Character Controllers

Game Development Foundations

Last modified 27/01/16 by Sam Cartwright

Contents

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

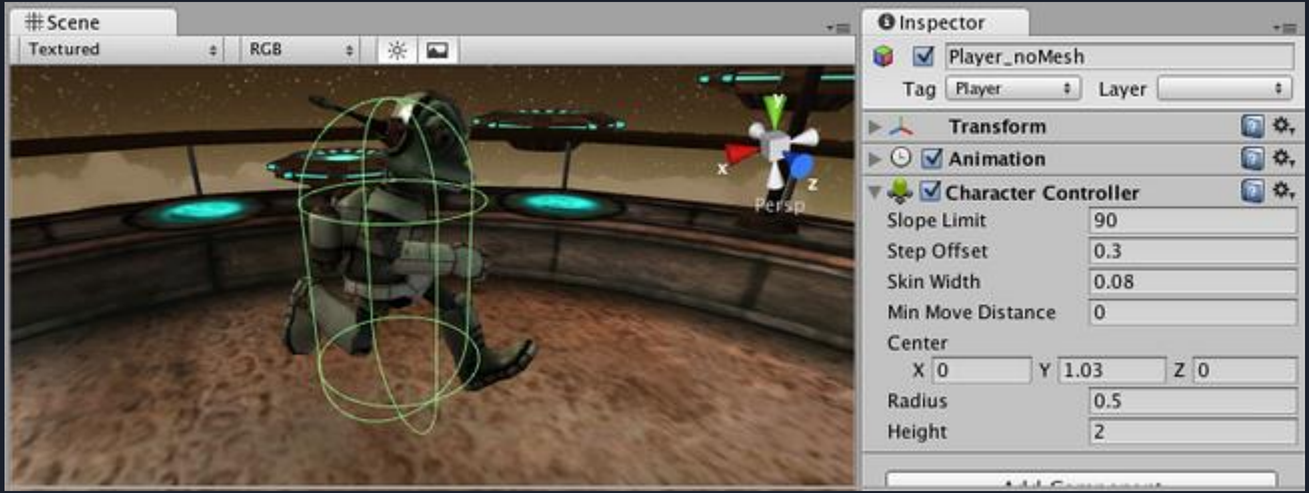
What is the Character Controller?

- Character Controllers are mainly used for 3rd or 1st person player control where you aren't using physics (Rigidbody objects)
- In your traditional Doom-style FPS, controls are not truly physically realistic (sometimes not even close!)
  - Characters might run, stop immediately, and turn on the spot; rocket jump; double-jump; etc.

What is the Character Controller?

- Character controllers don’t use physics
- Instead, they provide an easy (easier) way to move characters
  - Often we don’t really need our player/enemies to physically act/react in the same way as scenery
- When game characters move using real-world physics they often don’t “feel right”
  - Character controllers can solve this problem

How Can We Use It?



Player_noMesh
Tag Player Layer

Character Controller

Slope Limit	90
Step Offset	0.3
Skin Width	0.08
Min Move Distance	0
Center	X 0 Y 1.03 Z 0
Radius	0.5
Height	2

- Attach the Character Controller to your character
- But, you must pass it input via a script
- It is designed to be used *without* a Rigidbody attached

How Can We Use It?

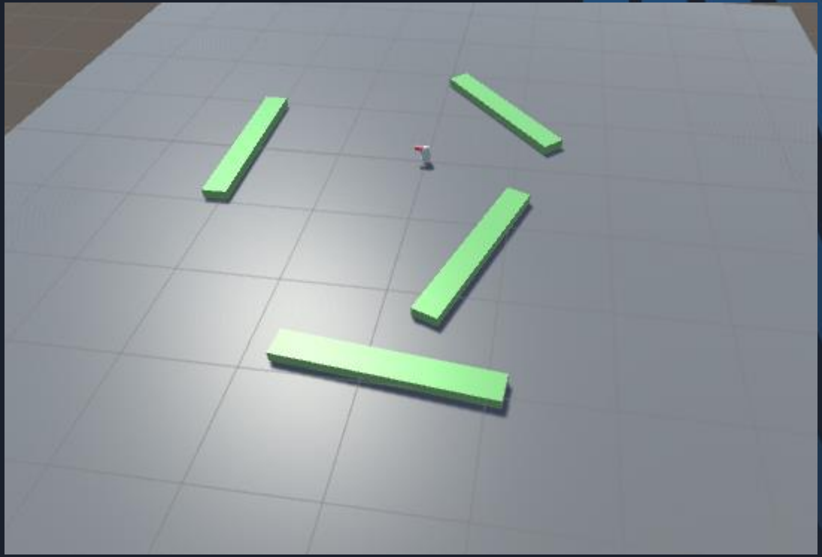
- The character controller by itself won’t move your character
- You must tell the controller *how* to move in response to input

```
using UnityEngine;
using System.Collections;

public class CharacterMovement : MonoBehaviour {

    public float MoveSpeed = 0.5f;
    public float RotateSpeed = 1f;
    CharacterController cc;

    void Start () {
        cc = GetComponent<CharacterController>();
    }

    void Update () {
        // rotate the character according to left/right key presses
        transform.Rotate(Vector3.up *
                Input.GetAxis("Horizontal") * RotateSpeed);
        // move forward/backward according to up/down key presses
        cc.Move(transform.forward *
                Input.GetAxis("Vertical") * MoveSpeed);
        // apply gravity
        cc.SimpleMove(Physics.gravity);
    }
}
```

aie SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

How Can We Use It?

- The controller will automatically respond to colliders in the scene
  - The colliders don't need to be physical objects (rigidbodies) either
- Try the sample scene provided



Fine-Tuning Your Character

- Modify the *Height* and *Radius* to fit your character’s mesh
  - It is recommended to always use around 2 meters for a human-like character
- You can also modify the center of the capsule if your pivot point is not the exact center of the character

Fine-Tuning Your Character

- *Step Offset* controls how high an obstacle can be before the character can’t step over it
  - Set this between 0.1 and 0.4 for a 2 meter sized human
- *Slope Limit* controls how steep a slope the character can climb
  - Don’t make this too small.
  - Often a value of 90 degrees works best
  - The character won’t be able to climb walls due to the capsule shape

Don’t Get Stuck

- *Skin Width* controls how deep two colliders can penetrate each other
- Its one of the most important properties to get right when setting up your character
  - Larger skin width reduces jitter
  - Lower skin width can cause the character to get stuck
  - A good value is 10% of the *Radius*, and at least greater than 0.01

Summary

- Character controllers are convenient to use when your player character doesn’t need to use physics
- Character controllers are useful for 1st and 3rd person games
- The controller will automatically react to any collision objects in your game (no rigidbodies needed)
- Movement must still be controlled by scripts

References

- Unity Technologies. 2016. *Unity - Manual: Character Controller*. [ONLINE] Available at: http://docs.unity3d.com/Manual/class-CharacterController.html. [Accessed 28 January 2016].