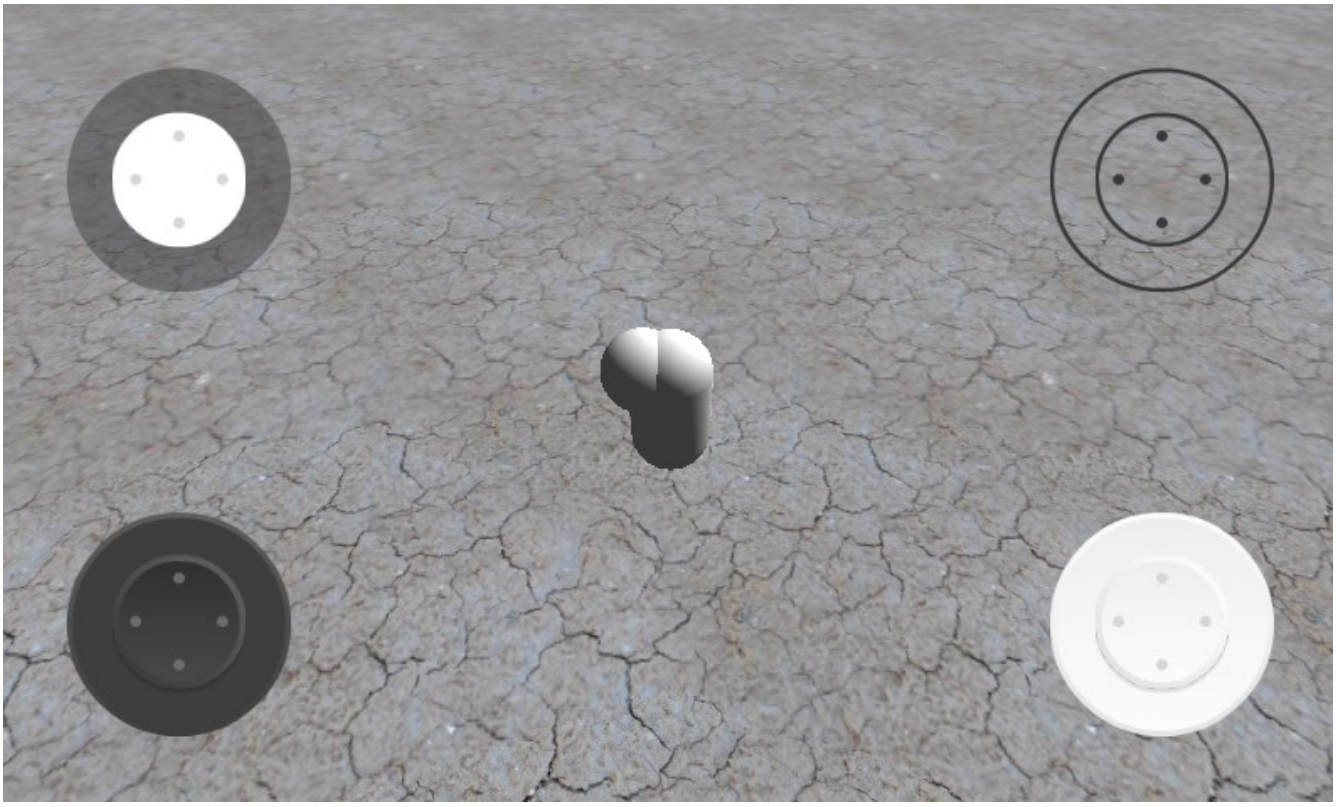
Mobile CNJoystick (v 0.1)



Contents:

- Abstract
- Installation guide
- CNJoytsick API
- Documentation, mechanics explained (educational or contribution purposes)

This package contains joystick screen controls made by an amazing artist Kenney

Abstract

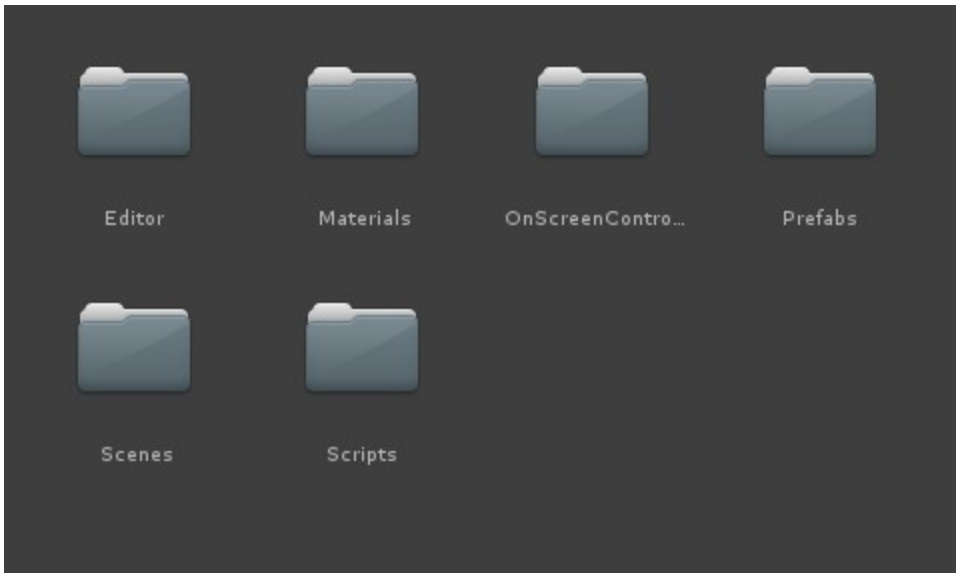
This package contains ready-to-use high performance mobile joystick, 8 different joystick styles, setup guide and detailed explanation of what's actually going on.

CNJoystick takes advantage of Unity3D *SpriteRenderer's* batching system, so if you have several controls on the screen from the same sprite atlas, it will batch to one drawcall. It also uses *Physics.Raycast(..)* only to capture the initial finger position, after that it takes screen coordinates of the captured finger.

So let's get started.

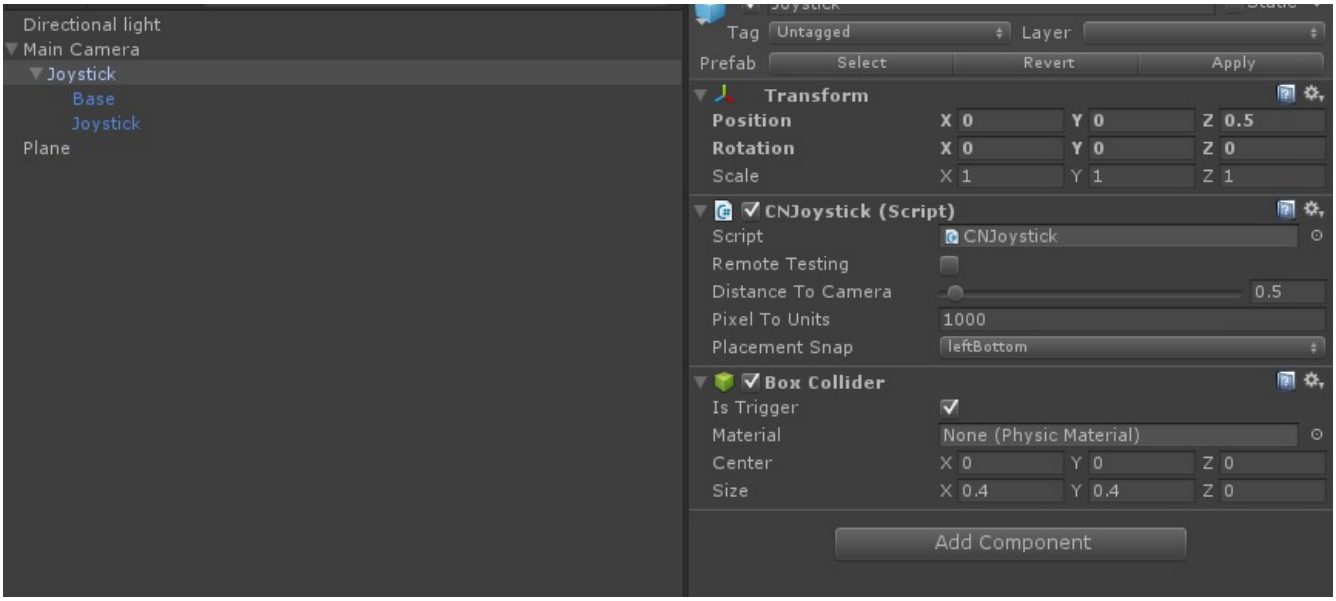
Installation guide

When you import your project, take a look at the contents of the CNJoystick folder:



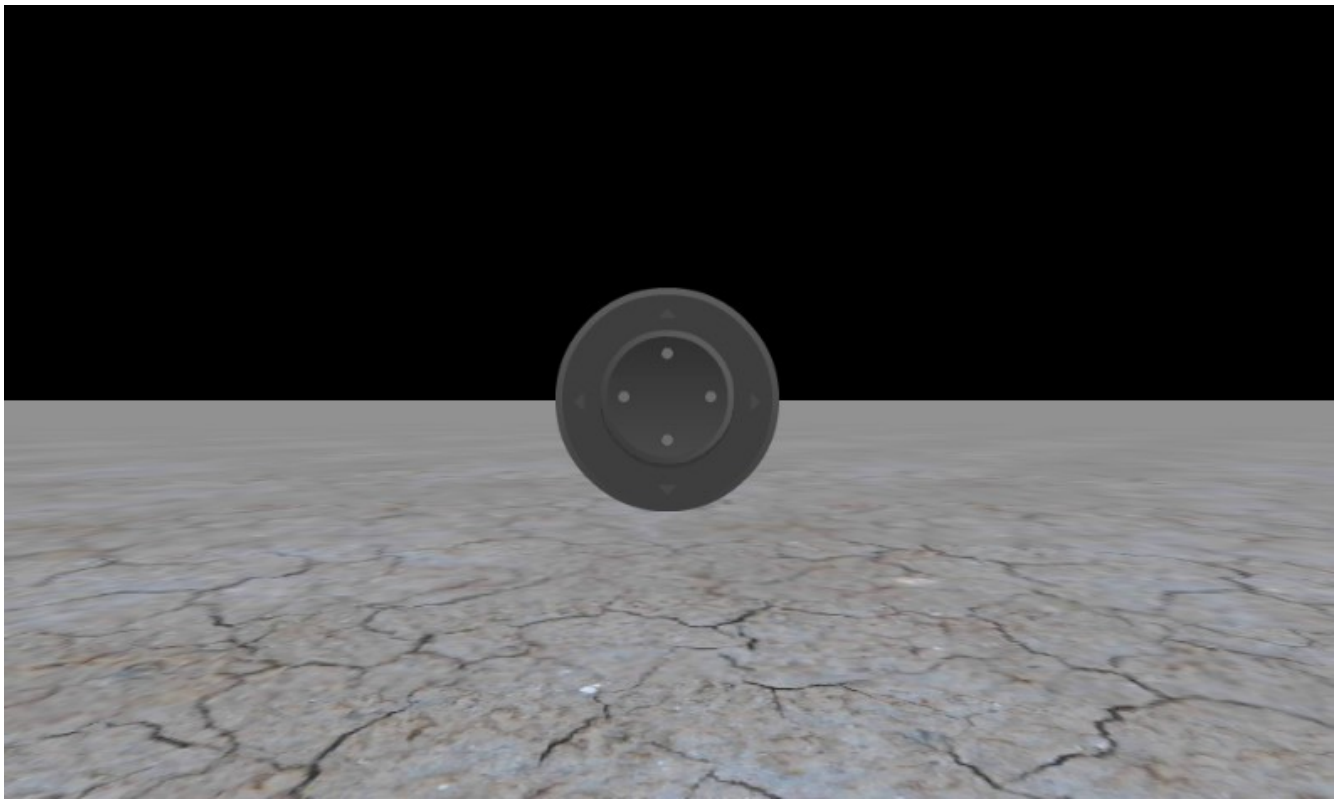
- **OnScreenControls** folder contains 8 different ready-to-use sprite atlases of 8 different joystick styles. It also contains original Kenney's On Screen Controls package so you can make sprite atlases of your choice.
- **Prefabs** – joystick prefab itself and a demo “character”
- **Scenes** – a demo scene
- **Scripts** – contains source of all used scripts

To use joystick, drag the *Joystick* prefab onto your **main camera**:

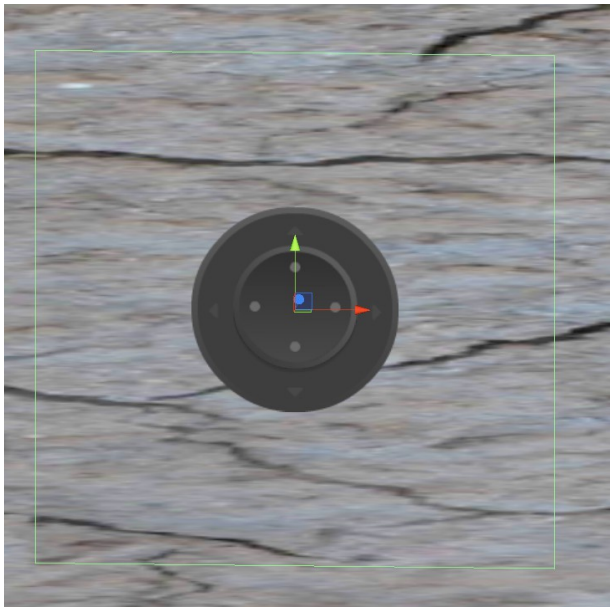


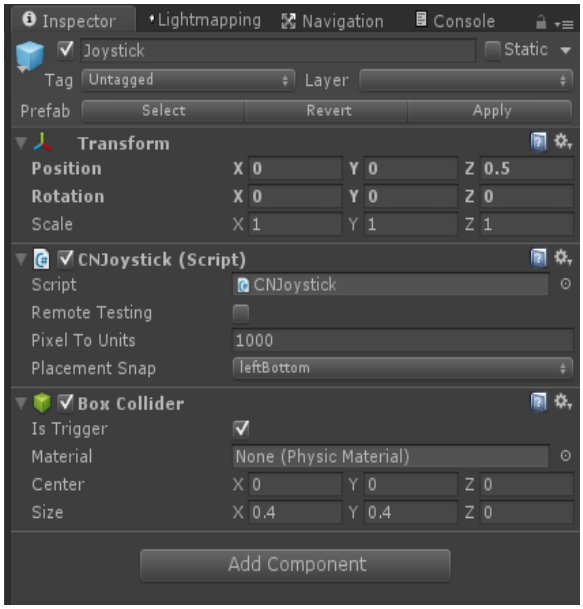
Notice that it's local positions are (0, 0, 0.5). Z position represents joystick's distance to the camera, and therefore it's size.

Take a look at the view of your camera now, it should look something like this:



Now look at the joystick itself, it has a Box Collider on it

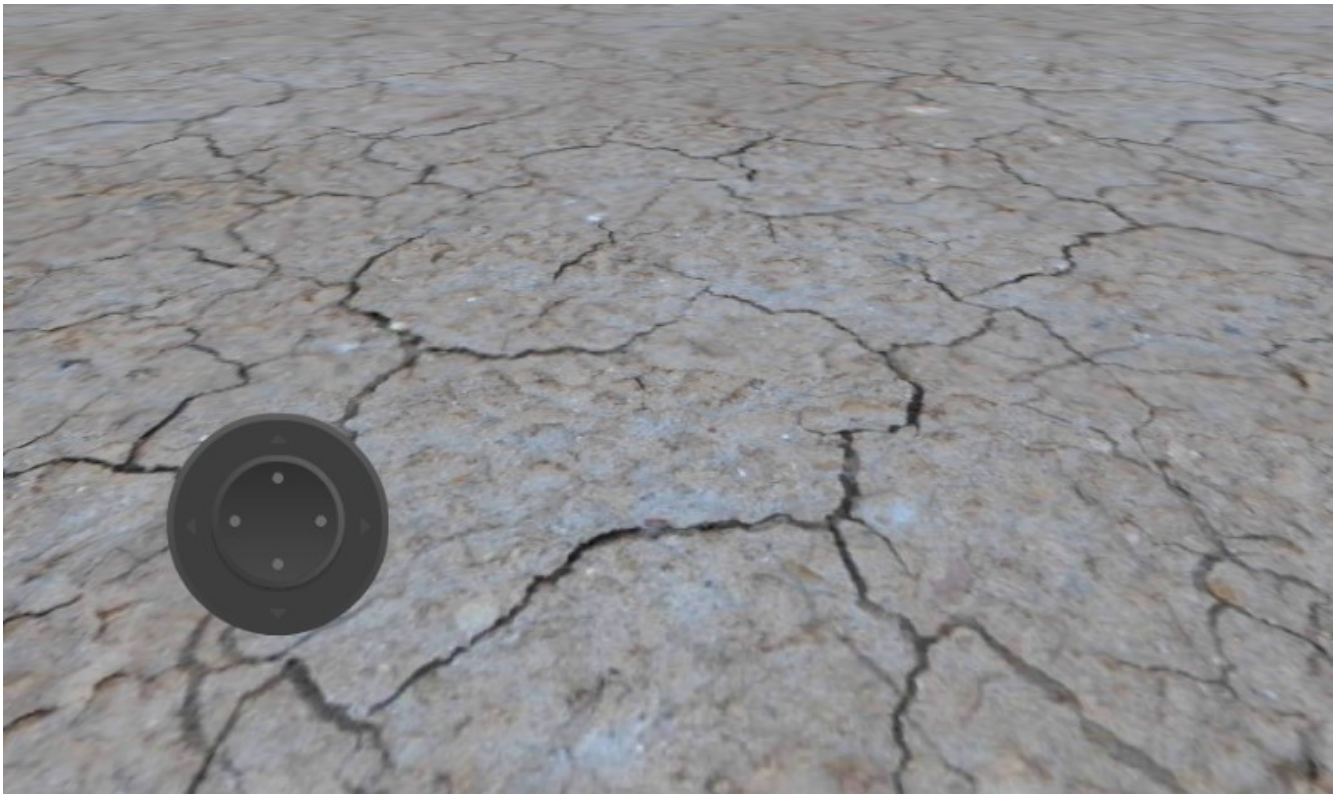




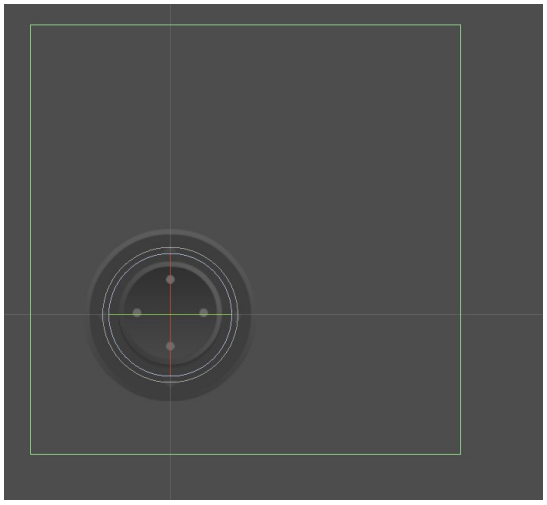
Properties:

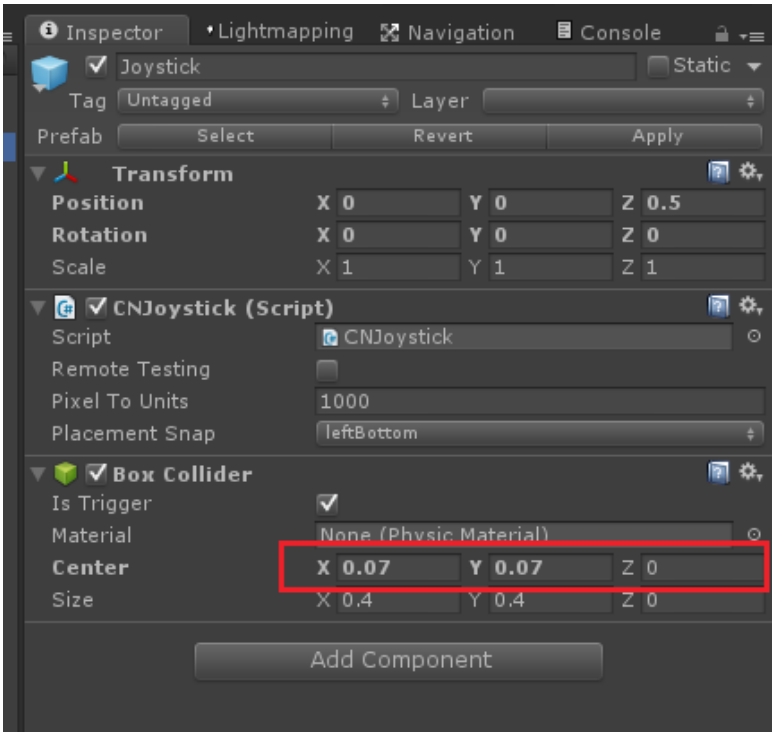
- **Remote Testing** represents whether you are testing your game using **Unity Remote** app or not. If you're going to use mouse cursor to test your game in unity, uncheck it. **However,** if you're going to use this Unity Remote app, check this button. **Otherwise, Unity will treat your touches as a mouse cursor!** Final mobile builds, however, will always use touches as input. Just keep in mind, that you have two choices of testing your game.
- **Pixel to Units** tells the script your sprite Pixel to Units value. **It should be the same as the Sprite import settings.** I didn't find any smart way to extract this value at runtime, so you'll have to specify it manually. It's 1000 by default and you wouldn't probably change it if you're going to use one of the 8 included styles.
- **Placement snap** represents the corner to which your joystick would snap when you hit play. It snaps with the borders of the **Box Collider,** so relative position of the box collider does matter. It's Left Bottom by default.

Now try running your scene in the editor. If you left Remote Testing propery unchecked, you'll be able to tweak the joystick with your mouse. Note that it snaps to your click position.

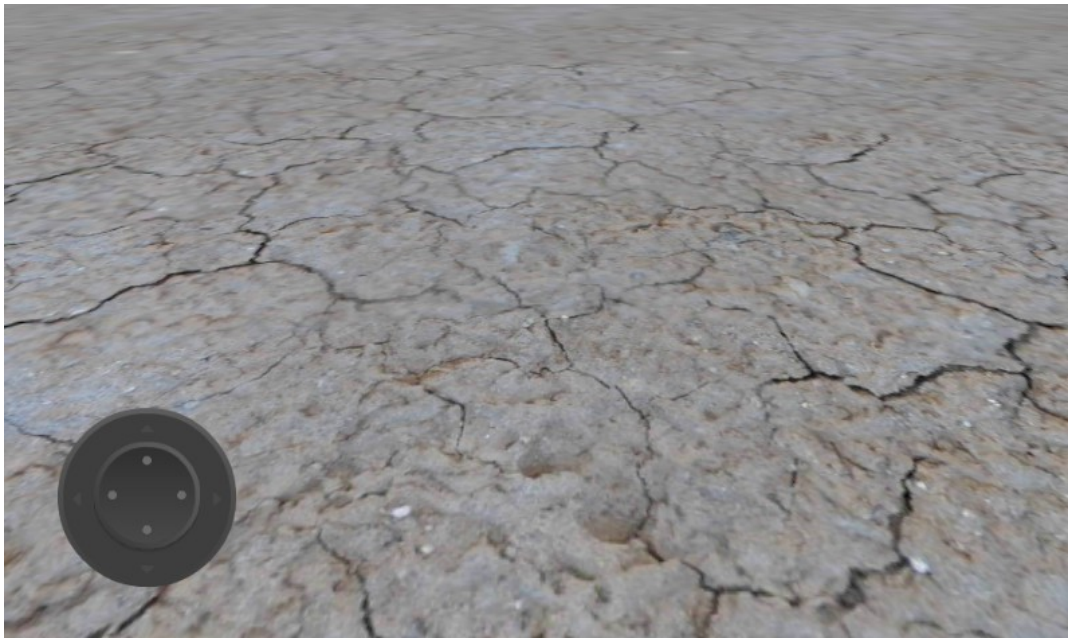


If you want to change it's relative position, you can change it's Box Collider properties like Center or Size. Let's say we want the joystick to be closer to the Left Bottom corner of the screen. We can do that by setting our Box Collider center to (0.07, 0.07)





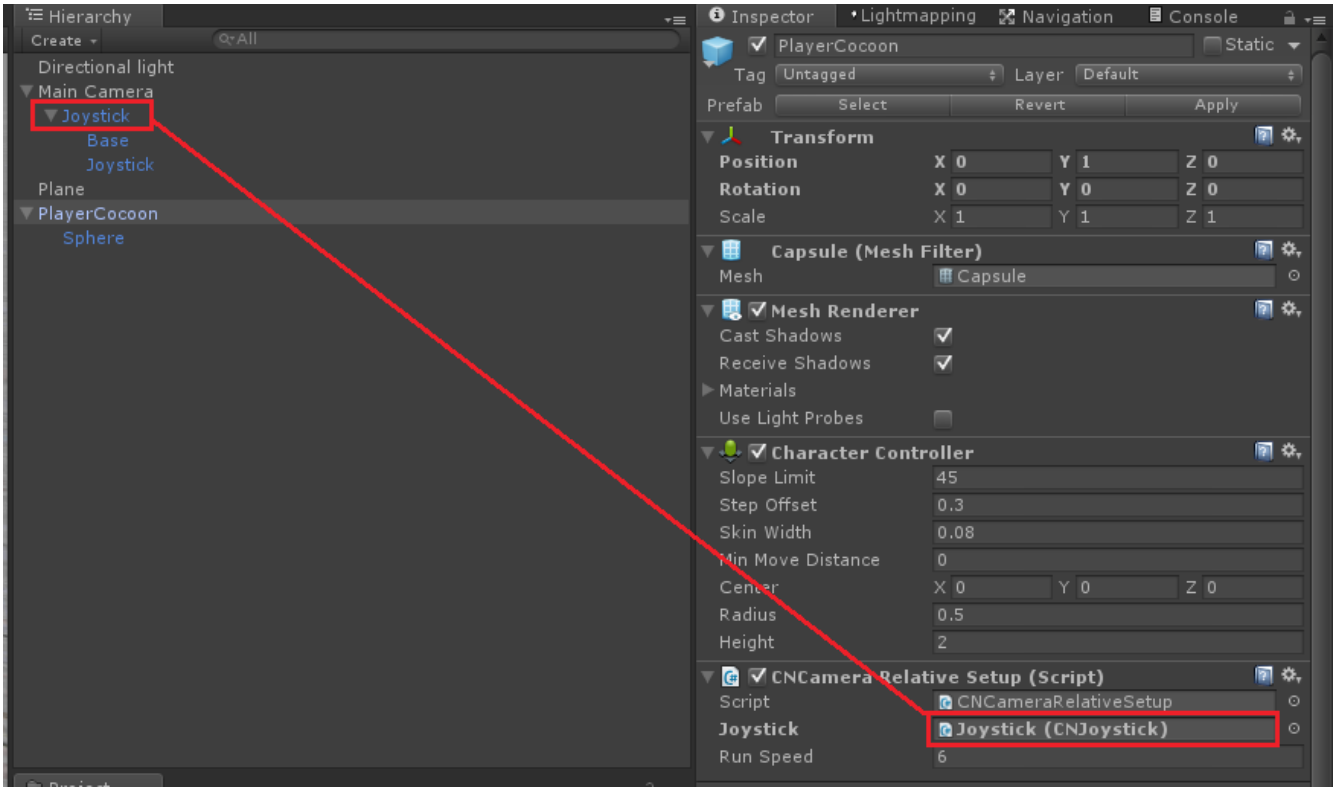
As you can see, joystick relatively moved to the left bottom corner. Now play the scene again.



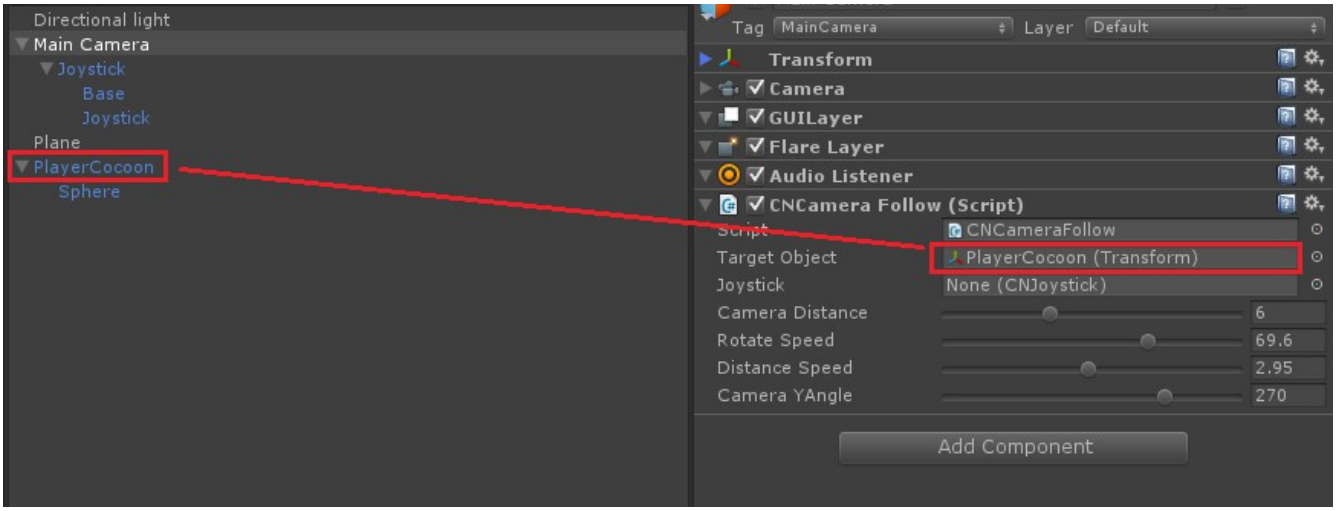
It's now closer to the border.

Let's get to character controls.
First, make a Plane for your character to run on if there's none yet. Second, drag *PlayerCocoon* prefab to the scene view and place it somewhere above the surface of the floor.

Note that it's already contains *CNCameraRelativeSetup* and *CharacterController* components. If you're going to make your custom character move, just drop this script to it's game object on it and it will automatically add a *CharacterController to it.*
Now drag your joystick to the *Joystick* slot in the *CNCameraRelativeSetup* component.



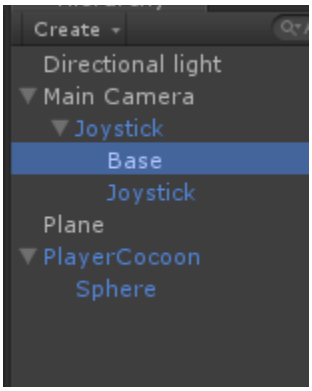
That's it, your character is already controllable. Yet your camera is not yet following it. Drag *CNCameraFollow* script to your camera and link your *PlayerCocoon* to it's *TargetObject* property.

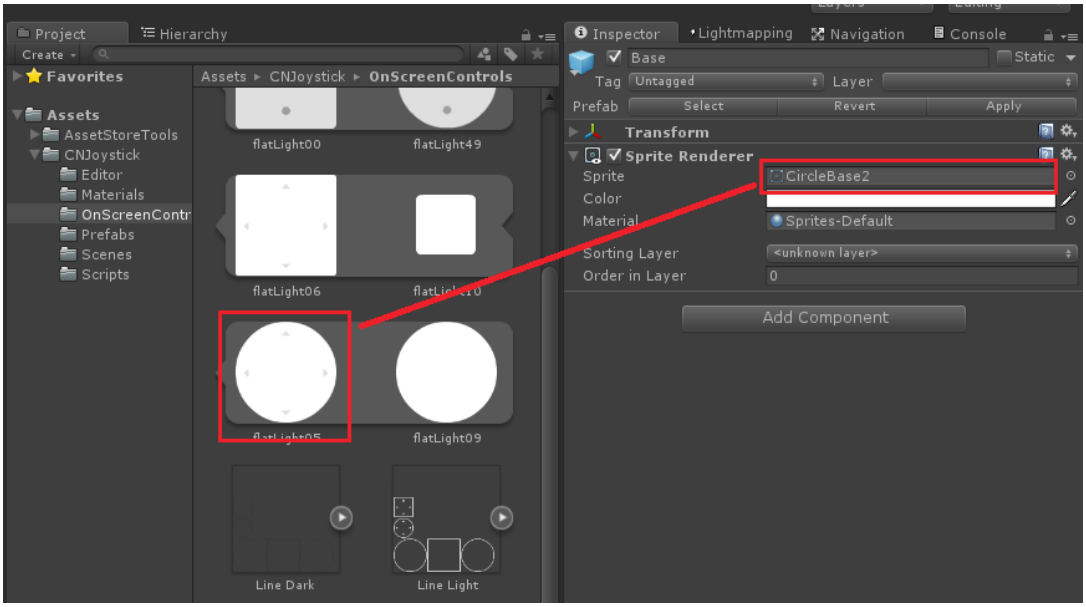


As you can see, *CNCameraFollow* has a joystick slot. It's not a mandatory but if you link any joystick to it, you'll be able to control the camera.

Try running your scene, you should be able to control your character.
You can also create a second joystick and control your camera with it.

If you want to change joystick style, just drag any sprite from OnScreenControls to the *Base* and *Joystick SpriteRenderers*

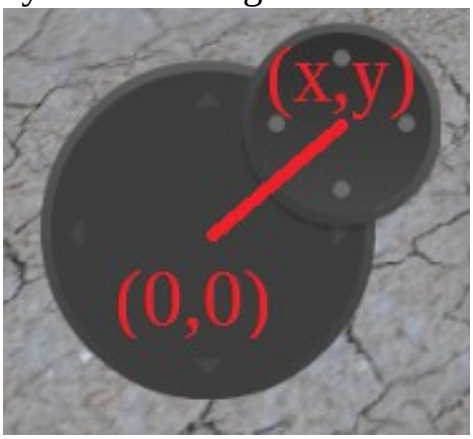




CNJoytsick API

If you're going to use *CNJoystick* with your custom character controller scripts you can use *CNJoystick's* public Events:

- *JoystickMovedEvent* is called each frame if player is currently tweaking the joystick. It has a Vector3 parameter which represents relative position of the joystick. It's magnitude will always be between 0 and 1, including 0 and 1.

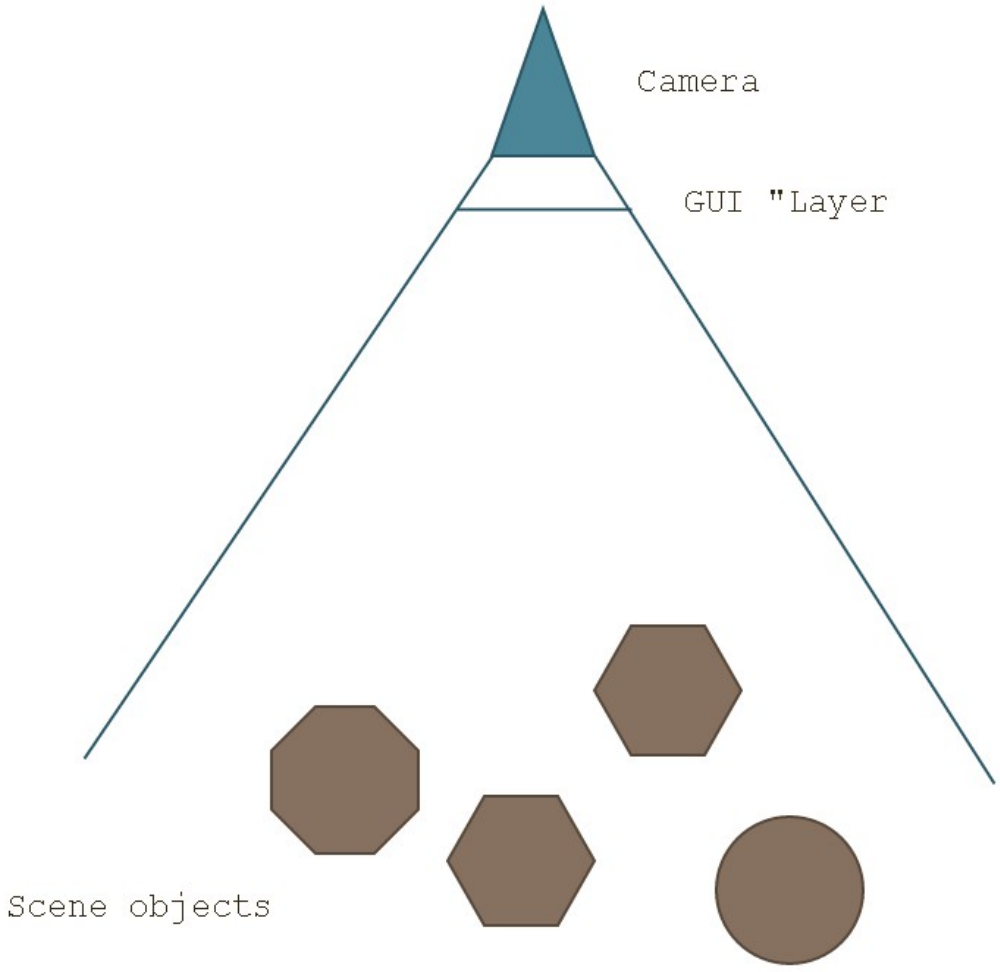


- *FingerLiftedEvent* is called once when user removes finger from the joystick. Useful for stopping your character.
- *FingerTouchedEvent* is called once when player begins tweaking the joystick. Useful when you need to know when user starts interaction. Note that this sometimes doesn't mean that joystick position differs from (0, 0), which will be fixed in future.

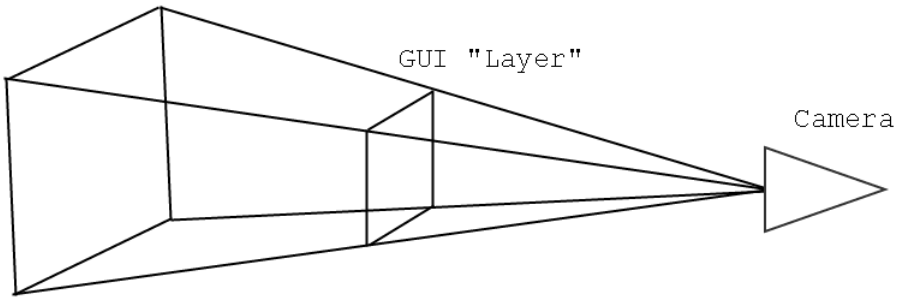
Just subscribe to these events from your script and you're good to go.

Documentation, mechanics explained (educational or contribution purposes)

The main idea behind this joystick is to use SpriteRenderer for joystick rendering, so it benefits from batching. I came up with this idea:



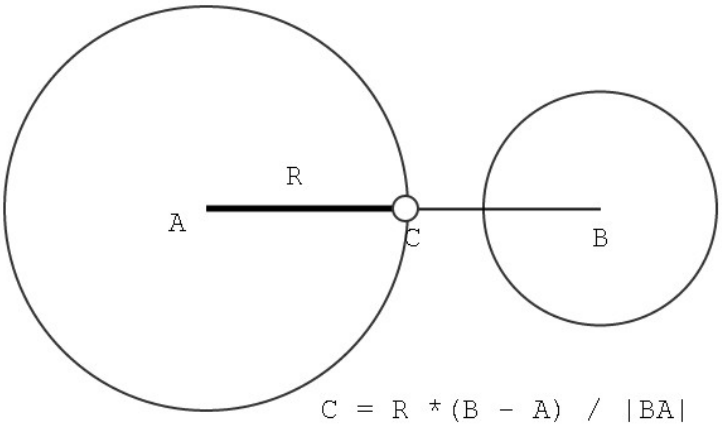
If you position your joystick in front of the camera with, let's say, 0.5 units, our joystick won't collide with any scene objects. So it looks something like this:



We need to find frustum height and width. Fortunately, there are ready-to-use equations:

```
frustumHeight = 2.0f * distance * Mathf.Tan(camera.fieldOfView * 0.5f * Mathf.Deg2Rad);
frustumWidth = frustumHeight * camera.aspect;
```

So we have our joystick with a box collider attached. We need to capture our first touch position and move our joystick there, so we can calculate relative movement of our joystick.



We can also find the coordinates of the border point of our joystick using formula above.