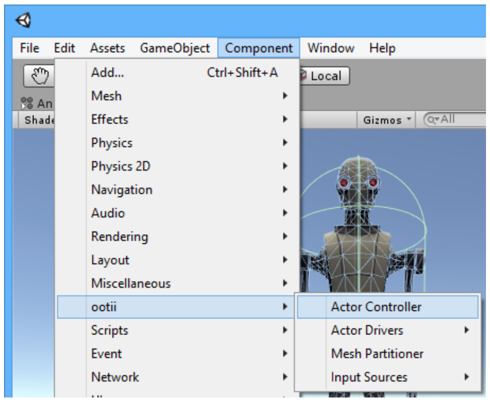
ACTOR CONTROLLER

10/21/2105

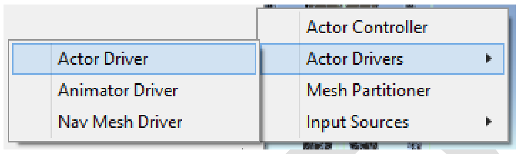
Quick Start

1. **Import** the Actor Controller package into your project.

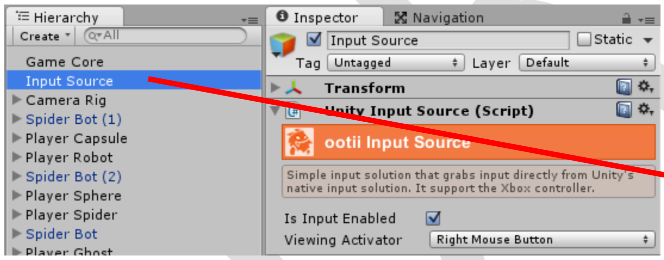
2. Select your character and add an **Actor Controller**.



3. Add an **Actor Driver** to your character.



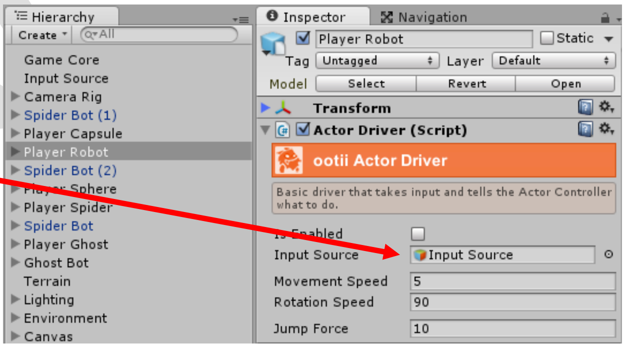
4. Add an **Input Source** to the scene and assign it to your Actor Driver.



Input Source — Static
Tag: Untagged — Layer: Default

Simple input solution that grabs input directly from Unity's native input solution. It support the Xbox controller.

Is Input Enabled	✓
Viewing Activator	Right Mouse Button

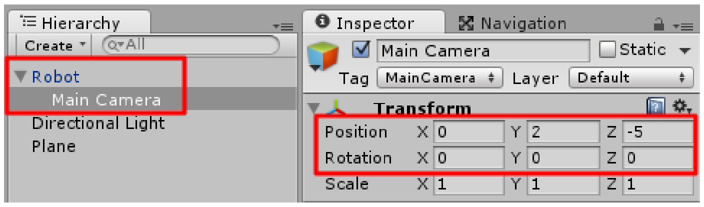


Player Robot — Static
Tag: Untagged — Layer: Default

Basic driver that takes input and tells the Actor Controller what to do.

Is Enabled	☐
Input Source	Input Source
Movement Speed	5
Rotation Speed	90
Jump Force	10

5. Parent the **camera** to your character (optional).



Main Camera — Static
Tag: MainCamera — Layer: Default

	X	Y	Z
Position	0	2	-5
Rotation	0	0	0
Scale	1	1	1

Foreword

Thank you for purchasing the Actor Controller!

I'm an independent developer and your feedback and support really means a lot to me. Please don't ever hesitate to contact me if you have a question, suggestion, or concern.

I'm also on the forums throughout the day:

Tim
tim@ootii.com

Overview

The Actor Controller is a replacement for Unity's standard character controller and provides advanced features.

While you can use it alone in any Unity 5 solution, it is also the foundation for the Motion Controller.

Features

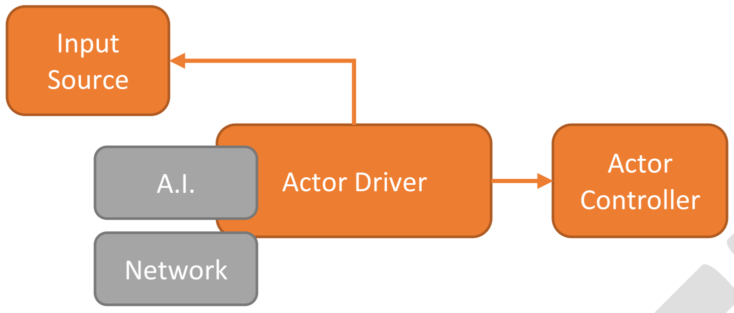
The Actor Controller supports the following features:

- Walk on walls, ceilings, etc.
- Move and rotate on platforms
- Move super-fast
- Slide on steep slopes
- Orient character to ground slope
- Create custom body shapes
- React to external forces
- Supports Nav Mesh Agents
- Supports root-motion
- Supports any input solution
- Zero Garbage Collection *
- Includes code (C#)

* Zero garbage collection when running in release mode. While in debug, some unity physics calls create minor amounts of garbage.

The Basics

In order to keep the Actor Controller as flexible as possible, I've created a clear distinction between the Actor Controller, the Actor Driver, and the Input Source. This allows you to use different components with the Actor Controller: including assets from other developers.



Each update, the **Actor Driver** gathers input from an **Input Source** or other creates its own and tells the actor what to do.

The **Actor Controller moves and rotates** while respecting the environment.

Actor Controller

Controls full-body character movement and rotation in response to the environment and external forces.

Actor Driver

Determines how the Actor Controller moves based on user input, AI, etc. The Actor Drivers I've included are usable, but can also be expanded on. In fact, you can create your own driver as needed.

Input Source

Used to gather user input from the keyboard, mouse, gamepads, etc. By splitting this out, you could use the basic Unity Input Source provided or create an input source that uses a 3rd party input solution.

With this approach, you can use any input solution you want. You can also change how your actor moves (ie walking vs. flying) by changing how the Actor Driver directs the Actor Controller.

This also means that different characters in the same scene could be controlled in different ways or even by different players.

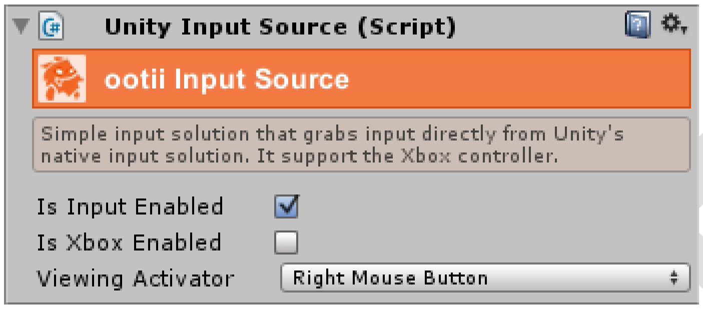
Custom Input Sources

Remember, the input source that I've included is option and only required by the Actor Driver if the driver itself requires user input.

For example, a spider using its own AI doesn't need an input source since it's not using user input.

Unity Input Source – Assets/ootii/Framework_v1/Code/Input

This is a default input source that just uses Unity's native Input Manager solution to read input. It supports basic movement and viewing with the keyboard, mouse, and Xbox controller.


Unity Input Source (Script)
ootii Input Source
Simple input solution that grabs input directly from Unity's native input solution. It support the Xbox controller.
Is Input Enabled ✓
Is Xbox Enabled ☐
Viewing Activator: Right Mouse Button

To enable the Xbox controller, just check the box. The appropriate values will be added to Unity's Input Manager list (if needed).

The 'Viewing Activator' property allows you to determine how rotation/viewing is activated. For example, when set to 'Right Mouse Button', actors will only rotate when the right mouse button is held down.

In the end, it's the driver's responsibility for honoring the input source as needed.

IInputSource Interface

For developers creating their own Input Sources, remember to implement the IInputSource interface. Once you do this, you can fill in the properties and functions based on the input solution you're using.

Custom Actor Drivers

Remember, the drivers that I've included are optional. These drivers handle basic movement cases and work for a variety of situations. However, you can also create your own driver to control the actor however you want.

In the end, the drivers are really calling functions in the Actor Controller:

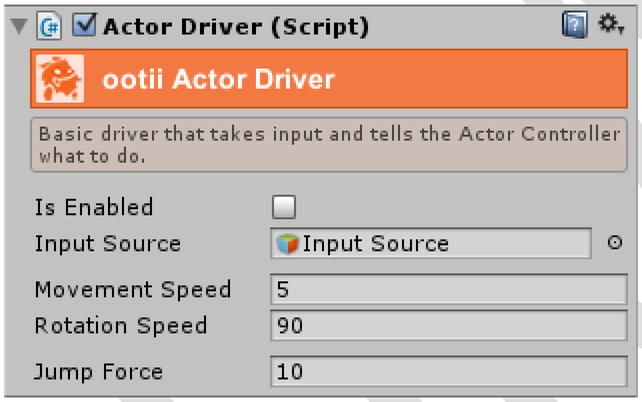
```
ActorController.Move()

ActorController.Rotate()

ActorController.AddImpulse()
```

There are several other functions you can use when creating your own driver, but the ones above are the basics.

Included in the package are several Actor Drivers that you can use:

Actor Driver – Assets/ootii/ActorController/Code/Actors/CharacterControllers

This is the default driver. It takes input from the keyboard, mouse, and Xbox controller and turns that into movement and rotation that is relative to the character's forward direction. It then calls the Actor Controller functions to actually move the actor.


Actor Driver (Script)

ootii Actor Driver

Basic driver that takes input and tells the Actor Controller what to do.

Is Enabled	☐
Input Source	Input Source
Movement Speed	5
Rotation Speed	90
Jump Force	10

Animator Driver – Assets/ootii/ActorController/Code/Actors/CharacterControllers/Drivers

Inherits from Actor Driver, but looks for a Unity Animator that is attached to the game object. If found, it will query for root-motion data and use that to move and rotate the character. If no root-motion data is found, input will be used to control the character.

Nav Mesh Driver – Assets/ootii/ActorController/Code/Actors/CharacterControllers/Drivers

Inherits from Animator Driver, but uses a Nav Mesh Agent to move the actor to a specific target. If an Animator is found, it will query for root-motion data and use that to move and rotate the character. If no root-motion data is found, speed will be set on the component.

Sphere Actor Driver – Assets/ootii/ActorController/Demos/Code

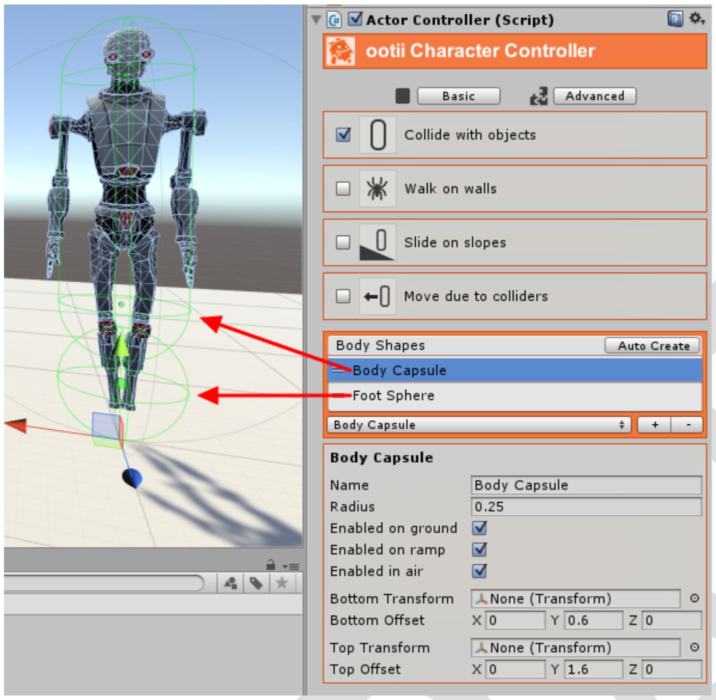
Inherits from Actor Driver. When an "inner" sphere is found it will use this the actor's body and rotate it based on the direction the character is moving. This give the impression that the sphere is rolling.

Spider Actor Driver – Assets/ootii/ActorController/Demos/Code

Inherits from Animator Driver. Similar to the other drivers, but when jump is pressed (and the actor is facing a wall), it will jump onto the wall so it can climb.

Body Shapes

Body shapes are used for collision detection by the Actor Controller. By using spheres and capsules, we can represent the shape of a human as well as other non-simple characters. We also have the ability to change these shapes during run-time in order to match the character's pose.



[✓] Collide with objects
[ ] Walk on walls
[ ] Slide on slopes
[ ] Move due to colliders

Body Capsule

Name	Body Capsule
Radius	0.25
Enabled on ground	✓
Enabled on ramp	✓
Enabled in air	✓
Bottom Transform	None (Transform)
Bottom Offset	X 0 Y 0.6 Z 0
Top Transform	None (Transform)
Top Offset	X 0 Y 1.6 Z 0

The default setup has two shapes:

Body Capsule – This capsule is similar to a traditional character controller, but is raised off the ground. This allows the actor to get right up to the edge of a step… which is great for foot IK.

Foot Sphere – This sphere is only active while in the air. This keeps our feet from entering things as we jump over objects. However, it doesn't block us when we move close to a ledge or step.

It's important to note that body shapes are NOT colliders. So, external raycasts or rigid-bodies won't react to them. If you want an external raycasts to hit your character, you can add traditional Unity colliders to your actor as you normally would.

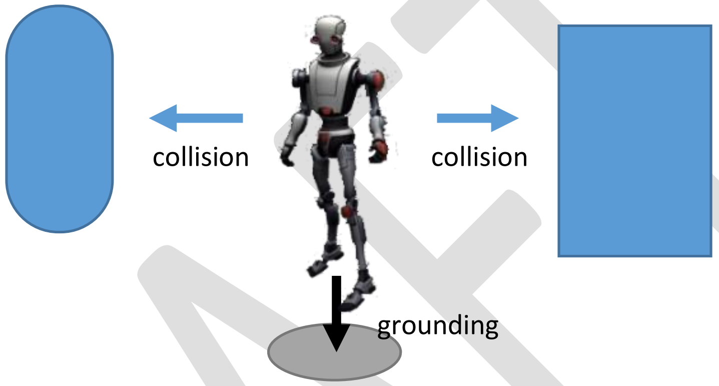
Collisions vs. Grounding

"Colliding" and "Ground" are two different things.

Colliding means collision detection has determined you've bumped into another object… say a wall.

Grounding means we've tested the ground directly under the actor and they can stand on it.

So, you can completely disable collisions and your character will still be able to walk on the ground, move up slopes, etc. There is a huge performance boost when you disable collisions. This is great for NPCs that may be on rails or have limited movement.

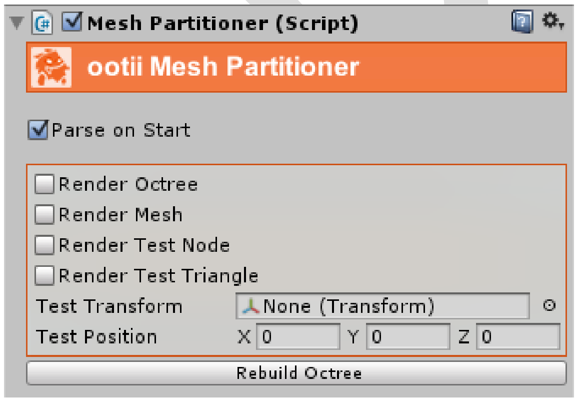


Colliders

Actor Controller is able to collide against standard unity colliders. The fastest colliders are sphere, box, and plane. However, it can also detect collisions against mesh colliders. However, like Unity's Character Controller, there's a performance impact when colliding against mesh colliders.

To help minimize the impact, we parse the mesh collider and store its information for future collisions. For small meshes, this can be done on first impact. However, for large meshes you'll want the Actor Controller to pre-process the mesh collider.

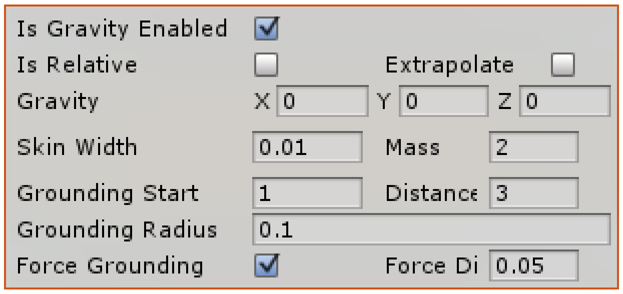
To do this, add an ootii Mesh Partitioner component to the game object that has a Mesh Collider object. Then, check 'Parse on Start'.


Mesh Partitioner (Script)
ootii Mesh Partitioner
☑ Parse on Start
☐ Render Octree
☐ Render Mesh
☐ Render Test Node
☐ Render Test Triangle
Test Transform: None (Transform)
Test Position: X 0 Y 0 Z 0

All the remaining options are for visualizing the partitions in the editor and should not be enable.

Actor Controller Advanced Settings

Grounding



Is Gravity Enabled	✓		
Is Relative	☐	Extrapolate	☐
Gravity	X 0 Y 0 Z 0		
Skin Width	0.01	Mass	2
Grounding Start	1	Distance	3
Grounding Radius	0.1		
Force Grounding	✓	Force Di	0.05

Is Gravity Enable
Determines if we use gravity or not.

Is Relative
Determines if gravity is based on the world orientation (Vector3.up) or the actor orientation (transform.up).

In order to walk-on-walls, this option must be set.

Extrapolate Physics
The Actor Controller uses the LateUpdate() function for all its processing. However, physics based information (like gravity and forces for jumps) are processed in FixedUpdate() in order to be consistent across frame-rates. These two functions aren't always in synch.

If you notice stuttering while falling, checking this options will smooth that out.

Gravity
Determines the force and direction of gravity. If no value is set, we use Unity's gravity value.

The final gravity direction will be determined by the **Is Relative** flag.

Skin Width
Small amount of room allowed between the body shapes of the actor and objects they collide with or ground to.

Mass
Mass of the actor. We use this value when applying external forces. We use Unity's standard which is roughly 1 unit = 1 cube = ~35 kg.

Grounding Start
When shooting the ray for grounding, this is the height above the character's root that we'll start shooting the ray downward.

Grounding Distance
Determines how far below the character we'll shoot the grounding ray.

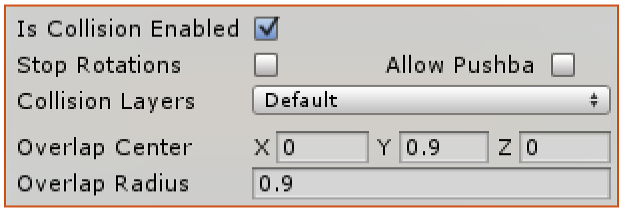
Grounding Radius
In addition to a ray, a sphere is cast downward in order to help prevent falling into gaps. This value should be the radius of the "feet" or bottom of the actor.

Force Grounding

Determines if we'll push the character down to the ground if they are within the **Force Distance** range.

Force Distance
Maximum distance the character can be from the ground and we can still force them too it when **Force Grounding** is checked.

Collisions



Is Collision Enabled	✓
Stop Rotations	☐ Allow Pushba ☐
Collision Layers	Default
Overlap Center	X 0 Y 0.9 Z 0
Overlap Radius	0.9

Is Collision Enabled
Determines if we'll test for collisions

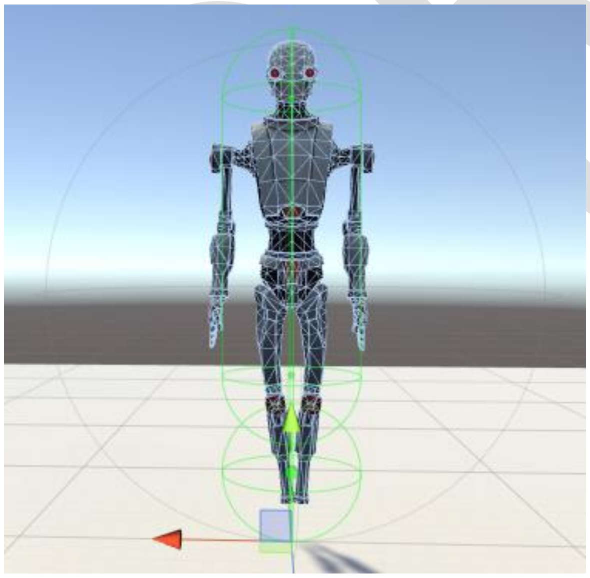
Stop Rotations
Determines if collisions will stop the character from rotating

Allow Pushbacks
Determines external colliders will cause the actor to move.

Assume the actor is just standing idle, but a collider is moving to the character. This object will allow the collider to push the character back in order to keep the character from penetrating it.

However, like the Unity Character Controller, the collider could push the character into another collider. So, care should be taken with this.

There is also a performance penalty for using this option. Use it sparingly.

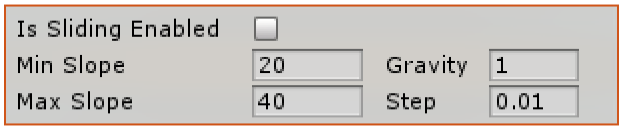


Overlap Center
When performing collision detection, we first see if there are objects within the range of the character. This is the center of that range.

Overlap Radius
When performing collision detection, we first see if there are objects within the range of the character. This is the radius we use.

In the editor, you'll see a light gray sphere representing the overlap circle that will be used for collision tests. Use the "Overlap" settings to ensure the majority of your character is covered.

Sliding



Is Sliding Enabled	☐		
Min Slope	20	Gravity	1
Max Slope	40	Step	0.01

Is Sliding Enabled
Determines if the actor will slide when on a ramp whose angle is greater than the **Min Slope** angle.

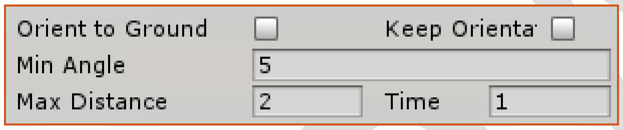
Min Slope
Minimum angle the slope (under the actor) needs to be before the actor starts to slide.

Gravity
When sliding, the percentage of gravity that will be applied to the slide. This helps to fake friction on the surface.

Max Slope
The maximum slope an actor can move up. If a slope is greater than this, the character will simply stop moving.

Step
Small distance used internally to find the closes point to where the slope actually starts. This value typically doesn't need to be changed

Orientation



Orient to Ground	☐	Keep Orienta	☐
Min Angle	5		
Max Distance	2	Time	1

These features are important when a character is mean to walk on walls or ceilings. In order to do this, the character must orient to the surface they are walking on.

Orient to Ground
Determines if the actor will change its "up" vector to match the normal of the surface he is standing on.

In order to walk-on-walls, this option must be set.

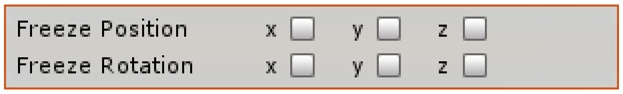
Keep Orientation
Determines if the actor will keep the last orientation it had while jumping. When not checked, the character will return to the natural orientation (Vector3.up) while jumping.

Min Angle
Minimum angle change that will cause an orientation change.

Max Distance
The maximum from the ground before the actor's orientation will start to change to the natural ground (Vector3.up).

Time
The number of seconds it takes to reach the new orientation.

Freezing



Freeze Position x ☐ y ☐ z ☐
Freeze Rotation x ☐ y ☐ z ☐

Setting these options allow you to limit the movement and rotation of the specified axis.

Support

If you have any comments, questions, or issues, please don’t hesitate to email me at support@ootii.com. I’ll help any way I can.

Thanks!

Tim