

## CN Controls overview and installation guide

This overview contains all information on how to use CN Controls, some things you might want to use something you need to beware of. Please carefully read at least through the intro part.

### Contents:

- [Intro](#)
- [Specific info about each control](#)
- [Scripting usage](#)
- [Common questions](#)
- [Contacts](#)
- [Free joystick skins](#)

## Intro

This section covers the package contents, added menu items and common setup requirements.

### *Contents of the package*

In the *CNControls* folder you'll see the following folders:

- *Editor* – contains all menu items and custom inspectors
- *NotImportant* – contains stuff that can be safely deleted (demo scenes, materials and textures for these demo scenes, example controller scripts)
- *Prefabs* – contains prefabs for Joystick, Touchpad, Throwable touchpad and a CNControlCamera
- *Scripts* – contains scripts that make all the controls work

### *Added menu items*

After you import this package, you'll have these menu items added:

- *GameObject* ► *Create Other* ► *CNControls* ► ... (Fig.1.)
- In the Hierarchy view – *Create* ► *CNControls* ► ... (Fig.2.)

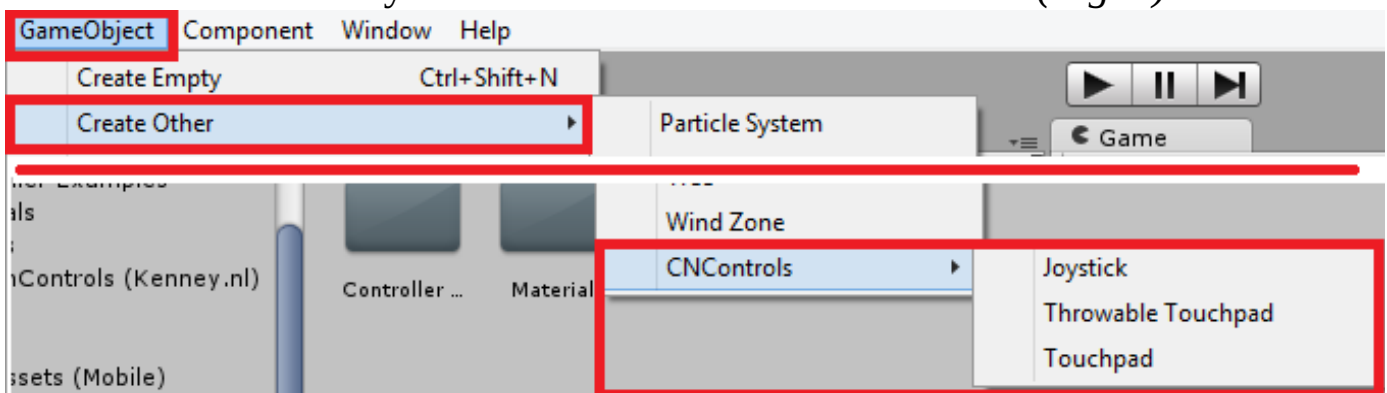


Fig.1. Added menu item

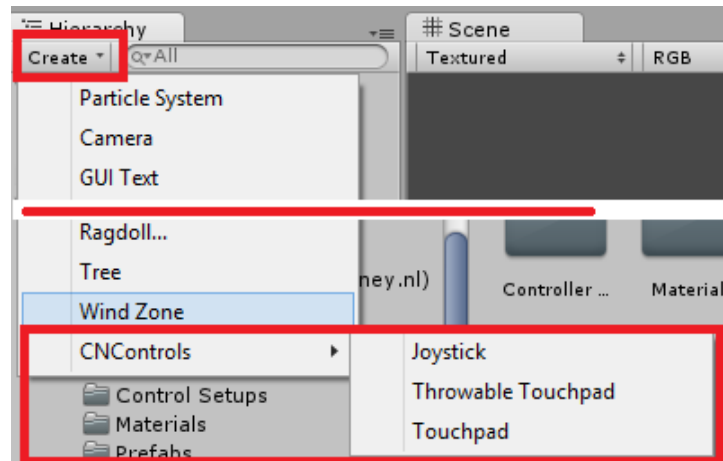


Fig.2. Added menu item

It will allow you to easily create new controls in the Scene view.

All controls are automatically added to the *CNControlCamera*. It will be created automatically the first time you create a new control through this menu. **Please note** that this camera assumes that all the controls belong to the *CNUI* layer. You should also make sure that your cameras don't have this *CNUI* layer in their *Culling Mask*, or it will eventually render the controls somewhere in the scene. These controls should only be rendered with a *CNControlCamera*.

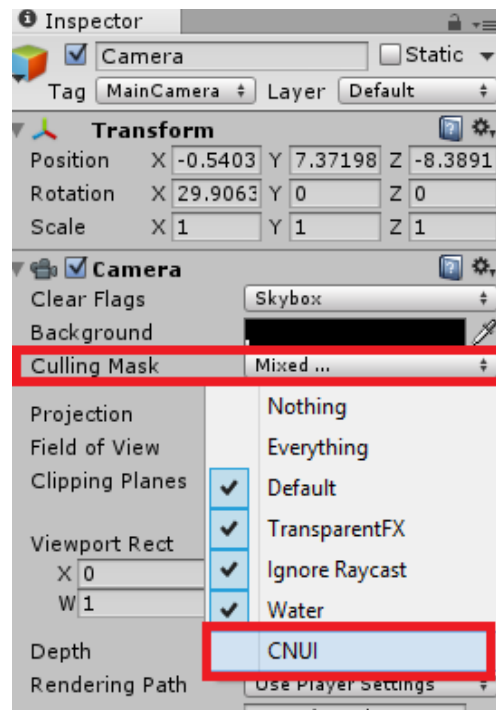


Fig.3. Make sure your main cameras don't have *CNUI* layer in their *Culling masks*

If you're using this pack with the new uGUI system, you can delete this layer and use a default *UI* layer instead.

When you create any control, it will appear inside the *CNControlCamera*. You can then reference it from any script just like you would do with any other object.

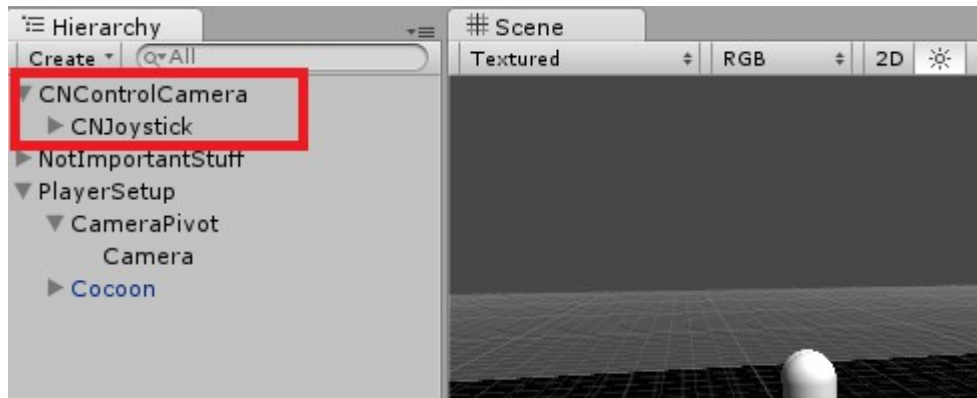


Fig.4. Created control inside it's camera

More controls will appear as children of the same camera.

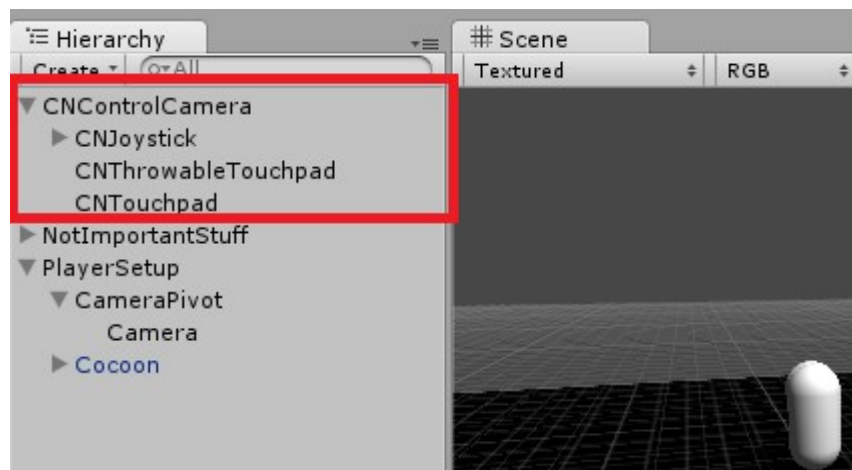


Fig.5. All your base controls are belong to us

## Specific info about each control

This pack currently consists of three controls:

- [Joystick](#)
- [Touchpad](#)
- [Throwable touchpad](#)

### Joystick

Joystick is the most common control. You can find an example inside the *3DExample* scene in the *CNControls/NotImportant/Scenes/3DExample*

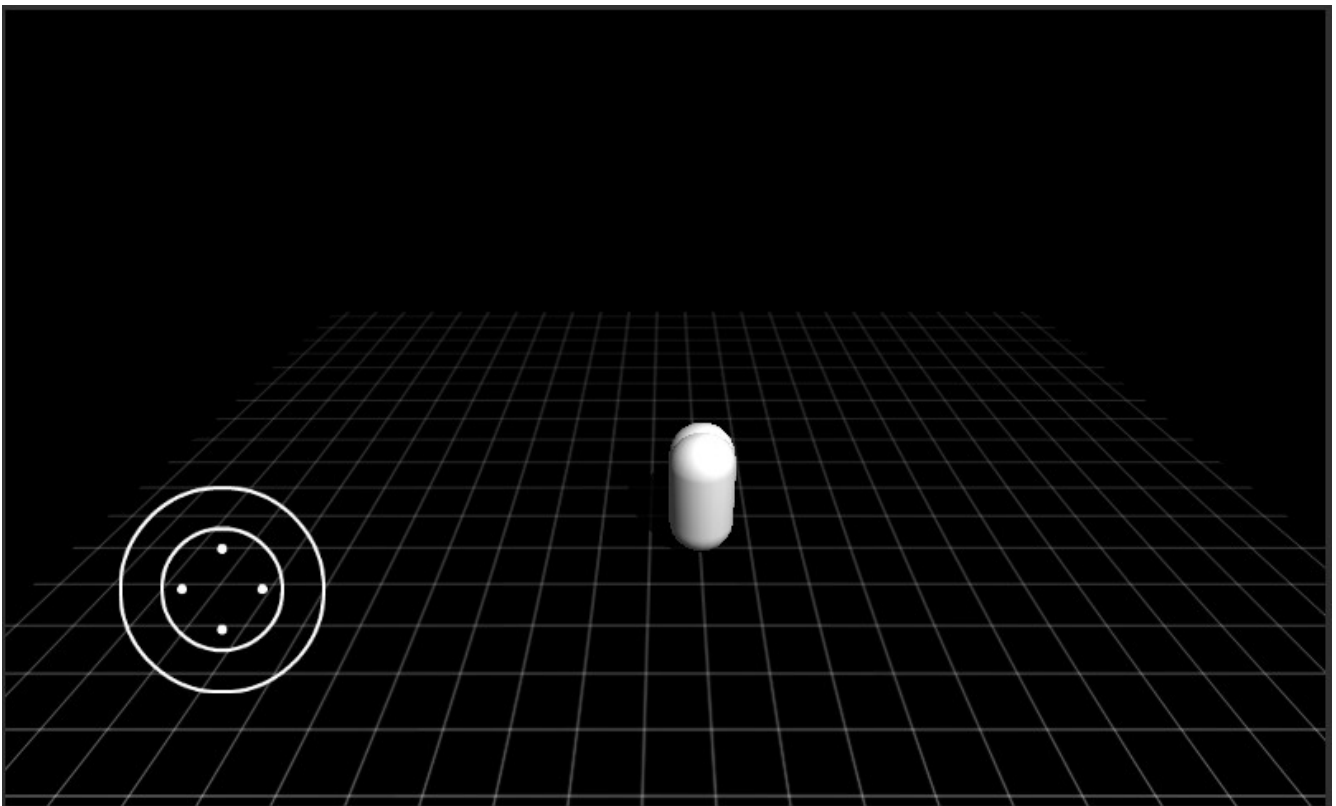


Fig.6. Joystick control inside the 3DExample scene

## Joystick settings

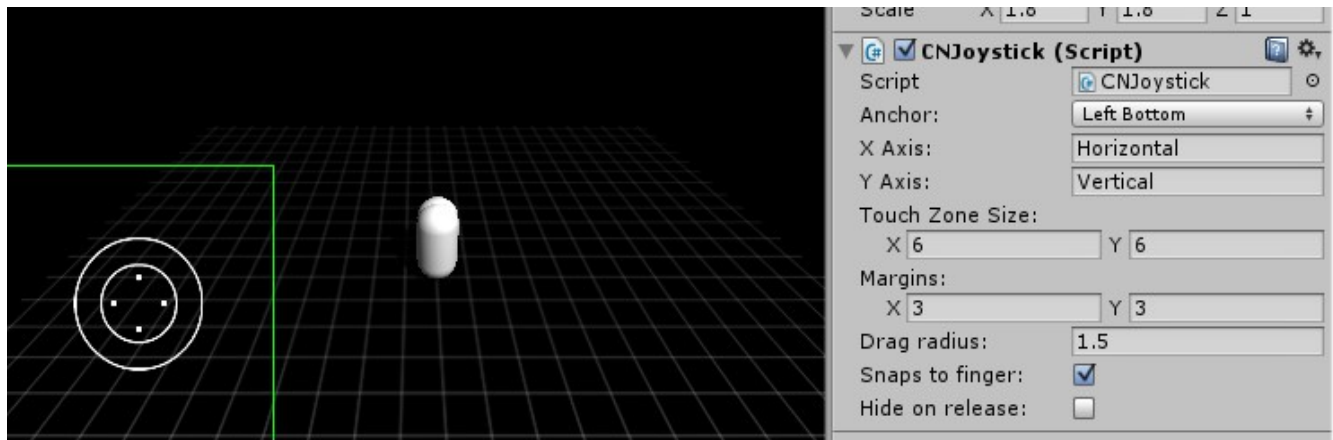


Fig.7. Joystick settings and visible green gizmo representing a touch zone

*Anchor:* Current anchor relative to the screen. Possible values are:

- Left Bottom
- Right Bottom
- Left Top
- Right Top

*X Axis:* The name of the X axis of the joystick. Used when querying the control with a ***GetAxis(string axisName)*** method.

*Y Axis:* The name of the X axis of the joystick. Used when querying the control with a ***GetAxis(string axisName)*** method.

*Touch Zone Size:* That's what it says – it's the size of the touch zone. The touch zone itself is visually represented as a green rectangle in the *Scene* or *Game* view (to see it in game view you should enable the *Gizmos* options at the right top corner of the *Game* view window)

*Margins:* The distance to the edges of the screen **in units**. It's resolution-independent.

*Drag radius:* Maximum distance on which a user can drag the stick from the base. Measured in units, resolution-independent.

*Snap to finger:* Indicates whether the joystick will jump under the finger when a user touches inside the touch zone.

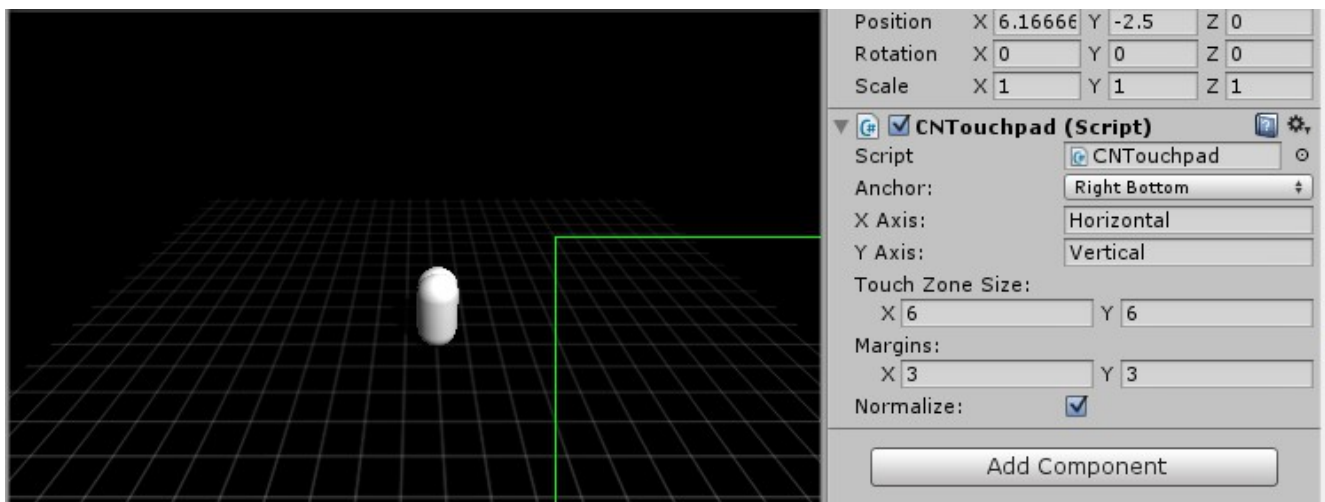
*Hide on release:* Indicates whether the joystick will be hidden if not being currently used. Just like in the Shadowgun or Dead Trigger 2 games.

## Touchpad

Touchpad is a control based on delta movements. Unlike joystick, it has no base and delta movement is calculated based on the previous finger position. It's much like the camera controller in Shadowgun or Dead Trigger 2.

It has no visible components by default, but you can easily add a sprite renderer to it or any other visual component, it won't break the control.

## Touchpad settings



*Anchor:* Current anchor relative to the screen. Possible values are:

- Left Bottom
- Right Bottom
- Left Top
- Right Top

*X Axis*: The name of the X axis of the touchpad. Used when querying the control with a ***GetAxis(string axisName)*** method.

*Y Axis*: The name of the X axis of the touchpad. Used when querying the control with a ***GetAxis(string axisName)*** method.

*Touch Zone Size*: That's what it says – it's the size of the touch zone. The touch zone itself is visually represented as a green rectangle in the *Scene* or *Game* view (to see it in game view you should enable the *Gizmos* options at the bottom top of the *Game* view window)

*Margins*: The distance to the edges of the screen **in units**. It's resolution-independent.

*Normalize*: Should the input be normalized? If true, it will not take users "pressure" or drag speed on the control, every drag will be of the same magnitude. Use when you want to fully control the input speed of the control. If false, though, the control will feel more responsive and axis values will depend on how gently (or roughly) the user will use the control. Use when you want to give the user more precise control.

## **Throwable touchpad**

Throwable touchpad is a cool extension of the original touchpad. It's much like a mobile scrollable list. You can scroll it once and watch it continue scrolling with gradually decreasing speed.

## **Throwable touchpad settings**

Same as [touchpad settings](#) with one additional item:

*Speed decay*: Rate at which its speed decays in [0; 1] range. 1 means it will never decay, 0 means it will immediately decay (it becomes an original touchpad control)



## Scripting usage

You can find an example character and camera controller scripts at the *CNControls/NotImportant/ControllerExamples* folder.

All controls provide two different methods of getting the user input:

1. Event system
2. **GetAxis**(string **axisName**) method

Event system uses a C# events feature. Every script has three events which you can subscribe to:

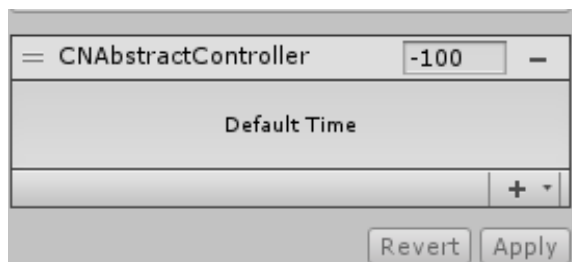
- *Action*<**Vector3**, *CNAbstractController*> **ControllerMovedEvent**
- *Action*<*CNAbstractController*> **FingerTouchedEvent**
- *Action*<*CNAbstractController*> **FingerLiftedEvent**

*ControllerMovedEvent* fires when a user uses a joystick. It passes a **Vector3** parameter, which represents the Input, and a reference to the control which is being tweaked (which has fired the event).

*FingerTouchedEvent* fires when the user touches the controller, making it active. Useful for launching some starting animations.

*FingerLiftedEvent* fires when the user lifts the finger from the touch, making it inactive. Useful for launching some ending animations.

You can also use a more common method of getting the input using a **GetAxis**(string **axisName**) method. You can just call this method on any of your controls and get a **float** value back. **Note though**, that if you don't specify the *Script Execution Order* for the *CNAbstractController* as negative, you may have a **one-frame-late input**. To set the script execution order, go to *Edit* → *Project Settings* → *Script Execution Order* and set the ***CNAbstractController*** script to be the first one in list (smallest negative value) so it appears before the **Default Time** and **before any other script**:



## Common questions

Q. Why isn't it using a new awesome uGUI system?

A. There's just no point using it, it won't give any advantage. uGUI uses raycasting system for getting the user touches, which is slower than calculating rects that is used in these controls.

Q. Can I get the source code of these controls?

A. These controls are actually consist of different .cs files, so you can tweak and edit them as you would like.

Q. Is there a github repository for this?

A. Yes, you can grab the latest update at the <https://github.com/KumoKairo/CNJoystick>

Q. I don't really get the math behind the code, do you have any explanation?

A. Yes, you can check my blog at the <http://blog.nadezhdin.org/>

Q. I have an issue, can I ask you a few questions?

A. Yes, there's a [section](#) that has all the contacts.

## Contacts

You can drop me a line at the [cyrill@nadezhdin.org](mailto:cyrill@nadezhdin.org)

Find me on twitter @KumoKairo

Or visit my blog at <http://blog.nadezhdin.org/>

You can always get the latest update of this pack at the <https://github.com/KumoKairo/CNJoystick> (you're more than welcome to contribute. Just try to follow the same coding conventions)

## Free joystick skins

You can get free joystick skins pack at

<http://opengameart.org/content/onscreen-controls-8-styles>

It has a lot of different styles. I didn't include it in the pack to keep it nice and clean.

You can also check my other pack with different joystick skins at the assetstore. Unlike the first one, it costs 2\$, but it's a great way of supporting this project:

<http://u3d.as/87s>