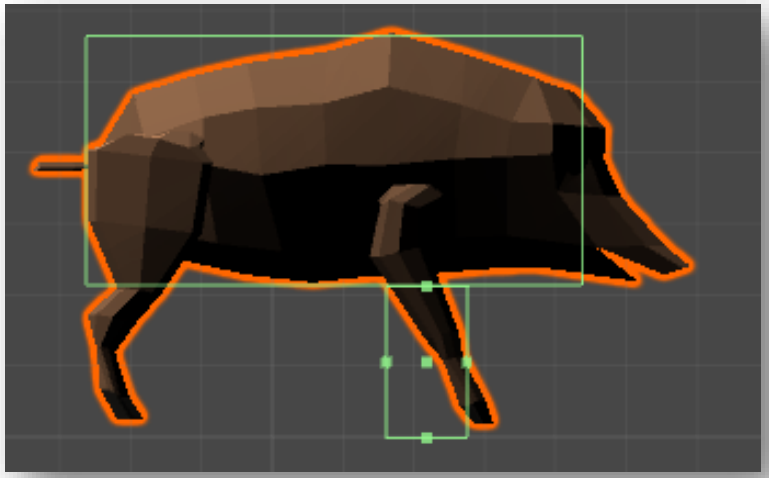
TEMA 6. FÍSICAS

- Detección de colisiones y triggers
  - Motor de dibujo vs. Motor de físicas
    - MeshRenderer (geometría visual) vs Collider (geometría física).
  - Collider:
    - Definición.
    - Unión de varios volúmenes de colisión.
      - **Compound Colliders**: para optimizar formas visuales complejas se pueden combinar volúmenes de colisión simples.



TEMA 6. FÍSICAS

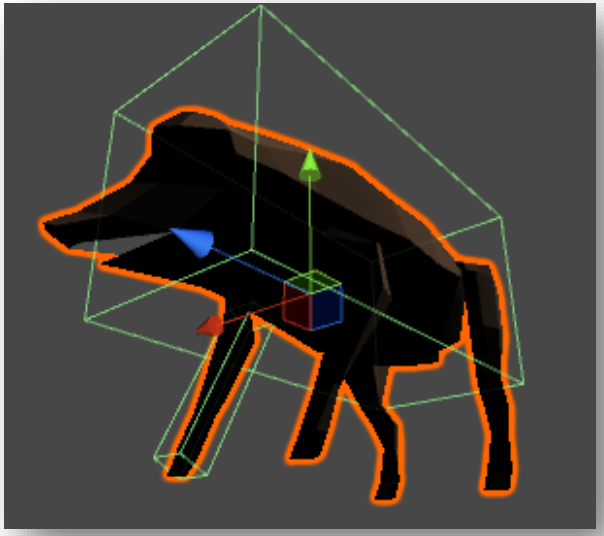
- Detección de colisiones y triggers
  - Compound Colliders. Técnicas:
    - Varios Colliders en un mismo GameObject. Sólo modelos simples.
      - Dificultad: ajustar los colliders a la geometría.
      - Dificultad: obtener las referencias desde los scripts (debe utilizarse GetComponents<Collider>)



TEMA 6. FÍSICAS

- Detección de colisiones y triggers
  - Compound Colliders. Técnicas:
    - Mediante jerarquía de GameObjects.
      - El objeto raíz tiene el componente Rigidbody.
      - Los hijos tienen su propio collider.
      - Los métodos de detección de Collider y Trigger sólo se ejecutan en el objeto raíz (el que tiene el Rigidbody) ya que todos los colliders se "agrupan" desde el punto de vista del punto de detección.

Si el objeto raíz no tiene Rigibody los colliders de los objetos hijos actuarán



TEMA 6. FÍSICAS

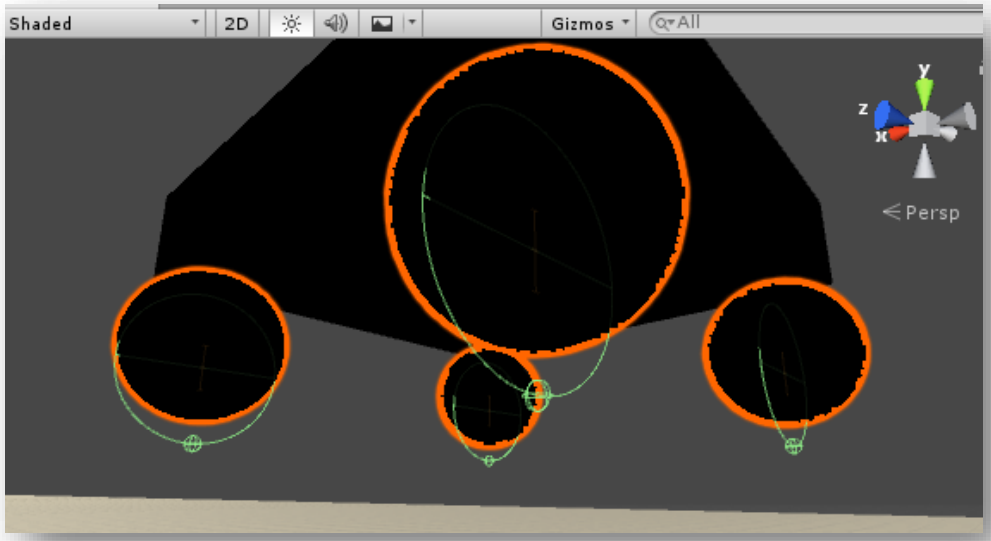
- Detección de colisiones y triggers
  - Propiedades de los Colliders:
    - Edit Collider. Permite modificar las dimensiones y la posición del collider.
    - Is Trigger. Determina si el objeto colisiona o sólo detecta la intersección.
    - Material. Permite asignar un **Physic Material**.
    - Center. Centro del collider.

TEMA 6. FÍSICAS

- Detección de colisiones y triggers
  - Tipos de collider:
    - Todos los tipos de collider hederan de la clase Collider.
    - Box collider.
    - Shere collider. Es el más eficiente de todos desde la perspectiva del cálculo matemático. No se puede deformar.
    - Capsule collider.
    - **Wheel collider**.
    - Terrain collider. Se genera y configura automáticamente con el editor de terrenos.
    - **Mesh collider**.

TEMA 6. FÍSICAS

- Detección de colisiones y triggers
  - Tipos de collider. Wheel collider.
    - Collider específico para vehículos de suelo.
    - Incluye detección de colisión, físicas de ruedas y modelo de deslizamiento.



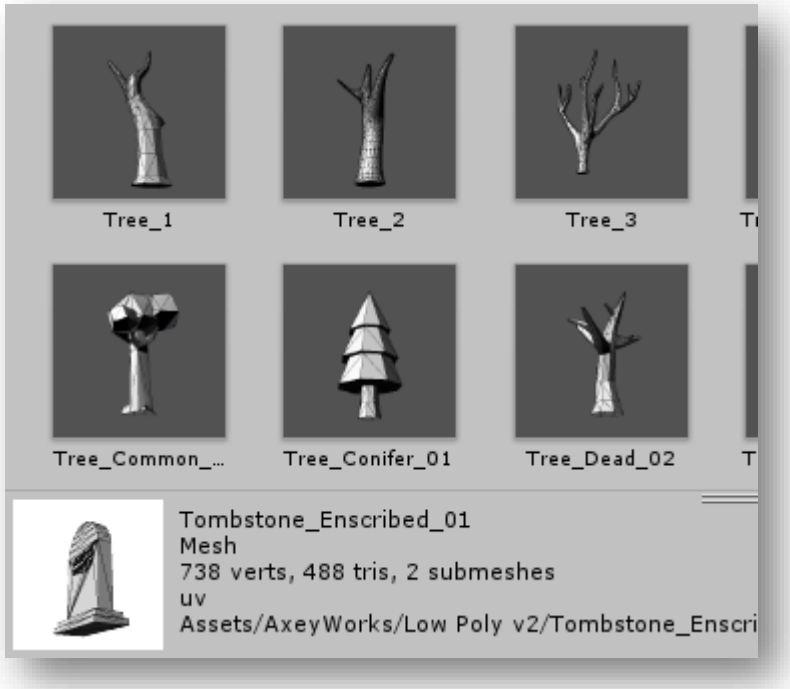
https://docs.unity3d.com/es/current/Manual/class-WheelCollider.html

TEMA 6. FÍSICAS

- Detección de colisiones y triggers
  - Tipos de collider. Mesh collider
    - Se ajusta al modelo a través de una malla tridimensional.
    - Permite seleccionar una malla física a través de la opción "Mesh".
    - **Muy preciso** pero m**uy costoso**: sólo se debe utilizar en objetos estáticos o en objetos muy importantes pero se recomienda utilizar una malla más sencilla que la malla gráfica.
    - Si la malla <= 255 triángulos y el volumen es convexo, se puede activar la propiedad **Convex. Entonces.**
      - **Puede asociarse a un rigidbody**
      - **Puede configurarse como trigger**
      - **Puede moverse por la escena**
    - En caso contrario
      - **No puede asociarse a un rigidbody**
      - **No puede configurarse como trigger**
      - **No puede moverse por la escena**

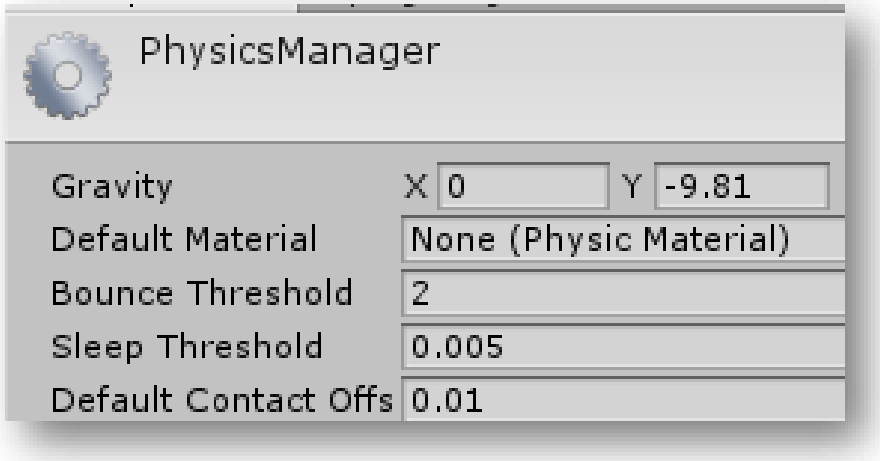
TEMA 6. FÍSICAS

- Detección de colisiones y triggers
  - Tipos de collider. Mesh collider
    - Permite seleccionar una malla física a través de la opción "Mesh". En dicha selección se indica el número de vértices y triángulos de la malla.



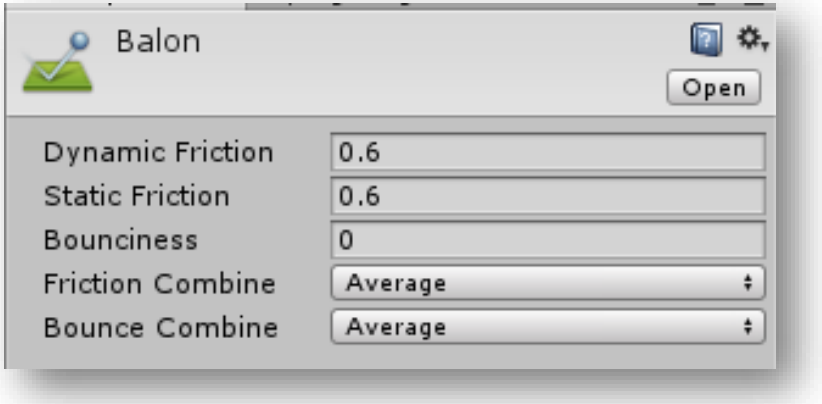
TEMA 6. FÍSICAS

- Detección de colisiones y triggers
  - Physic Material.
    - El material físico de los colliders permite especificar cómo se comporta físicamente el volumen de la colisión, principalmente en lo que al rozamiento y al rebote se refiere.
    - Se crean como un asset del proyecto con la opción **Create -> Physic Material**.
    - Los materiales físicos se asignan como una propiedad a los colliders.
    - Se puede aplicar un material físico a todos los objetos físicos configurando la propiedad **Default Material** a través de la propiedad **Edit -> Project Settings.**



TEMA 6. FÍSICAS

- Detección de colisiones y triggers
  - Physic Material.
    - Dynamic fricción: el rozamiento que experimenta un objeto físico al moverse contra otro objeto físico.
    - Static friction: el rozamiento que experimenta un objeto físico en estado de reposo al comenzar a moverse.
    - Bounciness: rebote, entre 0 (no rebota) y 1 (rebota y no pierde energía).
    - Friction Combine: cómo se combinan las fricciones de dos objetos físicos.
    - Bounce Combine: cómo se combinan los rebotes de dos objetos físicos.



TEMA 6. FÍSICAS

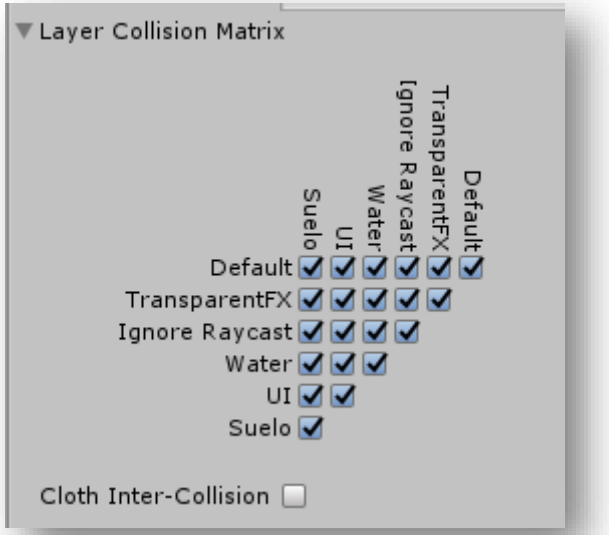
- Detección de colisiones y triggers: colliders
  - Detecta la colisión entre objetos físicos.
  - Eventos de tipos Collision:
    - Deben ser objetos dinámicos (isKinematic desactivado)
    - La propiedad isTrigger debe estar desactivada
    - Se disparan en los scripts de los objetos que colisionan que tienen el componente Rigidbody
    - Los eventos de colisión reciben un objeto de tipo Collision que dispone de información sobre el impacto (gameobject, puntos de impacto, velocidades relativas e impulso aplicado).
    - Eventos:
      - OnCollisionEnter
      - OnCollisionExit
      - OnCollisionStay. Permanencia dentro del área de contacto.
      - OnCollisionEnter2D
      - OnCollisionExit2D
      - OnCollisionStay2D

TEMA 6. FÍSICAS

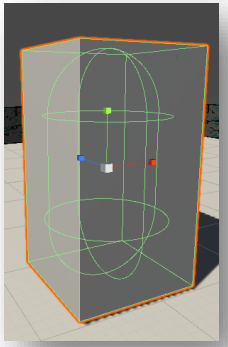
- Detección de colisiones y triggers: triggers.
  - En collider de tipo trigger los objetos se atraviesan en lugar de colisionar.
  - La propiedad isTrigger debe estar activada
  - Los eventos (de ambos objetos) se invocan cuando un objeto con Rigibody (sea dinámico o cinemático) a traviesa un trigger de otro objeto.
  - Los eventos de tipo trigger reciben un objeto de tipo Collider que dispone una referencia al gameobject sobre el que se ha disparado el evento.
  - Eventos de tipo *trigger*:
    - OnTriggerEnter
    - OnTriggerExit
    - OnTriggerStay. Permanencia dentro del área de contacto.
    - OnTriggerEnter2D
    - OnTriggerExit2D
    - OnTriggerStay2D

TEMA 6. FÍSICAS

- Detección de colisiones y triggers: matriz de colisiones.
  - Permite determinar qué objetos pueden colisionar o no con otros objetos.
  - La relación se establece a través de las capas (Layers) y se configura para todo el proyecto en la opción de configuración de propiedades físicas Edit->Project Settings->Physics.



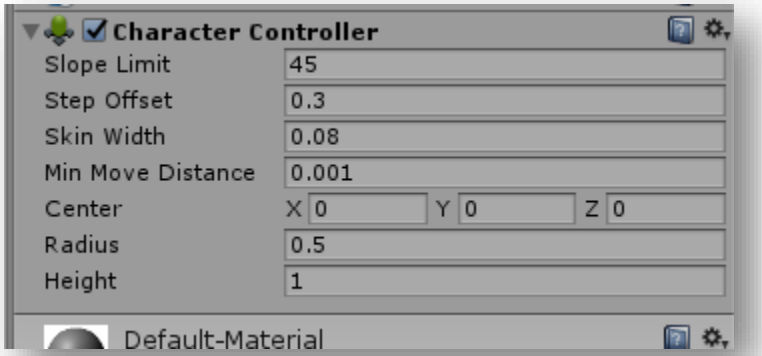
TEMA 6. FÍSICAS



- Físicas del controlador de personajes
  - El componente CharacterController es un volumen de colisión con forma de capsula.
  - No debe incluir ningún otro collider.
  - No requiere Rigidbody, pero si se incluye debe ser cinemático.
  - La física no es realista y no aplica fuerzas a otros objetos, aunque sí respeta las colisiones (no traspasa objetos).
  - Métodos de colisión:
    - **No genera** eventos OnCollisionX (sólo en algunas condiciones, pero es preferible no utilizarlo).
    - Genera un evento OnControllerColliderHit.
      - Sólo si el personaje está en movimiento.
      - Recibe un objeto ControllerColliderHit con toda la información referente a con qué objeto ha impactado y dónde.
    - Sí genera eventos OnTriggerX.

TEMA 6. FÍSICAS

- Físicas del controlador de personajes
  - Propiedades.
    - Slope Limit: inclinación máxima (en grados) de la pendiente que es capaz de subir el personaje .
    - Step Offset: altura máxima del escalón que es capaz de subir.
    - Skin Width: margen alrededor del volumen de colisión que no detecta colisiones (sobre el 10% del radio).
    - Min Move Distance: rango a partir del cual comienza a moverse el personaje.



TEMA 6. FÍSICAS

- Físicas del controlador de personajes
  - Movimiento.
    - Mediante los métodos SimpleMove y Move.
    - Dentro del método Update, sólo a uno de los dos métodos y en todos los frames.
    - SimpleMove.
      - Aplica un vector de velocidad sólo sobre los ejes x y z.
      - Simula gravedad (mientras se mueve)
      - Devuelve un bool que indica si el gameobject está en el suelo
    - Move.
      - Aplica un vector de desplazamiento sobre todos los ejes (permite modificar la altura).
      - No simula gravedad.
      - Devuelve un objeto de tipo CollisionFlags que indica si está tocando otros volúmenes de colisión dando información de sobre qué cara lo está haciendo.

TEMA 6. FÍSICAS

- Físicas del controlador de personajes
  - Movimiento. Ejemplo del método SimpleMove:

```
public class Personaje : MonoBehaviour {
     CharacterController cc;
     public float speed = 10f;
     Vector 3velocidad;
     void Start () {
             cc = GetComponent<CharacterController>();
     }
     void Update () {
             velocidad = transform.forward * speed *
     Input.GetAxis("Vertical");
             bool enSuelo = cc.SimpleMove(velocidad);
             print("EnSuelo:" + enSuelo);
     }
}
```

TEMA 6. FÍSICAS

- Físicas del controlador de personajes
  - Movimiento. Ejemplo del método Move (sobre un objeto volador):

```
[RequireComponent(typeof(CharacterController))]
public class SpaceShip : MonoBehaviour
{
    Camera camera;
    CharacterController cc;
    public float speed = 10f;
    public float angularSpeed = 1;
    Vector3 posicion;
    void Start()
    {
        cc = GetComponent<CharacterController>();
        camera = Camera.main;
    }
    void Update()
    {
        //Rotacion
        transform.Rotate(new Vector3(
            Input.GetAxis("Vertical") * angularSpeed * -1,
            Input.GetAxis("Horizontal") * angularSpeed * -1,
            Input.GetAxis("Horizontal") * angularSpeed) * - 1);
        //Movimiento
        cc.Move(transform.forward * speed * Time.deltaTime);
    }
}
```

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones:

Objetos dinámicos

Objetos cinemáticos

Objetos estáticos

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos dinámicos
  - Tiene un Rigidbody con la propiedad isKinematic desactivada.
  - Se mueven mediante el motor físico.
  - Experimentan cambios en su movimiento cuando colisionan contra otros objetos.
  - Deben ser controlados mediante el uso de fuerzas y torsiones aplicados al componente Rigidbody e **invocados desde el método FixedUpdate**:
    - AddForce
    - AddTorque

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos dinámicos
  - AddForce: aplica una fuerza con una determinada dirección y magnitud:
    - public void **AddForce**(Vector3 **force**, ForceMode **mode** = ForceMode.Force)
  - AddTorque: aplica un par de fuerzas que produce una rotación del objeto:
    - public void **AddTorque**(Vector3 **torque**, ForceMode **mode** = ForceMode.Force);

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos dinámicos
  - Parámetros de los métodos AddForce y AddTorque
    - El primer parámetro define el vector de dirección o de rotación y la intensidad.
    - El segundo parámetro define cómo aplicar la fuerza:
      - ForceMode.Force. Valor por defecto. Aplica la fórmula *Fuerza = masa * aceleración.* Permite incrementar la velocidad poco a poco. Diseñado para **aplicarse frame a frame**. Aceleración de un vehículo.
      - ForceMode.Acceleration. Aplica directamente una aceleración **sin considerar la masa** del objeto. Diseñado para **aplicarse frame a frame**. Efecto de la gravedad.
      - ForceMode.Impulse. Aplica la fórmula *Impulso = masa * incremento velocidad.* Incrementa la velocidad en una magnitud inversamente proporcional a su masa. Diseñado para aplicar un incremento de velocidad instantáneo sobre el objeto. Diseñado para aplicarse **en un único frame**. Golpeo de una pelota con una raqueta.
      - ForceMode.Velocity. Aplica directamente un incremento en la velocidad del objeto, **ignorando su masa**. Diseñado para ser aplicado en **un único frame**.

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos dinámicos
  - Otros métodos:
    - AddRelativeForce. Alternativa al método AddForce en coordenadas locales.
    - AddRelativeTorque. Alternativa al método AddTorque en coordenadas locales.
    - AddForceAtPosition. Aplicar una fuerza en una posición determinada.
    - AddExplosionForce. Aplicar una fuerza de explosión que afecte a un radio de distancia.
      - public void **AddExplosionForce** (float **explosionForce** , Vector3 **explosionPosition** , float **explosionRadius** , float **upwardsModifier** = 0.0f, ForceMode **mode** = ForceMode.Force));

```
GameObject[] explotables = GameObject.FindGameObjectsWithTag("Explotable");
foreach (var item in explotables)
{
  item.GetComponent<Rigidbody>().AddExplosionForce(fuerza, transform.position, radio, impulso);
}
```

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos dinámicos
  - Otras alternativas de interacción física:
    - Modificando directamente los atributos velocity y angularVelocity
    - Usar puntualmente en un único frame.
  - Otras consideraciones:
    - En objeto físicos dinámicos no es recomendable modificar las propiedades position y rotation del componente Transform, ya que provocarán errores en la simulación física.
    - Si se necesita controlar por física la posición y por programa la rotación, deberemos hacer uso de las *constraints* del componente RigidBody.

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos cinemáticos
  - Un objeto cinemático es un objeto cuyo componente Rigidbody tiene activada la propiedad **isKinematic**.
  - No les afecta el motor físico.
  - Son controlados a través de scripts: mediante el componente **Transform** o mediante el método **MovePosition** del componente **Rigidbody**.
  - Los objetos cinemáticos pueden colisionar con objetos dinámicos, pero no con otros objetos cinemáticos.
  - Los objetos cinemáticos no atienden los métodos OnCollisionX.
  - Los objetos cinemáticos sí atienden los métodos OnTriggerX.
  - La colisión entre cinemático y dinámico afecta sólo al objeto dinámico.
  - En caso de colisión entre cinemático y cinemático o entre cinemático y estático provoca que los objetos se atraviesen.

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos cinemáticos
  - Movimiento de objetos cinemáticos.
    - Se debe realizar a través de scripts o de animaciones.
    - Las operaciones físicas no afectan, por lo que se debe utilizar el componente Transform o el método MovePosition del componente Rigidbody.

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos cinemáticos
  - Ejemplo: gameobject cinemático que persigue un target y rota.

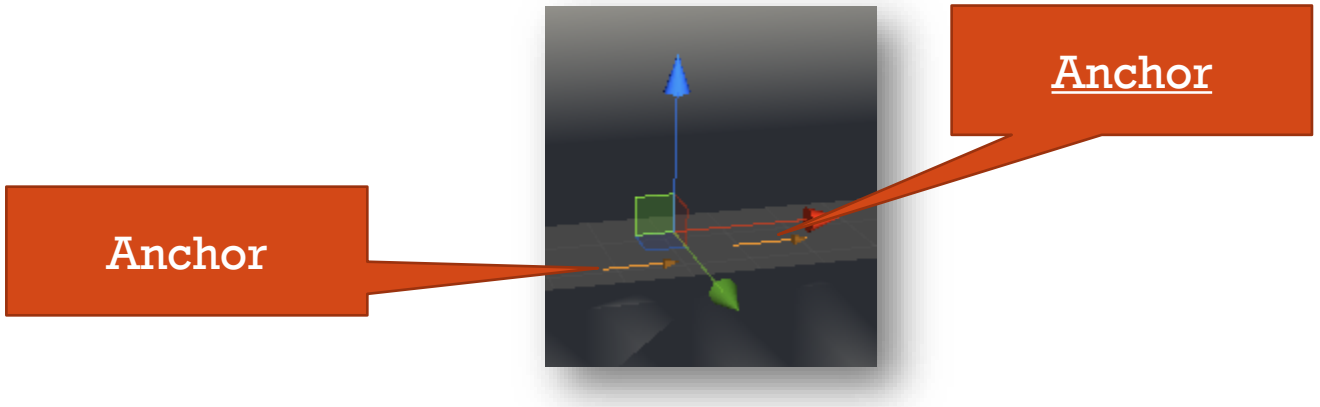
```
public class Bola : MonoBehaviour {
    public Transform target;
    private Rigidbody miRB;
    private float rango = 0.1f;//Evita que tiemble el objeto
    private void Start()
    {
        miRB = GetComponent<Rigidbody>();
    }

    private void FixedUpdate()
    {
        Vector3 distancia = (target.position - transform.position);
        if (distancia.magnitude>rango)
        {
            Vector3 direccion = distancia.normalized;
            Vector3 deltaPosition = direccion * Time.deltaTime;
            miRB.MovePosition(miRB.position + deltaPosition);
        }
        Quaternion deltaRotation = Quaternion.Euler(new Vector3(0,50,0)
* Time.deltaTime);
        miRB.MoveRotation(miRB.rotation * deltaRotation);

    }

}
```

TEMA 6. FÍSICAS

- Tipos de objetos físicos e interacciones: Objetos estáticos
  - No dispone ni él, ni ningún objeto de su jerarquía, de componente Rigidbody (no es lo mismo que un objeto marcado como static en el editor).
  - No deben moverse (aunque se puede) ni activarse ni desactivarse en ejecución.
  - No detectan las colisiones con objetos cinemáticos (son atravesados).
  - Detectan las colisiones con objetos dinámicos pero no sufren efectos físicos.

TEMA 6. FÍSICAS

- Uniones físicas entre cuerpos físicos: Joints
  - Componente Joint.
  - Permiten unir un cuerpo físico con otro o con un punto fijo.
  - Tipos:
    - Fixed Joint. Hacen depender un objeto de otro.
    - Hinge Joint. Une dos objetos como si estuviesen unidos por una bisagra.
    - Spring Joint. Une dos objetos como si estuviesen unidos por un muelle.



https://docs.unity3d.com/es/current/Manual/class-HingeJoint.html
https://docs.unity3d.com/es/current/Manual/class-SpringJoint.html
https://docs.unity3d.com/es/current/Manual/class-FixedJoint.html

TEMA 6. FÍSICAS

- Detección de objetos físicos mediante raycast
  - Permite detectar la intersección entre un rayo y los **colliders** de la escena.
  - Son métodos estáticos de la clase Physics.
  - No impactan sobre las caras internas de los colliders.
  - Métodos:
    - Raycast
      - Devuelve un bool que indica si ha habido colisión.
      - Recoge la información de colisión en un parámetro pasado por referencia (out)
    - RaycastAll.
      - Devuelve un array de los colliders que ha ido atravesando el rayo.
    - BoxCast.
    - CapsuleCast.
    - SphereCast.
    - …

TEMA 6. FÍSICAS

- Detección de objetos físicos mediante raycast
  - Ejemplo: raycast proyectado desde la cámara hasta el infinito. De manera complementaria se muestra una lína en la vista de escena de dicha proyección.

```
Vector3 posicionCamara = camara.transform.position;
            Vector3 forwardCamara = camara.transform.forward;
            Ray rayo = new Ray(posicionCamara, forwardCamara);
            //Debug.DrawLine(camara.transform.position, camara.transform.position + forwardCamara *
100, Color.red, 5);
            Debug.DrawRay(camara.transform.position, forwardCamara * 100, Color.red, 5);
            RaycastHit infoImpacto;
            if (Physics.Raycast(rayo, out infoImpacto))
            {
                print(infoImpacto.collider.gameObject.name);
            }
```

TEMA 6. FÍSICAS

- Detección de objetos físicos mediante raycast
  - Ejemplo: utilización del método OnDrawGizmos para visualizar la trayectoria del rayo.

```
private void OnDrawGizmos()
    {
        Vector3 posicionCamara = camara.transform.position;
        Vector3 forwardCamara = camara.transform.forward;
        Ray rayo = new Ray(posicionCamara, forwardCamara);
        Gizmos.color = Color.red;
        Gizmos.DrawLine(posicionCamara, posicionCamara + forwardCamara * 100);
    }
```

TEMA 6. FÍSICAS

- Otros métodos de la clase Physics.
  - OverlapBox.
  - OverlapSphere.
  - OverlapCapsule.

- Alternativas 2D.

- Otros componentes físicos
  - Ragdoll
  - Cloth

TEMA 6. FÍSICAS

- Físicas y bucles de juego
  - Bucle de juego de paso variable: el método Update.
    - El tiempo que transcurre entre dos frames es variable.
    - Requiere utilizar la propiedad Time.deltaTime para ajustar la velocidad.
  - Bucle de juego de paso fijo: el método FixedUpdate.
    - El tiempo que transcurre entre dos frames es fijo.
    - Configurable en Edit->Project Settings->Time.
    - Debe albergar las acciones físicas para que estas se calculen de forma correcta. Excepción: si la acción física es puntual puede aplicarse en el método Update.
    - No debe albergar cálculos complejos ajenos a la física para garantizar los tiempo de ejecución del método.
  - Escala de tiempo:
    - Permite modificar la velocidad de ejecución del juego: Edit->Project Settings ->Time ->Time Scale.
    - Require ajustar correctamente la propiedad Fixed Timestep.