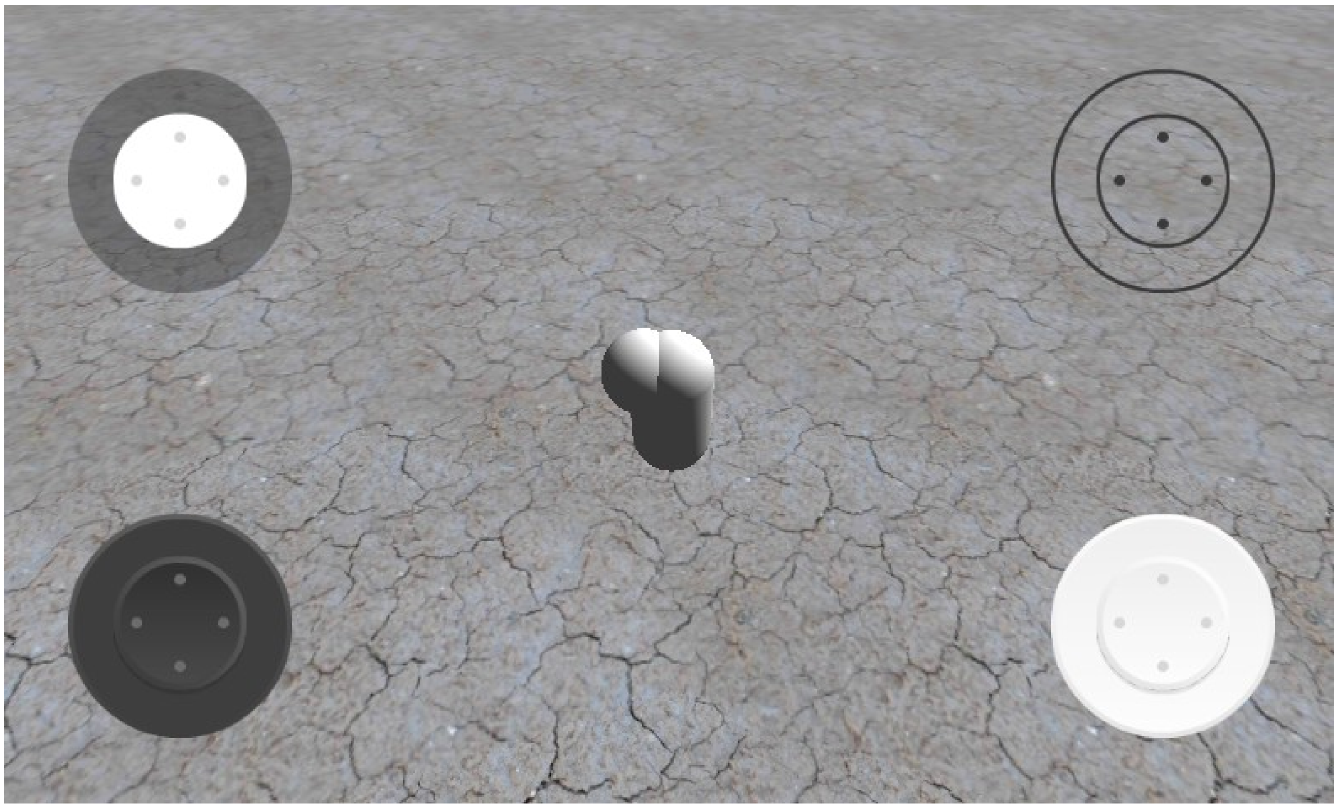
Mobile CNJoystick (v 0.2)



Contents:

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

Abstract

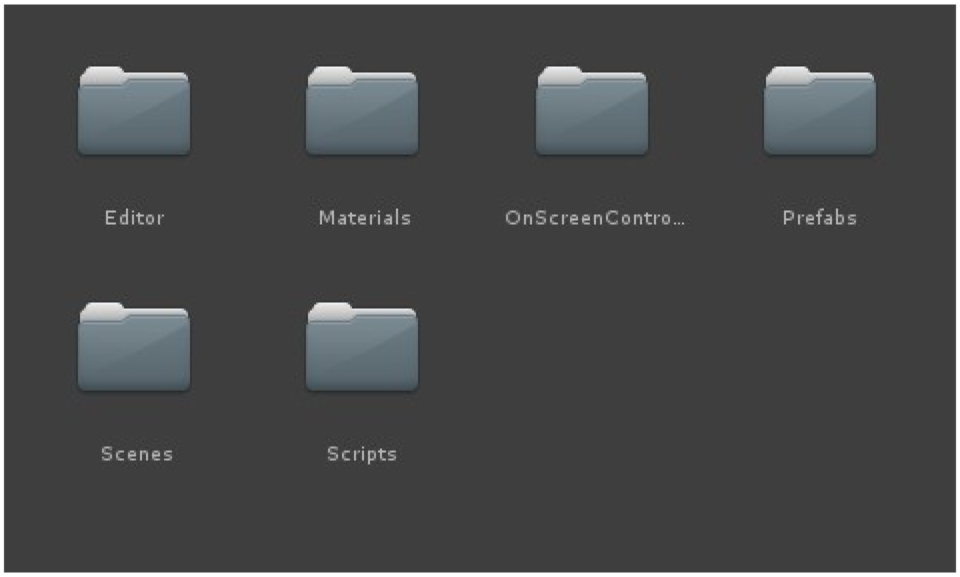
This package contains ready-to-use high performance mobile joystick, 8 different joystick styles, setup guide and detailed explanation of what's actually going on.

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.

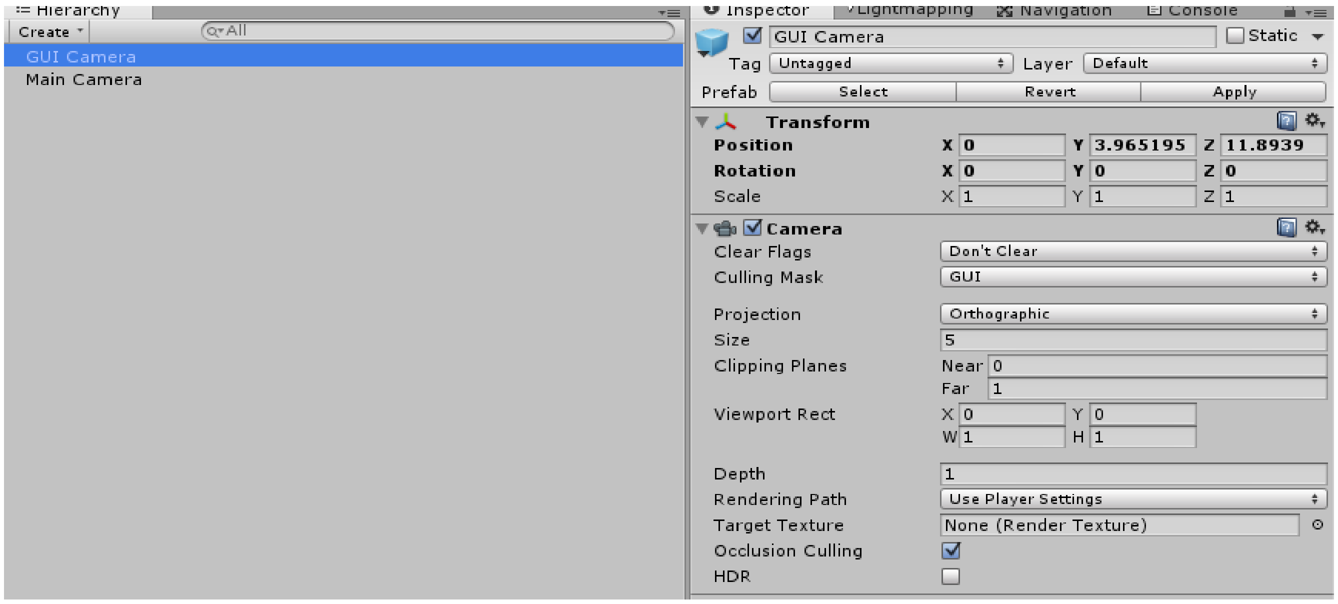
Installation guide

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, first drag the *GUI Camera* prefab onto your **scene**, position doesn't matter. You can also use any orthographical camera if you want, just keep in mind that it should have **Clear Flags – Don't Clear**, and **Culling Mask – GUI**:

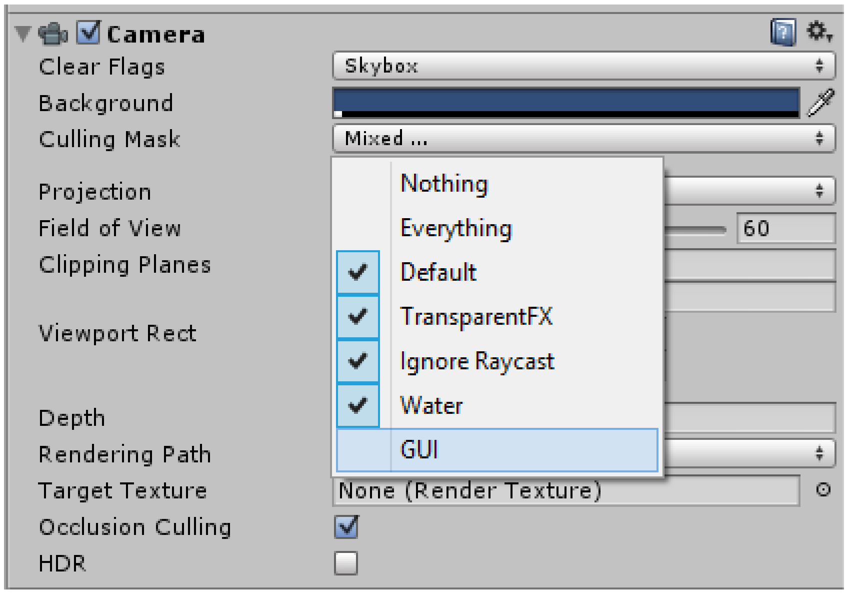

Hierarchy
Create
GUI Camera
Main Camera

Inspector | Lightmapping | Navigation | Console
GUI Camera   Static
Tag Untagged   Layer Default
Prefab Select Revert Apply

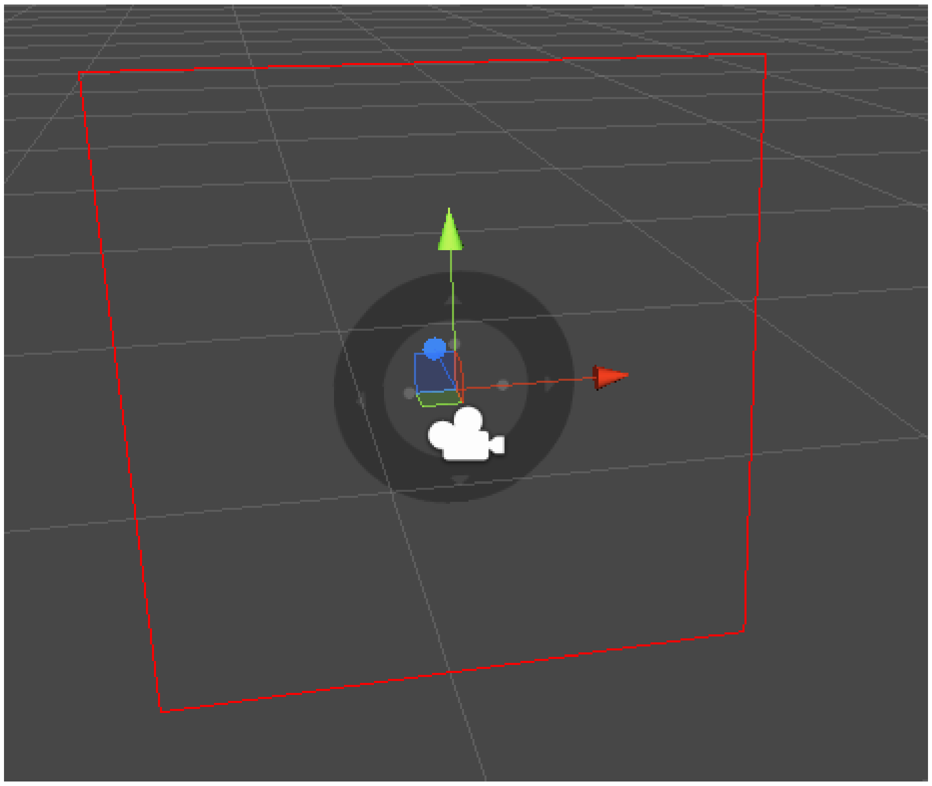
Transform
Position X 0 Y 3.965195 Z 11.8939
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags: Don't Clear
Culling Mask: GUI
Projection: Orthographic
Size: 5
Clipping Planes: Near 0, Far 1
Viewport Rect: X 0 Y 0 W 1 H 1
Depth: 1
Rendering Path: Use Player Settings
Target Texture: None (Render Texture)
Occlusion Culling: ✓
HDR: ☐

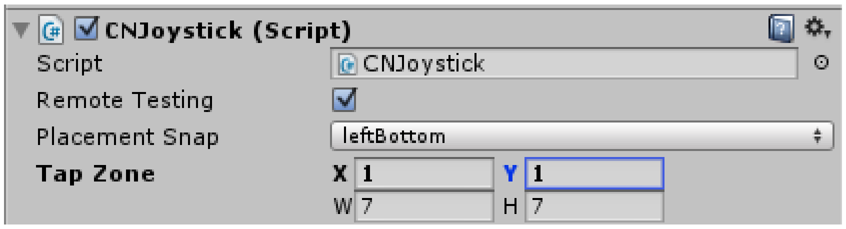
You should also **uncheck** *GUI* from *Culling Mask* on your **Main Camera:**


Camera
Clear Flags: Skybox
Background
Culling Mask: Mixed ...
Nothing
Everything
✓ Default
✓ TransparentFX
✓ Ignore Raycast
✓ Water
GUI
Projection
Field of View: 60
Clipping Planes
Viewport Rect
Depth
Rendering Path
Target Texture: None (Render Texture)
Occlusion Culling: ✓
HDR: ☐

Now drag *Joystick* prefab onto newly created *GUI Camera*, notice that it has red border around it – it's a TapZone, the effective zone of the joystick. Only taps that occur inside this zone will be used for control.



It also specifies how Joystick will snap to screen borders. You can control this by changing Tap Zone properties of the component. Change it's relative position (X and Y) to 1

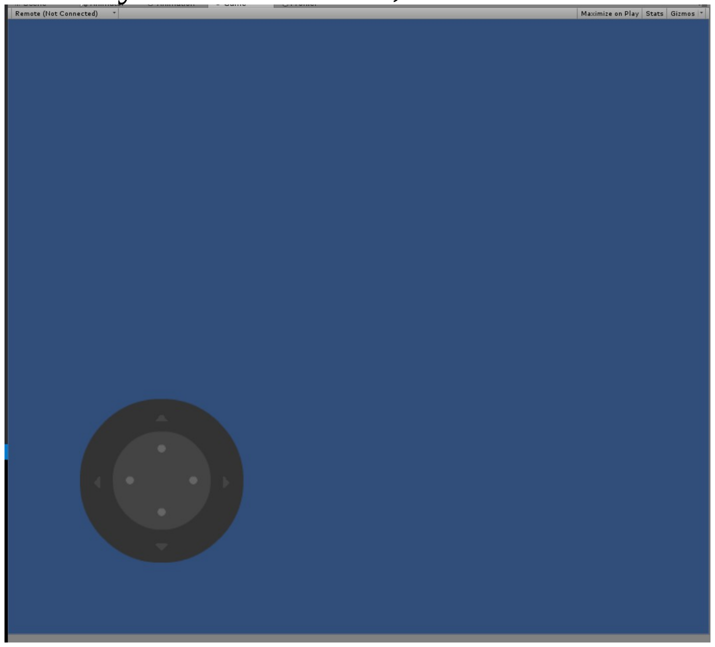


CNJoystick (Script)

Script	CNJoystick
Remote Testing	✓
Placement Snap	leftBottom
Tap Zone	X 1 Y 1
	W 7 H 7

You'll notice that the relative position of the joystick became a bit different.

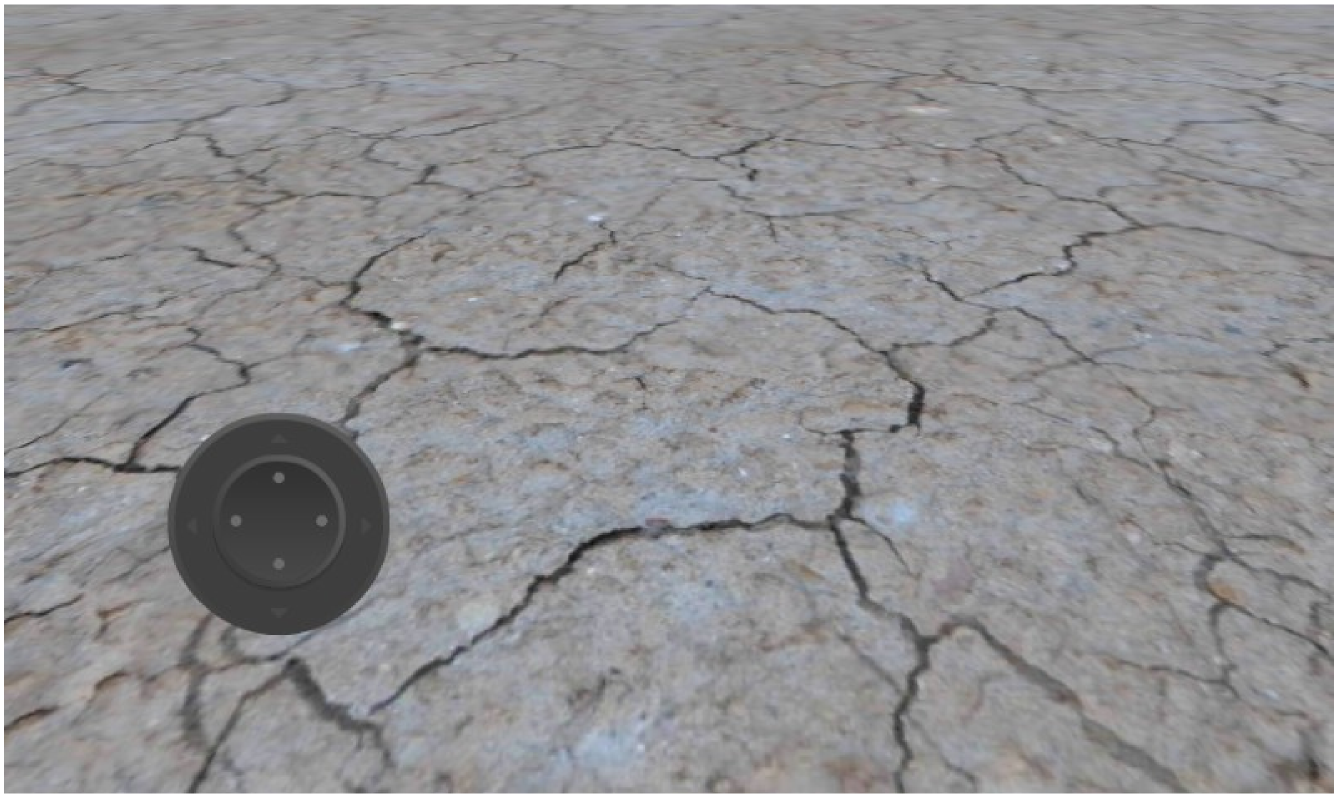
Take a look at the view of your camera now, it should look something like this:



Properties:

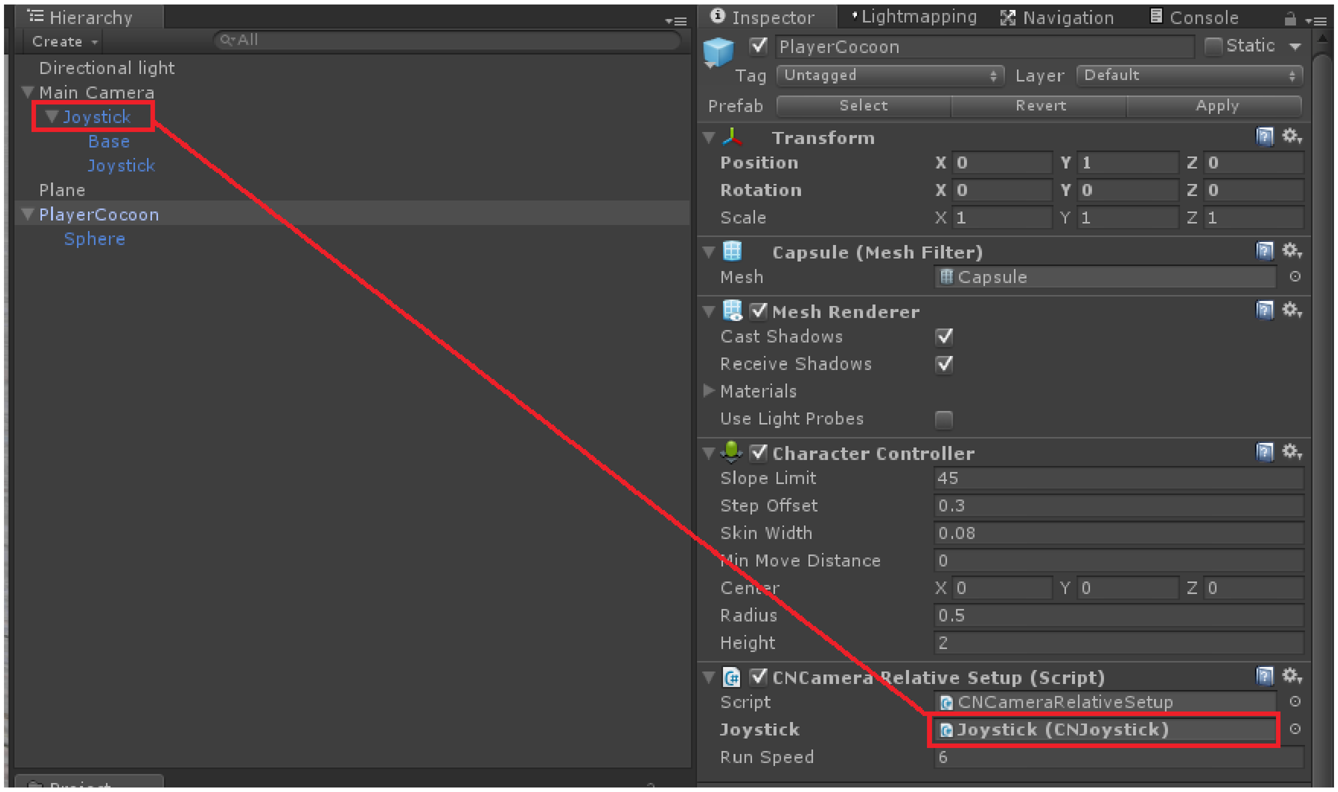
- **Remote Testing** represents whether you are testing your game using **Unity Remote** app or not. If you're going to use mouse cursor to test your game in unity, uncheck it. **However,** if you're going to use this Unity Remote app, check this button. **Otherwise, Unity will treat your touches as 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.
- **Placement snap** represents the corner to which your joystick would snap when you hit play. It snaps with the borders of the **Tap Zone,** so relative position of the it 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.



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.



Directional light
Main Camera
Joystick
Base
Joystick
Plane
PlayerCocoon
Sphere

PlayerCocoon ☐ Static
Tag Untagged Layer Default
Prefab Select Revert Apply

Transform
Position X 0 Y 1 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

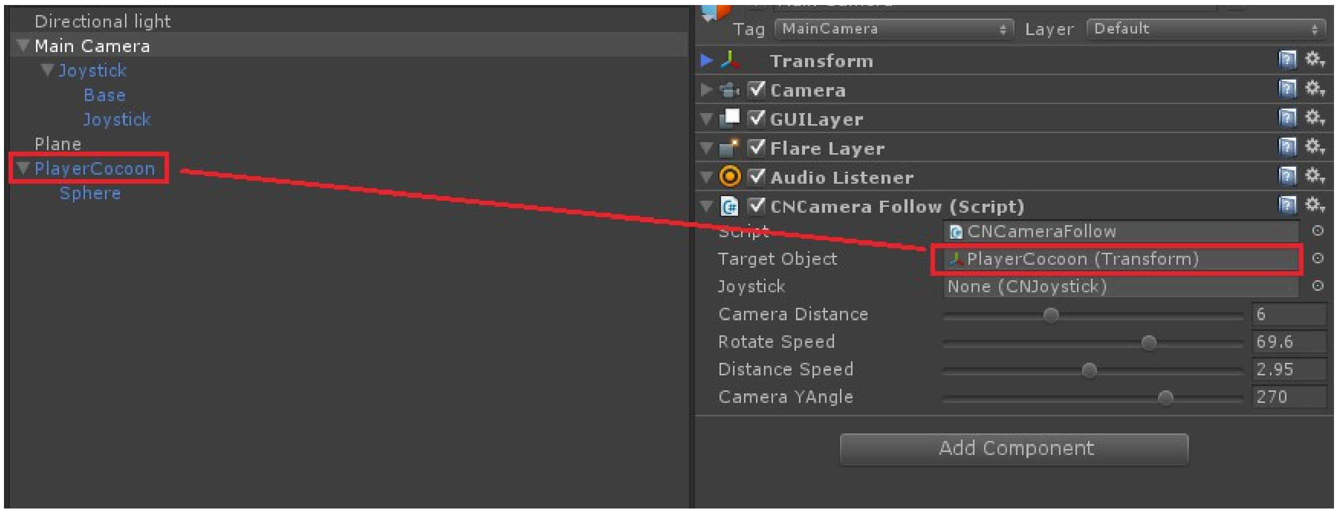
Capsule (Mesh Filter)
Mesh Capsule

Mesh Renderer
Cast Shadows ✓
Receive Shadows ✓
Materials
Use Light Probes ☐

Character Controller
Slope Limit 45
Step Offset 0.3
Skin Width 0.08
Min Move Distance 0
Center X 0 Y 0 Z 0
Radius 0.5
Height 2

CNCamera Relative Setup (Script)
Script CNCameraRelativeSetup
Joystick Joystick (CNJoystick)
Run Speed 6

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.



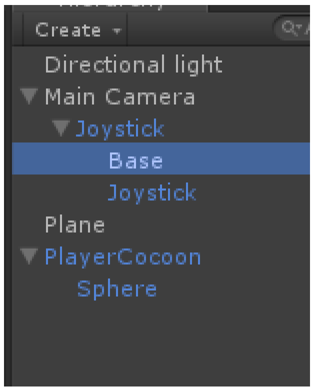
Tag MainCamera Layer Default

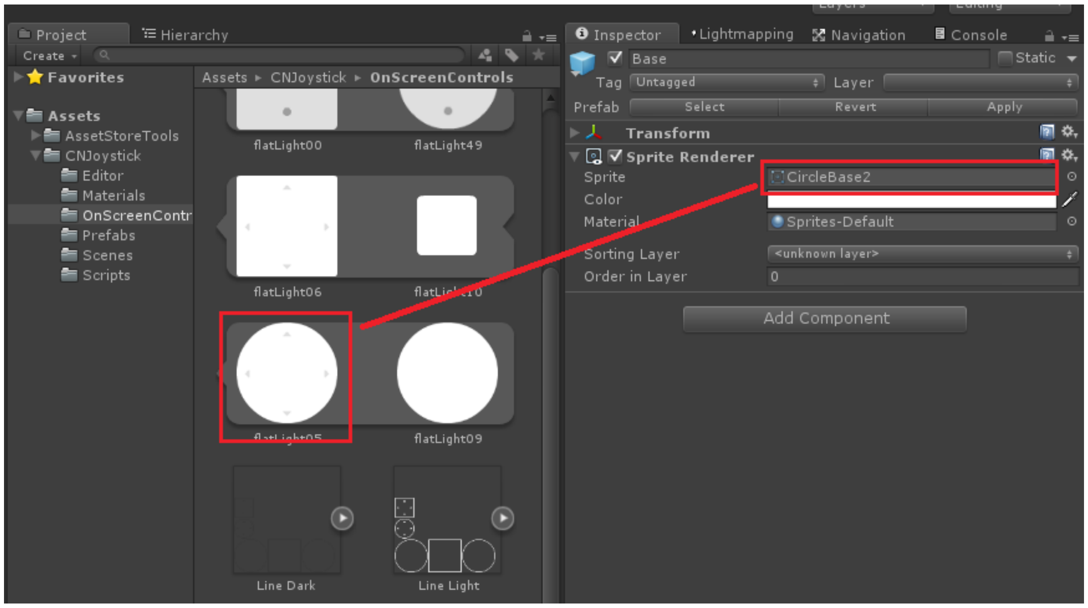
Script	CNCameraFollow
Target Object	PlayerCocoon (Transform)
Joystick	None (CNJoystick)
Camera Distance	6
Rotate Speed	69.6
Distance Speed	2.95
Camera YAngle	270

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*





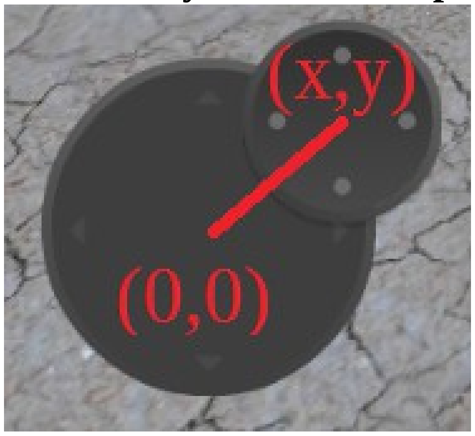
Base
Tag Untagged Layer

Sprite	CircleBase2
Color	
Material	Sprites-Default
Sorting Layer	<unknown layer>
Order in Layer	0

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. It also has **only X and Y components**, **Z** component is always **0**!



- *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.

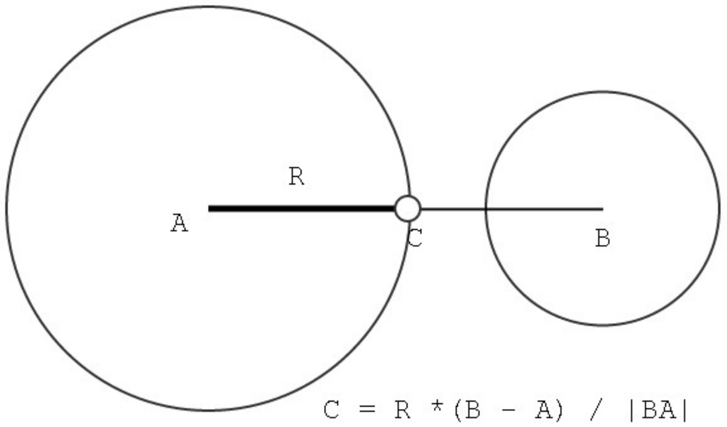
Just subscribe to these events from your script and you're good to go.

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. It renders the joystick through another GUI orthographical camera.
To find screen size in units, we can utilize these equations:

```
screenHeight = 2.0f * ofthoCamera.orthographicSize;
screenWidth = screenHeight * camera.aspect;
```

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



C = R *(B - A) / |BA|

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

Roadmap

What is to come in future versions:

- Touchpad
- ABC button pad
- Separate GUI camera? < DONE

I will definitely develop this project further.

Also, check Git repository if you feel like you want to help.