



Table of contents

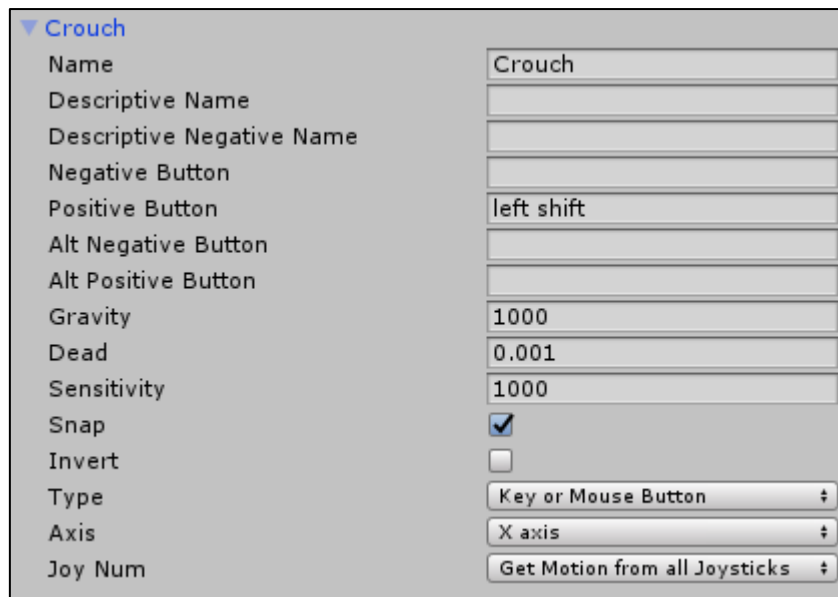
1 Getting started	2
1.1 Creating the needed inputs.....	2
1.2 Creating the “Planet” layer	2
1.3 Adjusting the project settings	2
1.4 Creating a minimalistic scene.....	2
1.5 Attaching the scripts.....	3
1.6 Adding animations.....	3
2 Mechanics explanations	3
3 Variable explanations	3
4 Helpful tools	3
5 Version history	4
6 Got questions or problems?	4

1 Getting started

This chapter covers everything you need to do for running the provided demos or creating your own minimalistic scene.

1.1 Creating the needed inputs

Set up the following inputs in the Project Input Settings (Edit > Project Settings > Input Manager) if you want to use Unity's input system. The used buttons shown in the pictures are the same as in the demo – of course you can use whatever buttons you want. You can also implement a custom input management system by adjusting the `GetInputs()` method which can be found in each script with input handling.



The screenshot shows the 'Crouch' input configuration in the Unity Project Input Settings. The settings are as follows:

Property	Value
Name	Crouch
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	left shift
Alt Negative Button	
Alt Positive Button	
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input checked="" type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button
Axis	X axis
Joy Num	Get Motion from all Joysticks

1.2 Creating the “Planet” layer

Click on any game object in the scene and navigate via its layer to the dropdown menu option “Add layer...”. Add a layer with name “Planet”.

Hint: If this is not the first custom layer you have added to your project, you have to assign the “Planet” layer manually in the demo scenes to make them work. As an example for the first demo scene, search for “Cube” and “Sphere” objects in the hierarchy window which are *not* part of an asset, i.e. font color is in white, and assign the “Planet” layer.

1.3 Adjusting the project settings

Navigate to Edit > Project Settings > Physics and set the gravity vector to (0,0,0).

Now you should already be good to go to run the demos.

1.4 Creating a minimalistic scene

Go to folder Planet Platformer Controller > Prefabs and drag a first ground object (Planet or Platform) for your character to stand on into the scene. This object already includes a default surface, a Gravitational Field and a Camera Settings object. See section 2 for more information on their function.

Now place your character object on the surface of the chosen ground and rotate it so that its feet point to the ground. If your character does not have a Rigidbody component attached yet, this is the time to add it. Also, tag the object as “Player”.

1.5 Attaching the scripts

The PPMotor script is the most essential script of this asset. It provides the movement of the game object, including all applied physics and pose calculations. The following scripts are optional depending on what you additionally need:

- Attach the PPController script to your character game object if you would like to have the character controls. Alternatively for NPC game objects, you can attach the very basic NPC controller script PPNpcAI.
- Attach the PPCamera script if you want to use the provided camera mechanics described in section 2.3.

1.6 Adding animations

Just add one of the provided animator components in folder “Animator Controllers” to your character or NPC game object, respectively, and assign the desired animations to their corresponding states in the animator tab. The animations provided with the asset are only dummy animations. The originals are from [Rheedo Animations](#). They perfectly fit to the setting and offer more than needed. Their names are the same as the names of the dummy animations so that you can easily map them.

Note: If you want to use your own animator controller or animations, animation events have to be set up. Whenever methods Attack or Damage are called inside the PPMotor script, the movement of the corresponding object is disabled (via method DisableMovement). Inside method Attack, this prevents movement during the attack animation, whereas inside method Damage, this prevents movement when the object is damaged, e.g. pushed. Therefore, method EnableMovement of the PP Motor has to be called by an animation event of the corresponding animation to enable movement again. Two examples: 1) The attack animation of the NPC calls EnableMovement at the end to prevent the NPC from instantly turning around after throwing the target object. 2) The getting up animation of the character (played after being pushed or thrown) calls EnableMovement to allow movement again after the character landed and stood up.

2 Mechanics explanations

Not included in the public version of this manual.

3 Variable explanations

Not included in the public version of this manual.

4 Helpful tools

Not included in the public version of this manual.

5 Version history

If you are interested in the release notes, please refer to my website johnstairs.com/ppc.

6 Got questions or problems?

Feel free to send me an email to mail@johnstairs.com if your question is not covered by the FAQs on johnstairs.com/ppc! Please attach a screenshot showing your scene, the used variable values and occurring error messages.

Best regards,

John Stairs