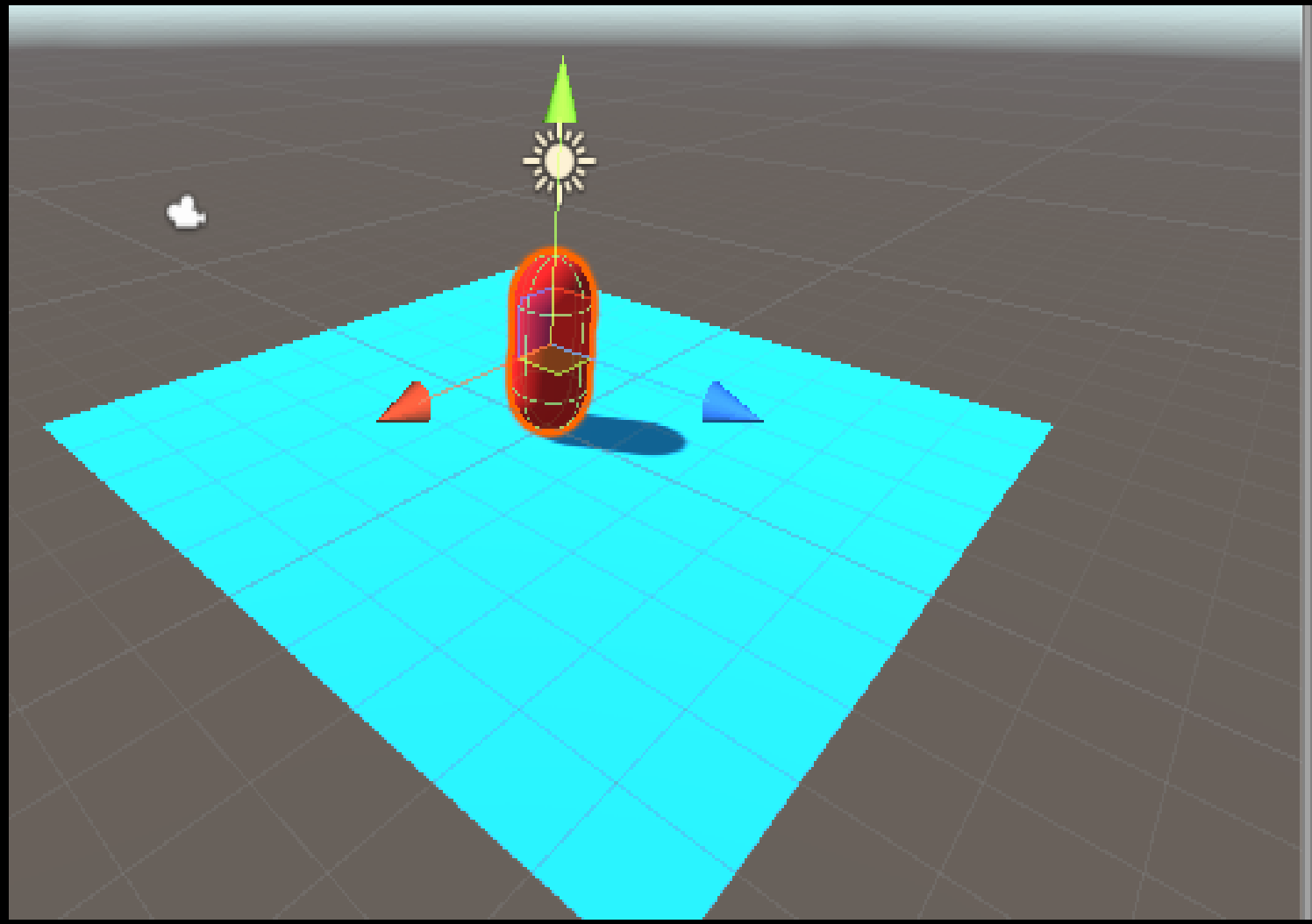
Universidad Continental

DESARROLLO DE VIDEO JUEGOS
CONTROL DE PERSONAJES
CHARACTER CONTROLLER

PEDRO YURI MARQUEZ SOLIS

CREAMOS UNA NUEVA ESCENA



Tag Untagged Layer Default

Transform

Position	X	0	Y	1	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Capsule (Mesh Filter)

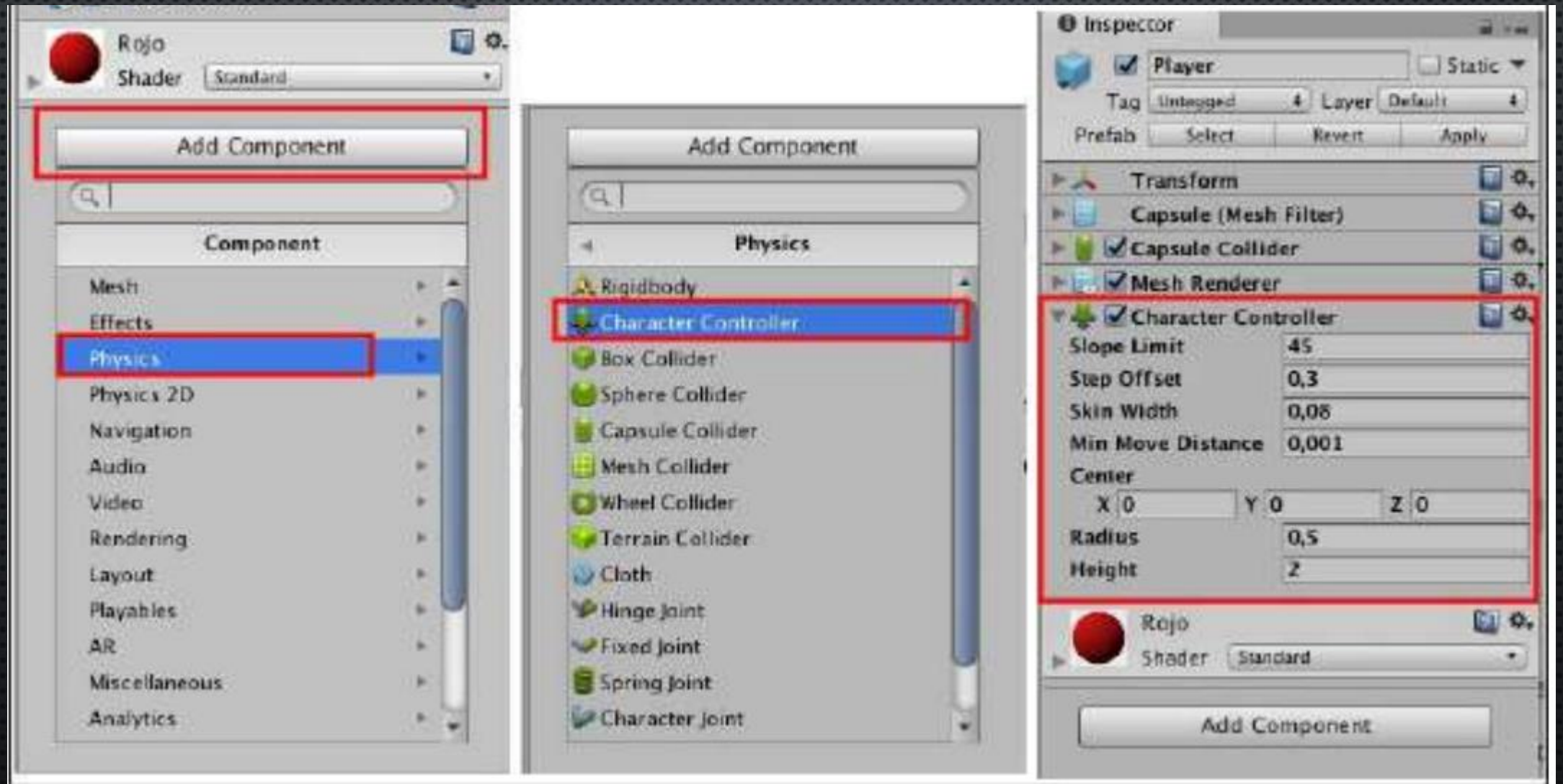
Mesh: Capsule

Mesh Renderer

Materials

Light Probes	Blend Probes
Reflection Probes	Blend Probes
Anchor Override	None (Transform)
Cast Shadows	On
Receive Shadows	☑
Motion Vectors	Per Object Motion
Lightmap Static	☐

AGREGAR EL CARÁCTER CONTROLLER



PROPIEDADES DEL CARÁCTER CONTROLLER

Propiedad	Función
Slope Limit	Limita el colisionador al subir sólo pendientes menos empinadas (en grados) que el valor indicado.
Step Offset	El personaje subirá una escalera sólo si está más cerca del suelo que el valor indicado. Este valor no debe ser mayor que la altura del Character Controller o generará un error.
Skin widht	Dos colisionadores pueden penetrar entre sí tan profundamente como su ancho de piel. Una anchura baja de la piel puede hacer que el personaje se quede pegado. Un buen ajuste seria el 10% del radio.
Min Move Distance	Si el personaje trata de moverse por debajo del valor indicado, no se moverá en absoluto. En la mayoría de situaciones este valor debe dejarse en 0.
Center	Esta propiedad ayuda a compensar el colisionador en forma de capsula en el espacio global, sin que afecte en el giro del personaje.
Radius	Longitud del radio del colisionador. Es la anchura del colisionador.
Height	Es la altura del colisionador. Es decir cuanto mayor es el valor, más se alarga el colisionador a lo largo del eje Y.

```
public class ControladorPlayer : MonoBehaviour
{
    public CharacterController controlador;

    void Start()
    {
        this.controlador = gameObject.GetComponent<CharacterController>();
    }

}
```

una variable pública de tipo Character Controller con el nombre controlador

En la función Start() accedemos al componente y guardamos la información en la variable.

AGREGANDO MOVIMIENTO

```
public class ControladorPlayer : MonoBehaviour
{
    public CharacterController controlador;
    public float velocidad;
    void Start()
    {
        this.controlador = gameObject.GetComponent<CharacterController>();
        this.velocidad= 15f;
    }
    void Update()
    {
        this.controlador.SimpleMove(Vector3.right * this.velocidad* Time.deltaTime);
    }
}
```

MOVER OBJETOS CON SIMPLEMOVE DIRECCIÓN LOCAL - MÉTODO TRANSFORM.TRANSFORMDIRECTION

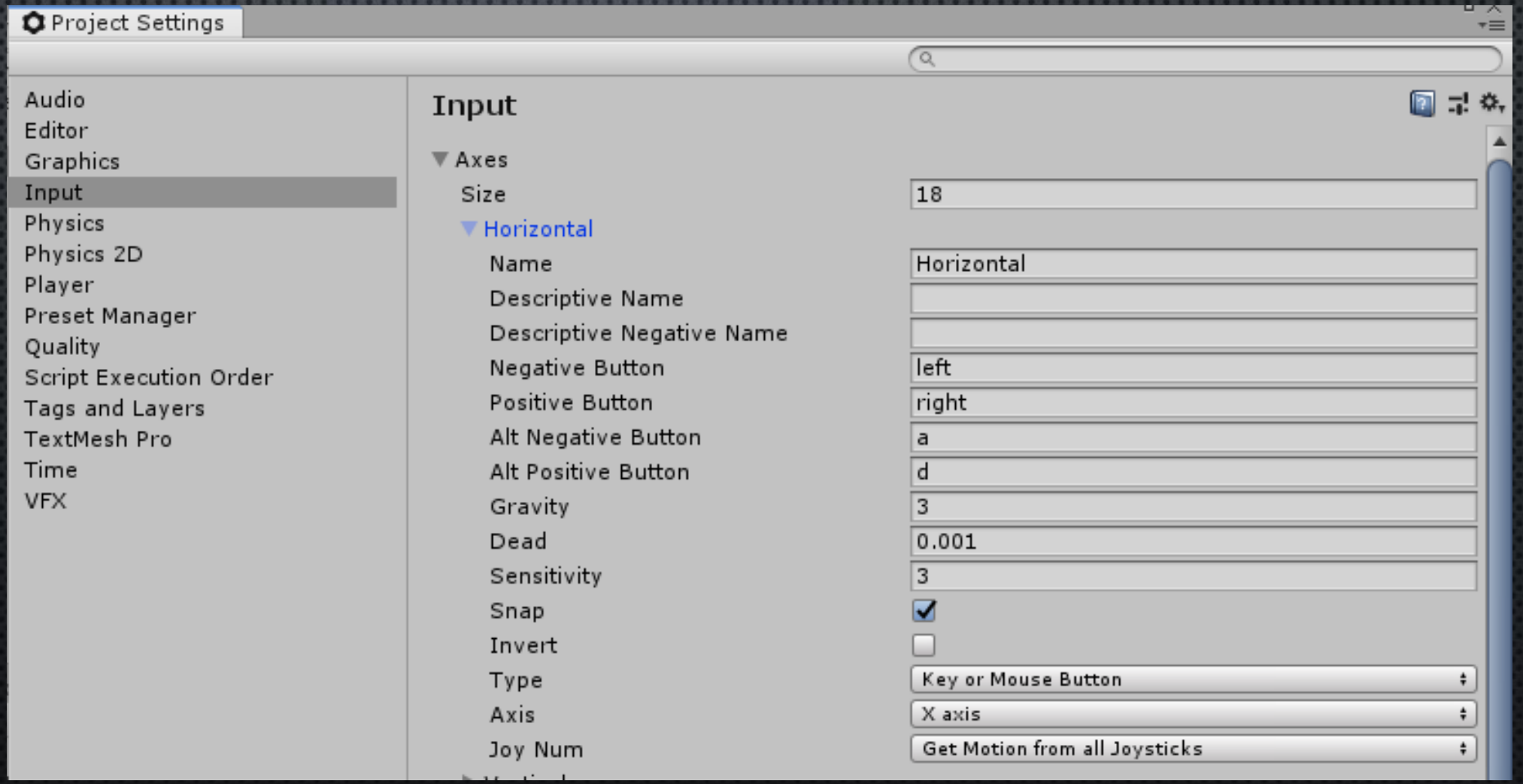
```
public class ControladorPlayer : MonoBehaviour
{
    public CharacterController controlador;
    public float velocidad;
    void Start()
    {
        this.controlador = gameObject.GetComponent<CharacterController>();
        this.velocidad= 15f;
    }
    void Update()
    {
        this.controlador.SimpleMove(transform.TransformDirection(Vector3.right * this.velocidad* Time.deltaTime));
    }
}
```

MOVER OBJETOS CON MOVE DIRECCIÓN LOCAL

```
public class ControladorPlayerMove : MonoBehaviour
{
    public CharacterController controlador;
    public float velocidad;
    public float gravedad;
    public Vector3 direccion;
    void Start()
    {
        this.controlador = gameObject.GetComponent<CharacterController>();
        this.velocidad = 2f;
        this.gravedad =-9.8f;
    }
    void Update()
    {
        this.direccion = transform.TransformDirection(Vector3.right);
        this.direccion.y = this.gravedad * Time.deltaTime;
        this.controlador.Move(this.direccion * this.velocidad * Time.deltaTime);
    }
}
```

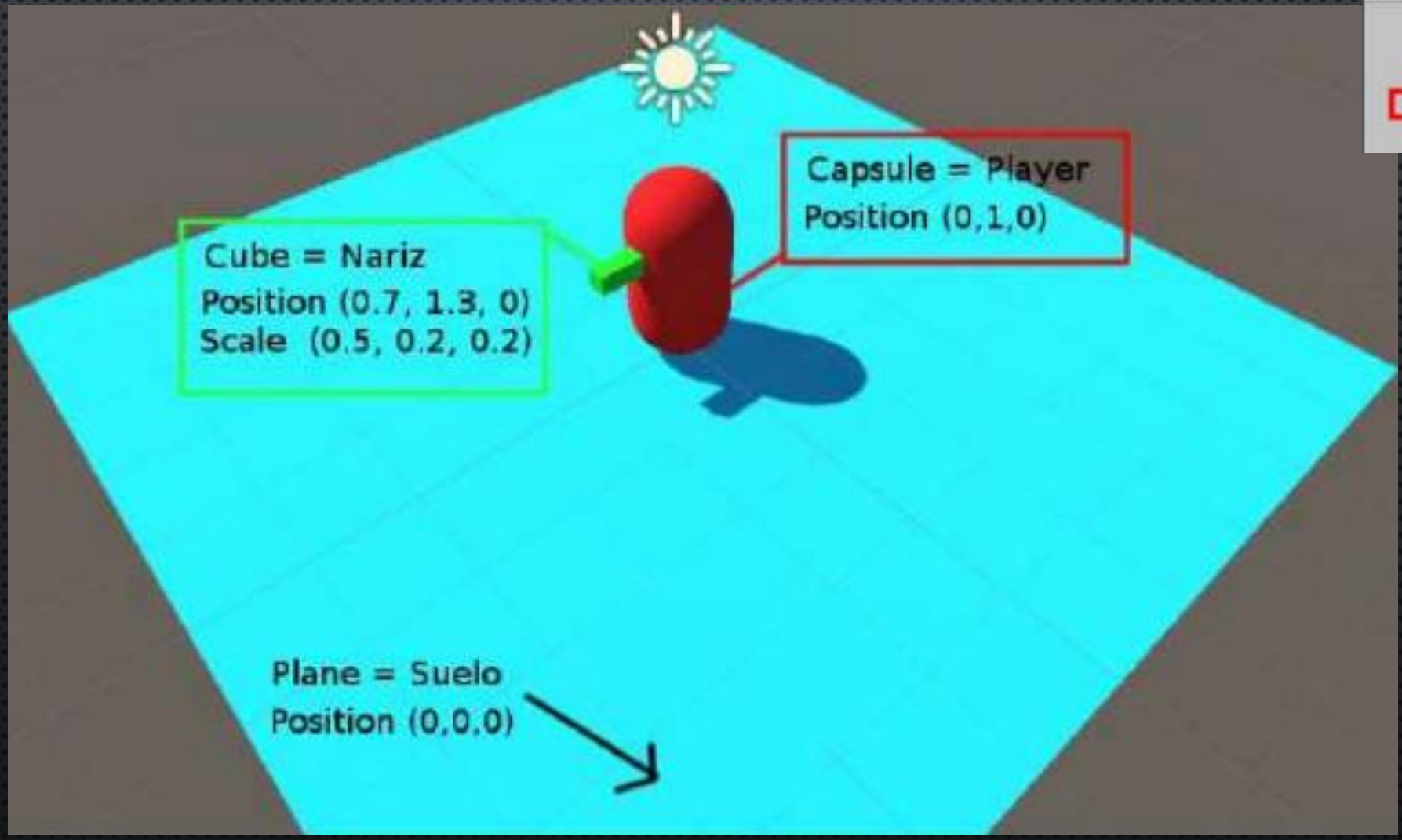
LOS INPUTS

- AL ACCEDER A EDIT > PROJECT SETTINGS > INPUT, NOS MUESTRA EL INPUTMANAGER.



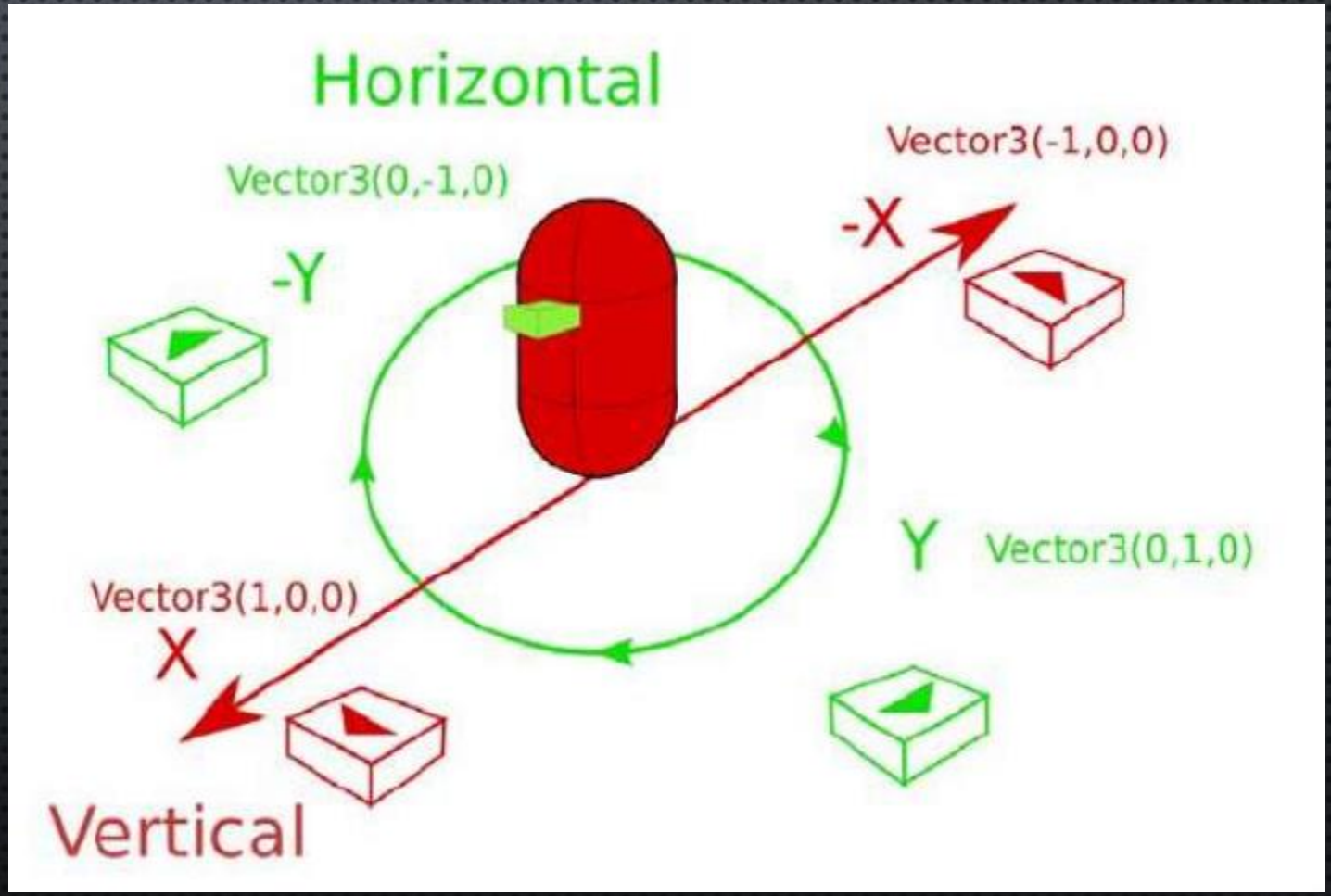
```
public class getAxisMetodo : MonoBehaviour
{
    public CharacterController controlador;
    public float velocidad;
    public float gravedad;
    public Vector3 direccion;
    void Start()
    {
        this.controlador = gameObject.GetComponent<CharacterController>();
        this.velocidad= 2f;
        this.gravedad = - 9.8f;
    }
    void Update()
    {
        this.direccion = Vector3.zero;
        this.direccion.y -= this.gravedad * Time.deltaTime;
        this.direccion.x = Input.GetAxis("Horizontal") * velocidad;
        this.direccion.z = Input.GetAxis("Vertical") * velocidad;
        this.controlador.Move(this.direccion * Time.deltaTime);
    }
}
```

MOVER Y ROTAR CHARACTER CONTROLLER



DESCRIPCIÓN DEL MOVIMIENTO

CREAREMOS UN MOVIMIENTO DE DESPLAZAMIENTO HACIA ADELANTE Y HACIA ATRÁS CONTROLADAS CON LAS FECHAS DEL TECLADO UP , DOWN (ARRIBA, ABAJO) Y PARA LA ROTACIÓN UTILIZAREMOS LAS FLECHAS DEL TECLADO LEFT Y RIGHT (IZQUIERDA, DERECHA).



```
public class MovimientoPlayer : MonoBehaviour
{
  public CharacterController controlador;
  public Vector3 direccion;
  public float gravedad=9.8f;
  public float rotacion;
  public float speed = 5f;
  public float rotSpeed=5f;

   void Start()
    {
        this.controlador = gameObject.GetComponent<CharacterController>();
    }

   void Update()
    {
        direccion = gameObject.transform.TransformDirection(new Vector3(Input.GetAxis("Vertical"),0,0) * speed);
        rotacion = Input.GetAxis("Horizontal")*rotSpeed;
        direccion -= new Vector3(0, gravedad * Time.deltaTime, 0);
        controlador.transform.Rotate(new Vector3(0f,rotacion,0f));
        controlador.Move(direccion * Time.deltaTime);
    }
}
```

SALTAR OBSTÁCULOS

```
public class MovPlayerWithJump : MonoBehaviour
{   public CharacterController controlador;
    public Vector3 direccion;
    public float gravedad = 9.8f; public float rotacion; public float speed = 5f;
    public float rotSpeed = 5f;  public float salto = 50f;
     void Start()
    {
        this.controlador = gameObject.GetComponent<CharacterController>();
    }

    void Update()
    {
        if (controlador.isGrounded)
        {
            direccion = gameObject.transform.TransformDirection(new Vector3(Input.GetAxis("Vertical"), 0, 0) * speed);
            rotacion = Input.GetAxis("Horizontal") * rotSpeed;
            if (Input.GetKeyDown(KeyCode.Z))
            {
                direccion.y += salto * Time.deltaTime * speed;
                controlador.transform.Rotate(Vector3.zero);
            }
        }
        direccion -= new Vector3(0, gravedad * Time.deltaTime, 0);
        controlador.transform.Rotate(new Vector3(0f,rotacion,0f));
        controlador.Move(direccion * Time.deltaTime);
    }
}
```

isGrounded booleano indica si toca el suelo o no

Se presiona la tecla “Z”?

SÓLO PARA TENER EN CUENTA

Vector3	Variable estática
new Vector3 (0,0,0);	Vector3.zero
new Vector3 (1,1,1);	Vector3.one
new Vector3 (0,1,0);	Vector3.up
new Vector3 (0,0,1);	Vector3.forward
new Vector3 (1,0,0);	Vector3.right