

TUTORIAL GUÍA DE UNITY : ROLL A BALL

Integrantes: Kelvin Paul Pucho Zevallos

Profesor: Diego Alonso Iquira Becerra

Fecha de realización: 16 de mayo de 2022

Fecha de entrega: 17 de mayo de 2022

Arequipa - Perú

Índice de Contenidos

1. Creación de Objetos	1
1.1. Vista de Juego	1
1.2. Hierarchy	3
2. Assets	4
3. Materials	4
4. Inspector del Objeto Player	5
5. Input Actions	6
6. Codigos	7
6.1. Rotator	7
6.2. Camera Controller	7
6.3. Player Controller	8
Referencias	14

Índice de Figuras

1. Escenario	1
2. Vista del Juego	2
3. Vista del Box Collider	2
4. Hierarchy	3
5. Assets	4
6. Materials	4
7. Sphere Collider, Rigidbody, Player Input	5
8. Player Controller (Script), Character Controller	6
9. Input Actions	6
10. Rotator.cs	7
11. CameraControllerr.cs	8
12. PlayerControllerr.cs	9
13. PlayerControllerr.cs	10
14. PlayerControllerr.cs	10
15. PlayerControllerr.cs	11
16. PlayerControllerr.cs	11
17. PlayerControllerr.cs	12
18. PlayerControllerr.cs	13

1. Creación de Objetos

El tutorial se realizó correctamente y se importó un modelo Hoop, después se colocó el resto del tablero.

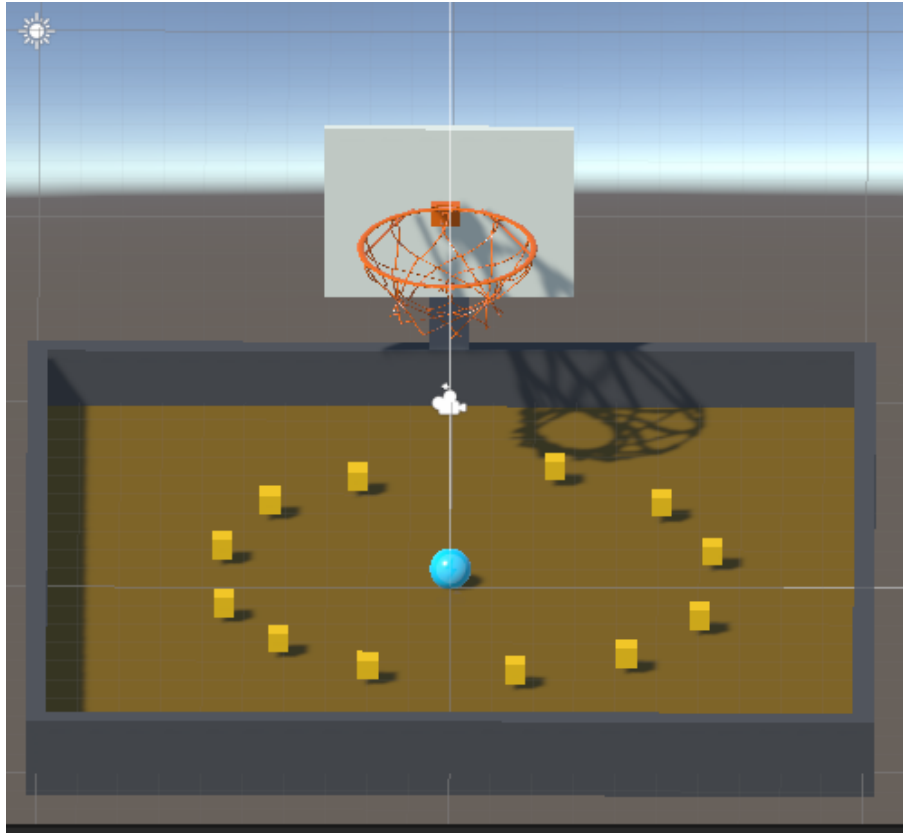


Figura 1: Escenario

1.1. Vista de Juego

La cámara enfoca la ubicación del juego además de que en las esquinas superior se lleva el conteo de cubos colisionados y el conteo de puntos encestados. Para encestar y sumar puntos se hizo uso del Mesh Collider y Box Collider.



Figura 2: Vista del Juego

El mesh Collider es la canasta en si pero para poder generar puntos, la pelota se debe encestar por ello se uso el Box collider modificado de la siguiente manera:

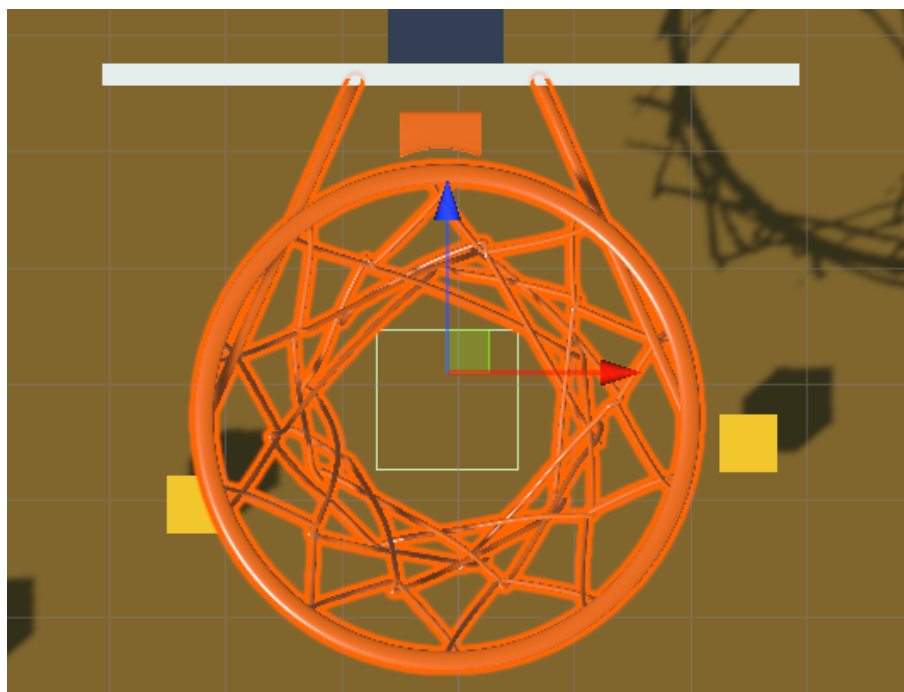


Figura 3: Vista del Box Collider

Cuando la pelota pasa por dentro de la canasta colisiona con el box collider del objeto de la canasta. Y mediante el trigger se obtiene la instancia del collider para codificar en el script. Para tener éxito se debe desactivar el Is Trigger del Box Collider y tener un tag definido del objeto. para poder localizar con que objeto esta colisionando la pelota.

1.2. Hierarchy

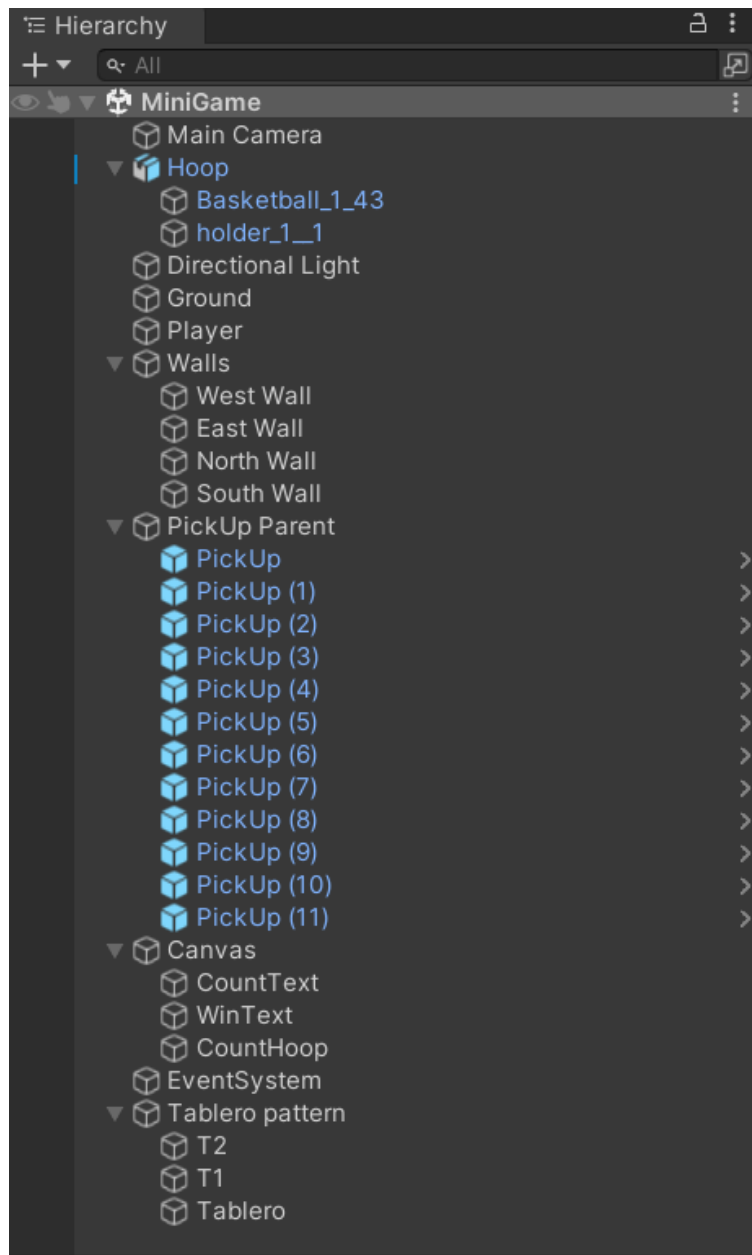


Figura 4: Hierarchy

Aparte de los objetos 3D creados en el tutorial se uso el tablero pattern y el Prefabs Hoop que representa a la modelo de la canasta

2. Assets

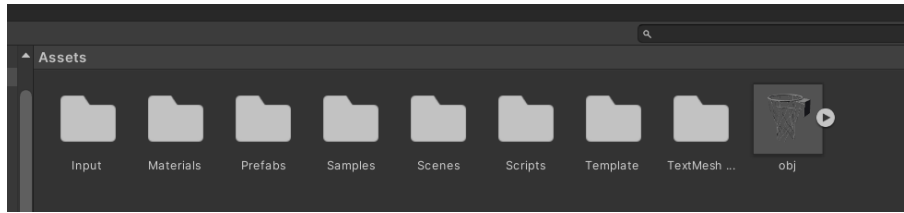


Figura 5: Assets

3. Materials

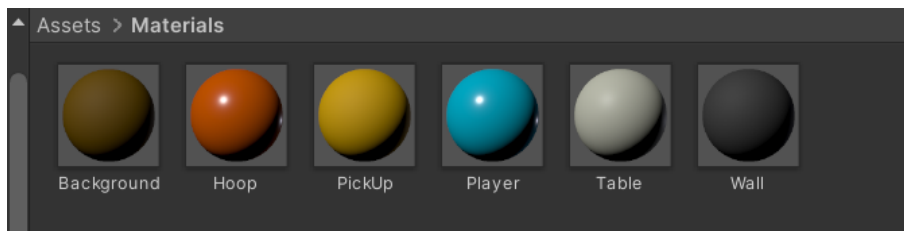


Figura 6: Materials

4. Inspector del Objeto Player

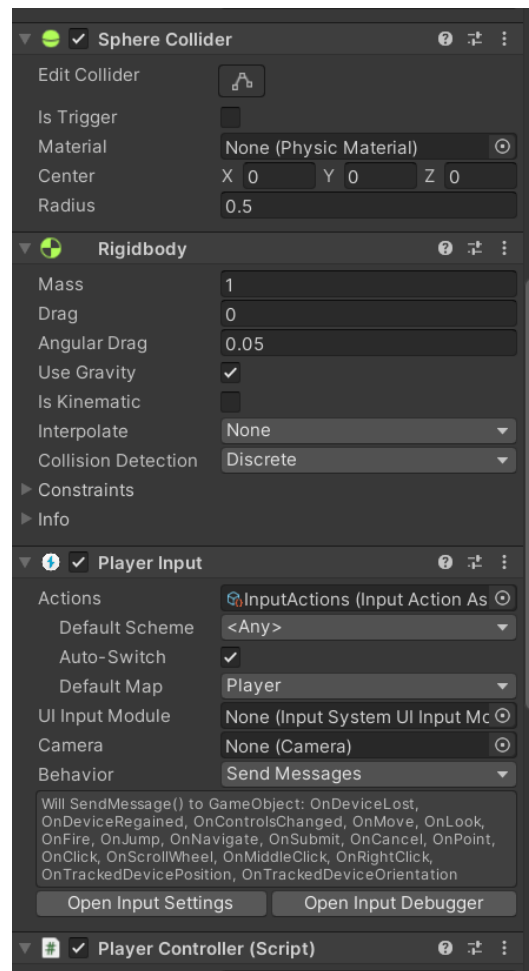


Figura 7: Sphere Collider, Rigidbody, Player Input



Figura 8: Player Controller (Script), Character Controller

5. Input Actions

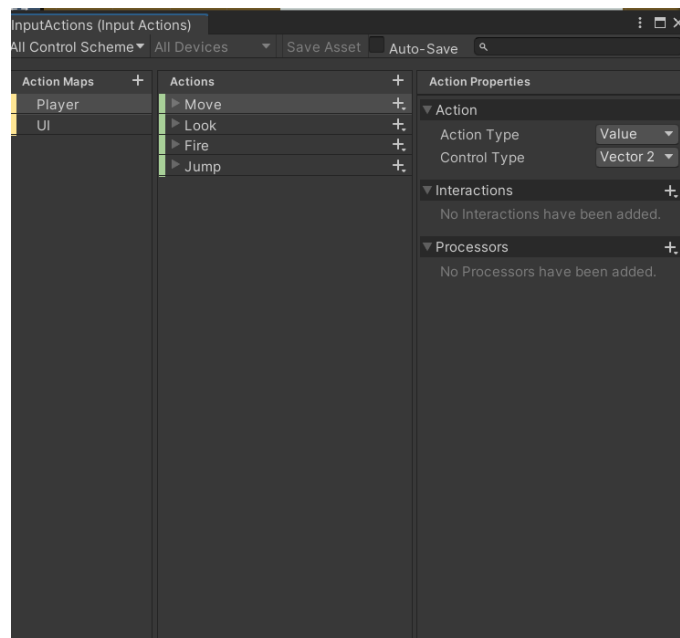


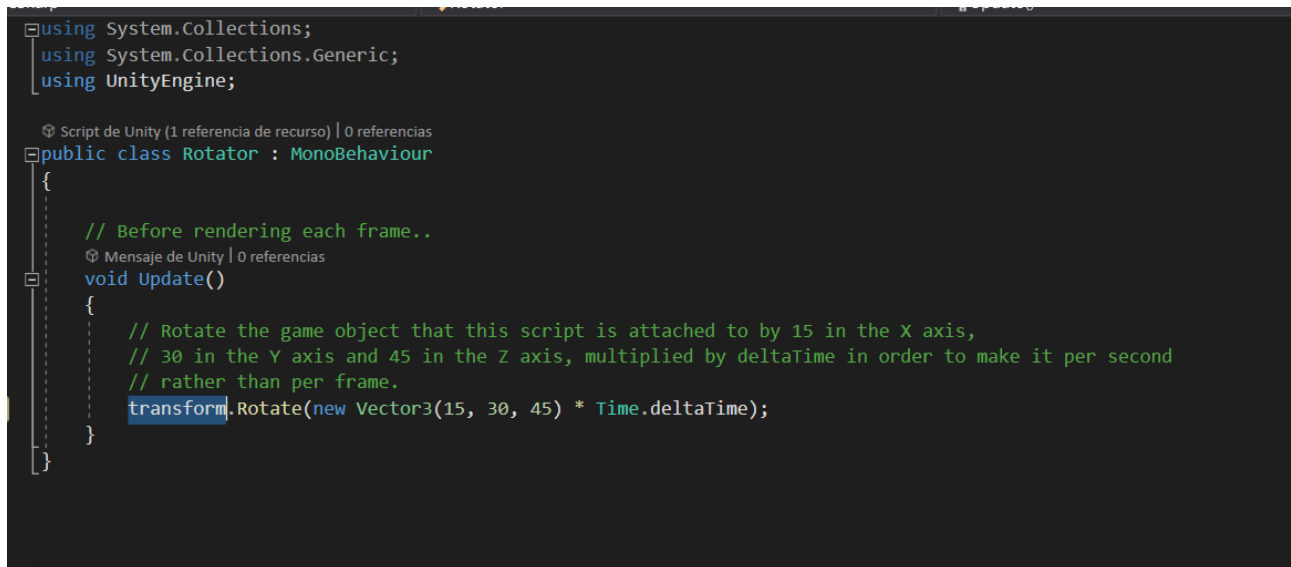
Figura 9: Input Actions

Algunas entradas vienen predefinidas pero tuve que crear una acción Jump con tipo de acción botón y con el vínculo de la tecla espacio para que pueda saltar mi personaje.

6. Codigos

6.1. Rotator

Este código es solo para rotar los prefabs que son cubos amarillos. Se uso dentro de la función Update para que sea renderizado en cada frame

A screenshot of a code editor showing the Rotator.cs script. The script is a C# class that inherits from MonoBehaviour. It contains a single method, Update, which is called every frame. Inside the Update method, the transform of the game object is rotated by 15 degrees around the X-axis, 30 degrees around the Y-axis, and 45 degrees around the Z-axis, multiplied by Time.deltaTime to make the rotation per second instead of per frame. The code is as follows:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Rotator : MonoBehaviour
{
    // Before rendering each frame..
    void Update()
    {
        // Rotate the game object that this script is attached to by 15 in the X axis,
        // 30 in the Y axis and 45 in the Z axis, multiplied by deltaTime in order to make it per second
        // rather than per frame.
        transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
    }
}
```

Figura 10: Rotator.cs

6.2. Camera Controller

Este codigo es para que la camara siga al objeto definido mediante la diferencia de sus posiciones para hallar su distancia.

Como se requiere que la cámara persiga todo el rato al jugador entonces se debe renderizar en cada frame por ello se inserta en la función LateUpdate, esta función es para prevenir errores al momento de un transform de un objeto así que primero espera a que se ejecuten todos las funciones Update para luego ejecutar LateUpdate.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Script de Unity (1 referencia de recurso) | 0 referencias
public class CameraController : MonoBehaviour
{
    public GameObject player;

    private Vector3 offset;

    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        offset = transform.position - player.transform.position;
    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void LateUpdate()
    {
        transform.position = player.transform.position + offset;
    }
}
```

Figura 11: CameraControllerr.cs

6.3. Player Controller

Para capturar los eventos y enviar resultados a un text editor se requiere importar las clases como son el INputSystem y TMpro.

Existen tipos de variables primitivas y de unity las cuales declare para instanciar o agregar valores ya sea privativa o publica.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.InputSystem;
using TMPro;

Script de Unity (1 referencia de recurso) | 0 referencias
public class PlayerController : MonoBehaviour
{
    private Rigidbody rb;
    private CharacterController chController;
    public float speed = 5;

    private float movementX;
    private float movementY;
    private int count;
    private int countHoop;

    //jump variables
    public float jumpSpeed = 5;
    public float gravity = 9.8f;
    public float coeffl = 1;
    private bool jumpPress = false;

    private Vector3 movement;
    private bool isGround;

    public TextMeshProUGUI countText;
    public TextMeshProUGUI HoopText;
    public GameObject winTextObject;

    // Start is called before the first frame update
```

Figura 12: PlayerControllerr.cs

En esta función es donde se definen por primera vez las variables declaradas. Además de llamar a la función para que sea ejecutada una vez empiece el juego.

```
void Start()
{
    rb = GetComponent<Rigidbody>();
    chController = GetComponent<CharacterController>();

    count = 0;
    countHoop = 0;
    SetCountText();
    SetCountHoop();
    winTextObject.SetActive(false);
}
```

Figura 13: PlayerControllerr.cs

El code incorporado en ambas funciones con términos Update son llamados cada frame, la diferencia de las dos funciones es que FixedUpdate se usa para las interacciones físicas de un objeto en el juego, mientras que Update es para transform.

```
private void Update()
{
    movement = new Vector3(movementX, 0.0f, movementY);
    chController.SimpleMove(movement * speed);
}

private void FixedUpdate()
{
    movement = new Vector3(movementX, 0.0f, movementY);
    rb.AddForce(movement * speed);
    actionJump();
}
```

Figura 14: PlayerControllerr.cs

Con la función de la figura 15 se puede capturar el evento del teclado y tomar el valor del Input, para luego almacenar las posiciones de movimiento.

```
private void OnMove(InputValue movementValue)
{
    Vector2 movementVector = movementValue.Get<Vector2>();

    movementX = movementVector.x;
    movementY = movementVector.y;
}
```

Figura 15: PlayerControllerr.cs

Después la función `OnTriggerEnter` es aquella que captura las colisiones que tengo el objeto player con los objetos restantes que tienen un boc Collider. En esta funcion al igual que el tutorial use el metodo `CompareTag` del objeto player para ver con que objeto esta colisionando con un tag de Hoop y así aumentar el puntaje de pelotas encestadas.

```
private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("PickUp"))
    {
        other.gameObject.SetActive(false);
        count++;
        // Run the 'SetCountText()' function (see below)
        SetCountText();
    }

    if (other.gameObject.CompareTag("Hoop"))
    {
        countHoop++;
        SetCountHoop();
    }
}
```

Figura 16: PlayerControllerr.cs

En la funcion `OnJump` es donde capturo la accion hecha por el usuario para verificar si el objeto esta en el suelo y si se ha pulsado el boton espacio para modificar una variable de tipo bool que indica true si la pelota salto y false sino.

La función `actionJump` en donde modifiko la posición del objeto player siempre y cuando cumpla las condiciones de estar presionado por la tecla espacio y por ubicarse en una superficie. La modificación lo hice a mi criterio con un velocidad de salto, la gravedad, duplicar el valor y por ultimo una variable publica `coeffL` que puede ser modificada mediante la ventana de unity para que pueda aumentar la altura del salto. Y así llegar a la canasta.

Si la pelota esta sobre el aire se debe disminuir el eje y para que baje según la gravedad que se defina y con el tiempo establecido para una caída que simula la realidad.

```
1 referencia
private void actionJump()
{
    if (isGround)
    {
        movement.y = 0.0f;
    }

    if (jumpPress && isGround)
    {
        movement.y += (jumpSpeed * gravity * 2* coeffl);
        jumpPress = false;
        Debug.Log("JUMP");
    }

    movement.y -= (gravity * Time.deltaTime);

    chController.Move(movement * Time.deltaTime);
}

0 referencias
private void OnJump()
{
    Debug.Log("Jump");
    isGround = chController.isGrounded;
    if (isGround)
    {
        jumpPress = true;
        Debug.Log("jumpepress");
    }
}
```

Figura 17: PlayerControllerr.cs

Y por ultimo las funciones para retornar los puntos adquiridos en cada colisión.

```
}  
2 referencias  
void SetCountText()  
{  
    countText.text = "Count: " + count.ToString();  
    if (count >= 12)  
    {  
        // Set the text value of your 'winText'  
        winTextObject.SetActive(true);  
    }  
}  
  
2 referencias  
void SetCountHoop()  
{  
    HoopText.text = "Puntos: " + countHoop.ToString();  
}
```

Figura 18: PlayerControllerr.cs

Referencias

- **Tutorial Roll a Ball:** <https://learn.unity.com/tutorial/building-the-game?uv=2019.4&projectId=5f158f1bedbc2a0020e51f0d#>
- **Modelo del objeto Hoop:** <https://www.turbosquid.com/AssetManager/Index.cfm?stgAction=getFiles&subAction=Download&intID=624665&intType=3&csrf=517FF4719D075B365F2ECF867D28D61D3244C716&showDownload=1&s=1>
- <https://code.tutsplus.com/es/tutorials/create-a-basketball-free-throw-game-with-unity--cms-21203>
- <https://gamedevbeginner.com/how-to-jump-in-unity-with-or-without-physics/>
- <https://medium.com/nerd-for-tech/moving-with-the-new-input-system-unity-a6c9cb100808>
- <https://docs.unity3d.com/ScriptReference/CharacterController.SimpleMove.html>
- <https://www.youtube.com/watch?v=qc0xU2Ph86Q>
- <https://www.youtube.com/watch?v=m5WsmLEOFiA>
- <https://www.youtube.com/watch?v=cnSgA4OIEk&t=119s>