# Unity

## Máster en Programación de Videojuegos


**Ignacio Martínez Rodríguez**

Curso 2013-2014

# Character controller

## INTRODUCCION

- Los personajes tienen características especiales de movimiento que no funcionan bien con un motor de física:
  - Fricciones
  - Saltos
  - Movimientos en el aire
  - Escaleras
  - No giran en algún eje (Y)
  - Aceleraciones imposibles
  - Etc…

# Character controller

- Usaremos un tipo de "física" especial que encapsulamos en un **Character Controller**.
- Permitirá que un control preciso del personaje por el escenario.
- Se usa tanto para el jugador como los personajes controlados por la CPU.

# Character controller

## REQUERIMIENTOS

- Requerimientos que el **controller** tiene que proveer al personaje:
  - No debe resbalar en rampas por debajo de una pendiente máxima.
  - No debe subir obstáculos por encima de una altura determinada.
  - No debe de perder velocidad al subir obstáculos.
  - Debe colisionar con el escenario.
  - Debe de deslizarse en las paredes.

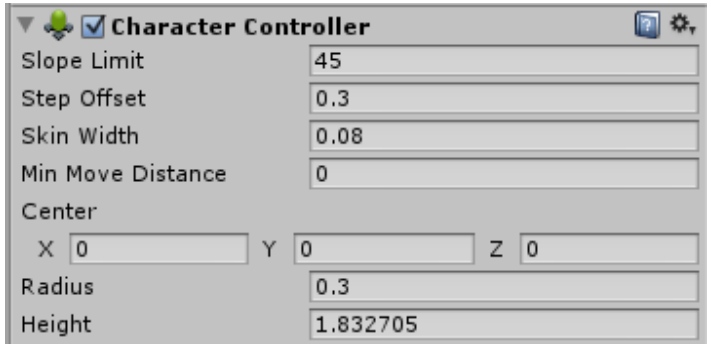
# Character controller

## NOTAS

- Cosas a tener en cuenta:
  - El **character controller** normalmente no interactúa con física.
  - Podemos hacer que afecte a objetos con física mediante programación.
  - El **character controller** no puede ser afectado por la física.

# Character controller

- **Unity** trae un **character controller** estándar que podemos usar.
- Component->Phisycs->Character Controller



# Character controller: variables

- **Height**: altura de la cápsula que actúa como collider.
- **Radius**: radio de la  cápsula que actúa como collider.
- **Center**: centro de la cápsula.
- **Slope limit**: límite de pendientes que se puede escalar.
- **Step Offset**:  límite de escalones que se puede subir.
- **Min Move Distance**: distancia mínima para moverse. Puede evitar temblores.
- **Skin Width**: si el personaje se atasca lo podemos modificar.

# Character controller: funciones

**bool isGrounded;**

Indica si el carácter está en el suelo.

```
using UnityEngine;
using System.Collections;

public class Example : MonoBehaviour {
    void Update() {
        CharacterController controller = GetComponent<CharacterController>();
        if (controller.isGrounded)
            print("We are grounded");
    }
}
```

# Character controller: funciones

Vector3 **velocity;**

Velocidad actual del carácter.

```
using UnityEngine;
using System.Collections;

public class Example : MonoBehaviour {
    void Update() {
        CharacterController controller = GetComponent<CharacterController>();
        Vector3 horizontalVelocity = controller.velocity;
        horizontalVelocity = new Vector3(controller.velocity.x, 0, controller.velocity.z);
        float horizontalSpeed = horizontalVelocity.magnitude;
        float verticalSpeed = controller.velocity.y;
        float overallSpeed = controller.velocity.magnitude;
    }
}
```

# Character controller: funciones

**bool SimpleMove(Vector3 speed);**

Mueve el carácter dada una velocidad. La velocidad en Y es ignorada. Se aplica la gravedad automáticamente.

```
using UnityEngine;
using System.Collections;

public class Example : MonoBehaviour {
    public float speed = 3.0F;
    public float rotateSpeed = 3.0F;
    void Update() {
        CharacterController controller = GetComponent<CharacterController>();
        float curSpeed = speed * Input.GetAxis("Vertical");
        controller.SimpleMove(transform.forward * curSpeed);
    }
}
```

# Character controller: funciones

**CollisionFlags Move(Vector3 motion);**

Mueve el carácter dado un desplazamiento. No aplica gravedad.

```
using UnityEngine;
using System.Collections;

public class Example : MonoBehaviour {
    public float speed = 6.0F;
    public float jumpSpeed = 8.0F;
    public float gravity = 20.0F;
    private Vector3 moveDirection = Vector3.zero;
    void Update() {
        CharacterController controller = GetComponent<CharacterController>();
        if (controller.isGrounded) {
            moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
            moveDirection *= speed;
            if (Input.GetButton("Jump"))
                moveDirection.y = jumpSpeed;

        }
        moveDirection.y -= gravity * Time.deltaTime;
        controller.Move(moveDirection * Time.deltaTime);
    }
}
```

# Character controller: funciones

CharacterController.OnControllerColliderHit(ControllerColliderHit)

Esta función es llamada cuando se produce una colisión con el carácter.

```
using UnityEngine;
using System.Collections;

public class Example : MonoBehaviour {
    public float pushPower = 2.0F;
    void OnControllerColliderHit(ControllerColliderHit hit) {
        Rigidbody body = hit.collider.attachedRigidbody;
        if (body == null || body.isKinematic)
            return;

        Vector3 pushDir = new Vector3(hit.moveDirection.x, 0, hit.moveDirection.z);
        body.velocity = pushDir * pushPower;
    }
}
```