

THIRD PERSON CAMERA WITH LOCK ON

version 1.1.0.

Copyright © 2022 NyangireWorks

nyangireworks@gmail.com

Table of Contents

1. Introduction.....	3
2. Brief overview.....	3
3. Quick Setup.....	5
3.1 Setting up the Third Person Cam script.....	5
3.2 Setting Up the Lock On Targets.....	5
3.3 [OPTIONAL] Setting Up Inputs.....	6
3.3.1 Input Manager.....	6
3.3.2 New Unity Input System.....	9
3.4 [OPTIONAL] Setting up Camera Fade objects.....	11
4. Controls.....	11
4.1 Third Person mode.....	11
4.2 Lock On Mode.....	12
5. Features.....	12
5.1 Third Person Mode.....	13
5.2 Y angle limits.....	13
5.3 Camera Reset.....	14
5.4 Dynamic Control Type detection.....	14
5.5 Lock On mode.....	14
5.6 Far camera in Lock on mode.....	15
5.7 Lock On Manual / Automatic mode.....	16
5.8 Lock On Breaks.....	16
5.9 Camera Fade objects.....	17
5.10 Camera Collision.....	17
5.11 Debug mode.....	18
5.12 Off mode.....	18
5.13 Custom Scripting.....	19
5.14 Simple Character Controller.....	19
6. Inspector properties.....	20
6.1 General.....	20
6.2 Input.....	20
6.3 Third Person Properties.....	22
6.4 Lock On Properties.....	25
6.5 Camera Fade Objects.....	29
6.6 Debug.....	31

1. Introduction

Thank you for purchasing *Third Person Camera With Lock on!*

If you have any issues, questions or bugs to report, suggestions, feature requests don't hesitate to contact me via email:

nyangireworks@gmail.com

This document will walk you through setting up the camera script with Unity3D, along with explaining its features and properties.

The camera can be setup without any C# knowledge, but some of its features do require usage of C#.

This document assumes a basic knowledge of Unity3D.

2. Brief overview

The ***Third Person Camera With Lock On*** is a C# script to be used with Unity's Camera *gameobject*.

The script has 2 modes of operation: *Third Person* and *Lock On*.

Third Person camera is defined by its *distance* and *angle* at which it follows a chosen *gameobject* (referred to as the *follow* in camera properties).

It can be rotated horizontally and vertically around the *follow*, and will follow that *gameobject* maintaining its distance and angle when possible.

The *Lock On* modes primary function is to keep the player and the *lock on target* in frame (with a preference for keeping the player in frame) while still giving the player control of the camera.

The *Lock On* targets the point in between the *follow* and the *target*, its distance is automatically calculated using the distance parameter, the positions of the player and the *lock on target*. The vertical angle is no longer controlled by the player but is also automatically calculated using parameters and

distances of the scene, this is necessary so that both of the points of interest keep in frame with minimal camera movements.

The Lock On function will target a targetable object that's closest to the followed *gameobject*.

The *Simple Character Controller* script contains movement other algorithm examples that work with the camera.

The script is merely an example of how to handle movement for your character to achieve the movement in the demo.

One advisable thing is to check out the movement algorithm that makes sure that inputs are matched to the cameras orientation for example when the player inputs forward the character will walk away from the camera.

The *Fade Objects* script is dynamically created during gameplay by the camera script. It aids the optional feature of fading objects that stand in front of the camera and would otherwise block the view.

3. Quick Setup

This part will walk you through the minimal steps necessary to have a functioning camera in your scene.

Refer to next chapters for a more in depth explanation and setup of advanced features.

You can use the demo scene provided to see the example but you will have to set up inputs talked about in section 3.3.

3.1 Setting up the Third Person Cam script

Simply drag and drop the *ThirdPersonCamera.cs* onto a Camera game object in your scene.

Examine the properties of the script in the inspector and locate the ***Follow*** property.

Drag and drop the *GameObject* which the camera will follow into the *Follow* field.

The camera should now be operational with the mouse and keyboard input.

Its *lock on* function will not work until your scene has a target for the *Lock On*.

If inputs do not work skip to the 3.3. section.

3.2 Setting Up the Lock On Targets

Targeting is achieved using tags.

Create a tag to be used to designate an object as target-able (such as “*LockOnTarget*”),

In the cameras inspector properties find *Lock On Targets Tag* and type in the chosen tag.

Assign the tag to game objects in the scene (or any of their children such as bones or objects).

The *Lock On* mode should now be functional.

3.3 [OPTIONAL] Setting Up Inputs

3.3.1 Input Manager

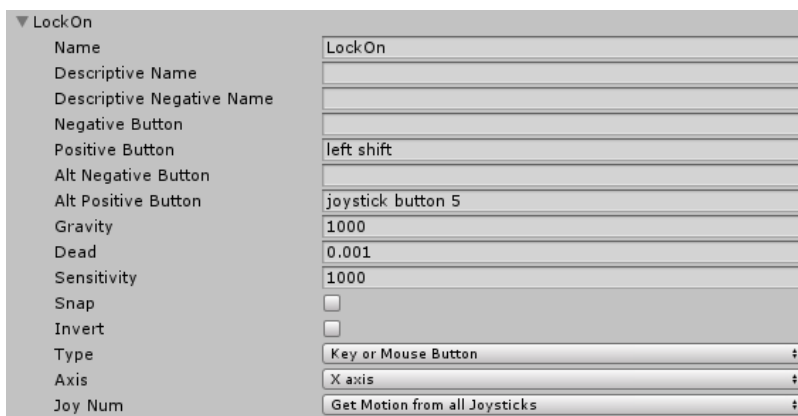
Button inputs come in two modes: *KeyCode* and *Input Manager Buttons*, while axis controls need to be setup in the *Input manager*.

To Set up inputs copy the text of the inputs you need from the *inputmanager.asset* file into your *inputmanager.asset* file located in your project folder → project settings.

Alternatively, you can set up inputs manually by going to the *Input manager* window in unity found at

Edit → Project Settings → Input:

Here are examples of setting up the necessary buttons:



The screenshot shows the Unity Input Manager window with the 'LockOn' input action selected. The settings are as follows:

Property	Value
Name	LockOn
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	left shift
Alt Negative Button	
Alt Positive Button	joystick button 5
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button
Axis	X axis
Joy Num	Get Motion from all Joysticks

Figure 1: Lock On Settings

▼ ChangeTargets	
Name	ChangeTargets
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	left ctrl
Alt Negative Button	
Alt Positive Button	joystick button 8
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button ↕
Axis	X axis ↕
Joy Num	Get Motion from all Joysticks ↕

Figure 2: Change targets button

▼ CamReset	
Name	CamReset
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	t
Alt Negative Button	
Alt Positive Button	joystick button 9
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button ↕
Axis	X axis ↕
Joy Num	Get Motion from all Joysticks ↕

Figure 3: Cam reset button

▼ Mouse X

Name	Mouse X
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	
Alt Negative Button	
Alt Positive Button	
Gravity	0
Dead	0
Sensitivity	0.1
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Mouse Movement
Axis	X axis
Joy Num	Get Motion from all Joysticks

Figure 4: Mouse X axis

▼ Mouse Y

Name	Mouse Y
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	
Alt Negative Button	
Alt Positive Button	
Gravity	0
Dead	0
Sensitivity	0.1
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Mouse Movement
Axis	Y axis
Joy Num	Get Motion from all Joysticks

Figure 5: Mouse Y axis

▼ RightStickX	
Name	RightStickX
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	
Alt Negative Button	
Alt Positive Button	joystick button 1
Gravity	3
Dead	0.5
Sensitivity	3
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Joystick Axis
Axis	4th axis (Joysticks)
Joy Num	Get Motion from all Joysticks

Figure 6: RightStick X axis

▼ RightStickY	
Name	RightStickY
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	
Alt Negative Button	
Alt Positive Button	joystick button 1
Gravity	3
Dead	0.5
Sensitivity	3
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Joystick Axis
Axis	5th axis (Joysticks)
Joy Num	Get Motion from all Joysticks

Figure 7: RightStick Y axis

3.3.2 New Unity Input System

Controls for the camera can also be setup for the new Unity Input System package (tested with 1.0.2)

To enable the Input system controls switch the **Camera Input Type** inspector variable to **InputSystem** on the *Third Person Camera* script. The Camera should now be using the generated *CameraInputActions.cs* script for controls.

If you wish to instead make your own input actions or use the Player Input component uncheck the **Input System Use Script** option, this option prevents the use of CameraInputActions c# script provided.

An example **InputAction** has been provided in which all inputs have been set to button except for Orbit – the input that controls the orbit of the camera. Its set to Pass through Vector 2.

NOTE: The input action asset is an example, its not being used by the camera instead the camera uses the **CameraInputActions** script, if you wish to use the provided input action check the ‘generate C# script’ in the inspector or use the PlayerInput.

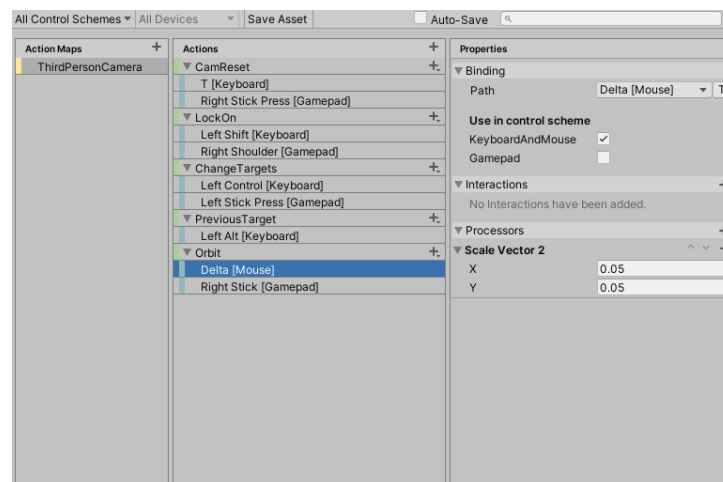


Figure 8: Note: to have mouse delta inputs be equivalent to the demo and keycode variants the vector needs to be scaled by factor 0.05

If you wish to control the camera through your own manager script use the provided functions.

OnLockOn(bool started) – call when lock on is pressed (started true) and when lock on is released (started false)

OnCamReset() - when the cam reset is pressed

OnOrbit(Vector2 dir) - call each frame and send the input from mouse delta or gamepad stick as vector2

OnChangeTarget() - when change targets is pressed

OnPreviousTarget() - when previous targets is pressed

OnControlsChanged(bool gamepad) - when the control scheme has changed, gamepad has special features and behaviours, for every other control scheme send false.

There are also equivalent functions for using PlayerInput Unity Events with *InputAction.CallbackContext* value parameter.

3.4 [OPTIONAL] Setting up Camera Fade objects

If you have objects in your scene that do not collide with the camera and the camera can pass through, or have objects that would otherwise block the view of the camera such as pillars, other players, physics objects. You might wish to set up the fade feature for the camera to avoid clipping or blocking the cameras view.

An example of this feature is set up for the **Built in render pipeline**.

If you are using **High Definition Render Pipeline** or **Universal Render pipeline**, set the *FadeShader* to *none*.

If you are using a *custom shader graph*, you can use the name of the property that controls transparency in the FadeObjects script or alternatively, write your own FadeObjects script or disable the feature altogether and simply use the shader graph for a similar result.

General Setup:

Set the Cam Fade Layer to restrict the fade feature to object on the selected layers.

The Objects you wish to fade need to also have a Transparency/fade material. This can be set up in the material or shader graph.

Alternatively, a shader can be assigned under *CamFadeShader* which will be applied to object when they are being faded and then switched back.

The camera script automatically manipulates the *Fade Objects* script during gameplay.

4. Controls

The camera has 2 modes of operation *Third Person* and *Lock On*.

The Camera will be in *Third Person* mode by default.

4.1 Third Person mode

In this mode use your horizontal and vertical controls to move the camera around the followed object.

Pressing the reset button returns the camera to the back of your player and to the default angle this doesn't work in *Lock On* mode.

Pressing and holding the *Lock On* button engages the *Lock On* mode.

4.2 Lock On Mode

In *Lock On* mode the horizontal controls spins the camera around, the vertical movement is no longer controllable and instead automated. The amount of horizontal movement also depends on certain factors such as distance from target.

Pressing the *Change Target* button switches *Lock On targets* while in *Lock On* mode, this doesn't work in *Third Person* mode.

Release the *Lock On button* to disengage *Lock On*.

There's also an Axis Target Switch optional input available if the Lock On Manual Mode is turned off.

Turning Off Lock On Manual centers the camera onto the follows back and the Right stick is no longer used for camera controls, so Axis Target change can be used to switch target based on the directional input of the mouse or right stick.

5. Features

The camera features the following scripts: *ThirdPersonCamera*, *FadeObjects*, *SimpleCharacterController*, *DrawIfAttribute* and *DrawIfProperty*.

The *FadeObject* script is used automatically by the *ThirdPersonCamera* and the *SimpleCharacterController* is an example script for a character controller that works with the camera.

The *ThirdPersonCamera* script is the main camera script and its features will be discussed in this section.

DrawIfAttribute and *DrawIfProperty* are extra scripts that allow for a cleaner inspector by conditionally hiding variables.

There's also an included shader `DiffuseZWriter` used for fading objects.

5.1 Third Person Mode

The third person mode is the normal mode of operation, the camera follows the transform position and orbits the followed object at a chosen distance and angle of viewing and its vertical and horizontal orbit can be controlled via mouse or controller input.

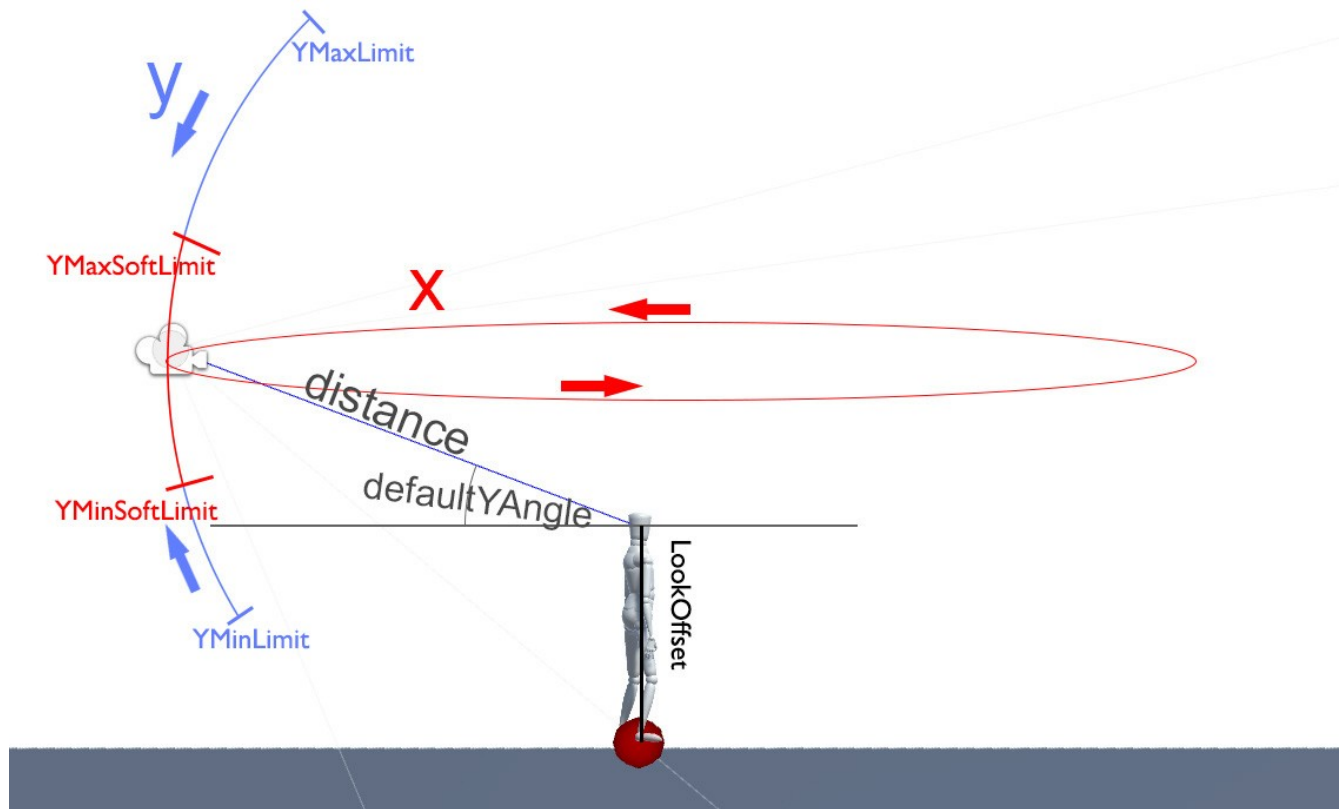


Figure 9: Third Person Mode

5.2 Y angle limits

The *Default Y Angle* is the default camera angle at which the camera looks at the follow object.

The *Y max* and *Y min limit* angles define the area in which the camera can be moved vertically.

There's optional functions that work **only with the controller** such as Camera linear interpolation to default and soft limits.

The *Lerp Camera to default* feature makes it so that once controller input is no longer read the camera will smoothly linearly interpolate back into the default Y angle, this can be used if you have a specific viewing angle in mind but still want to allow players to look up and down.

A different setup is to have *soft limits* enabled, then *Lerp Camera To Default* will lerp inside the soft limit range

Soft limits allow the camera to be positioned in between them, but also the player to look outside them, once the controller input is no longer read the camera will linearly interpolate to be back in between the *Soft limits*.

This can be useful if you want to allow the player vertical control of the camera but also allow extreme angles of viewing that wont interfere with regular gameplay.

5.3 Camera Reset

Camera reset is an optional feature, pressing the *camera reset button* will move the camera to its neutral position and angle such as back to the characters back and to the default viewing angle.

5.4 Dynamic Control Type detection

This is an optional feature that allows the camera to switch its input from controller to mouse and vice versa when it detect the use of the device.

Its incompatible with the *Use mouse* and *Use Controller* options as it will set them automatically.

If you turn off this feature then the *Use Mouse* and *Use Controller* can be used to set the input device as needed.

5.5 Lock On mode

The *lock on mode* is the second mode of operation for the camera, its vertical angle and distance is calculated automatically and is designed to have minimal transition from the *third person mode*. Its also designed to keep both the *target* and the *follow* in view with the *follow* having a higher priority.

The *lock on* camera will always be looking at the center point between the *follow* and the *target*.

The distance of the camera is calculated automatically using the *distance* property, the angles of viewing, and the *Screen Bottom Margin* property.

The *screen bottom margin* makes sure the camera will always have the *follow* in view above the bottom of the screen by a set in game distance (see 16).

The *Lock on mode* features 2 sub modes: *close* and *far camera* that are automatically switched between.

The *close camera* can be freely horizontally rotated.

The *far camera* is designed to work with targets that are more distant from the follow and require different algorithms to keep them and the follow in frame.

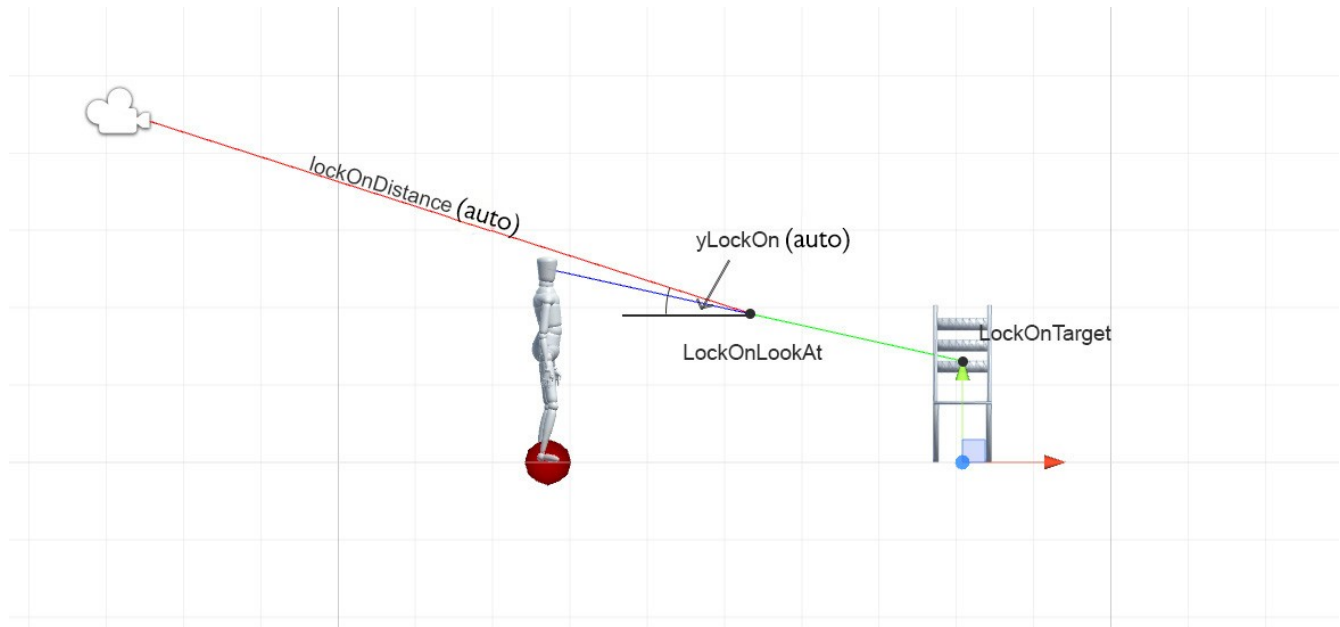


Figure 10: Lock On Mode

Lock On mode can still orbit around its look at target.

That can be turned off to keep the lock on camera at the characters back by disabling *LockOnManual*

5.6 Far camera in Lock on mode

Lock On operates in two modes *Close* and *Far*, the close mode is limited by the *Lock On Camera Full Rotation Max Distance*.

In *Close mode* the lock on camera can rotate horizontally freely.

However in *Far mode* the assumption is that the orbit radius is becoming too big, to prevent being able to orbit the camera around an entire level the camera can only orbit a limited angle around the follow.

This angle is calculated automatically using the orbit radius and the *Lock On Rotation Range Percent* parameter.

Lock On Rotation Range Percent determines how much the camera can rotate by setting how far is the *follow* allowed to be off center (see 16) with 0% meaning the camera cannot rotate at all and the *follow* stays in the center of the screen and 100% meaning the camera can rotate just enough to make the character go to the edge of the screen, higher percentage values are also possible to increase the camera rotation.

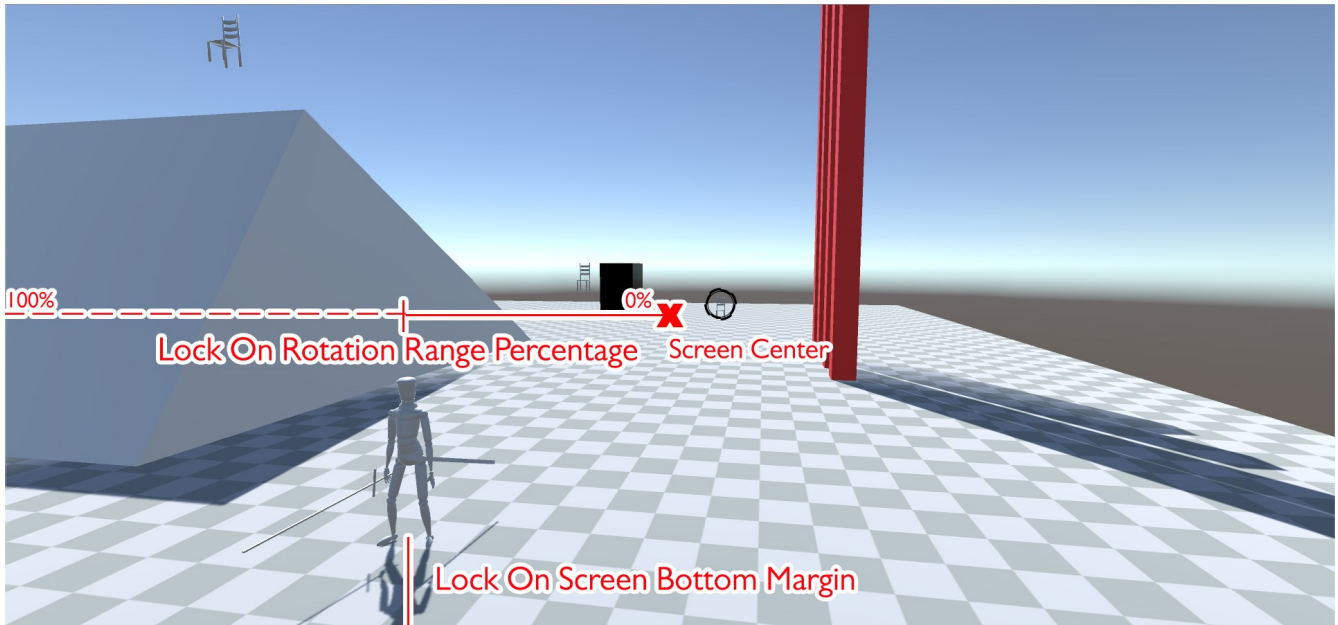


Figure 11: Far Camera Function

5.7 Lock On Manual / Automatic mode

Lock On Manual (default) allows the camera to be manually controlled while in lock on mode allowing orbiting around the look target. Turnign off manual the Lock On will center the cameras orbit on the follows back and the controls for orbitin can be used for target switching instead

5.8 Lock On Breaks

Lock On can be disallowed to target an enemy based on factors such as distance, whether its whithing the view frustum of the camera, behind a wall or whether the player has the target obscured.

The Lock On can also be broken during play.

If the *Break Lock On When Out Of Range* is set to true, any *lock on targets* that exit the distance limit will be automatically locked off.

If *Break Lock On When Target Behind Wall* is checked the camera will preform a check for when the target is no longer visible and stop locking on.

5.9 Camera Fade objects

An optional feature deigned to avoid objects getting in between the player and the camera such as pillars, small physic objects and other players. Or for objects that do not collide with the camera to avoid the '*clipping*' look when intersected by the camera.

Note: Clipping means that the surface of the object is intersecting with the cameras view plane, making it obvious that the object is hollow.

This feature requires that all the objects to be faded by the camera are in the appropriate layer that is also part of the *Camera Fade Layer* group.

The camera script automatically assigns and destroys the *Fade Objects* script during gameplay.

5.10 Camera Collision

The camera uses a *boxcast* for collision detection, the *boxcast* dimensions are that of the *camera near clipping plane* plus *collision margins*.

When the camera is colliding it uses the *Cam Clipping Smooth Damp Time* for its movement so that it gets out of collisions faster.

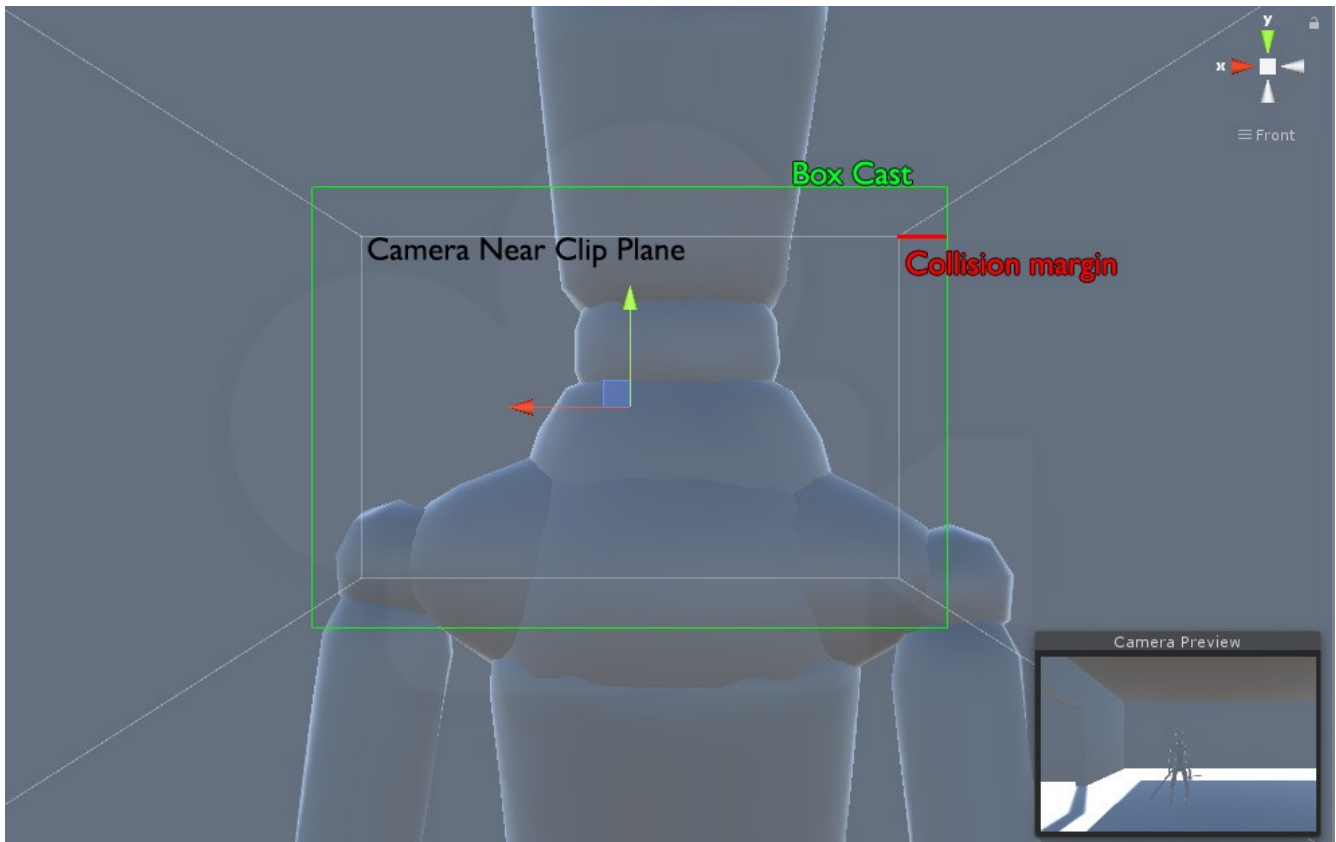


Figure 12: Camera collision

5.11 Debug mode

The *debug* inspector options can help visualize parts of the camera algorithm and log values, its mostly intended for development and debugging of the script which isn't advisable unless you wish to change the script.

5.12 Off mode

The camera script has 3 modes – *Third person*, *reset cam*, *lock on* and *off mode*

These modes control the cameras function during gameplay.

The *off mode* is a mode in which the camera isn't being moved by the script, setting the camera to this mode can be done via script and can serve the purpose to allow the control of the camera via custom scripts.

5.13 Custom Scripting

The ThirdPersonCamera scripts internal variables and function can be accessed from custom scripts in order to control the camera from a script.

The camera can be fully controlled in this manner in order to create custom movement modes or preset animations.

An example of toggling lock On is available as *LockOnThroughScriptExample* script.

Setting the camera to Off Mode will turn off the internal movement script.

```
public void MoveCamera(float x, float y, float distance, Vector3 lookAt, bool collisionYlock = false);
```

The *MoveCamera* function designed to be called every frame to move the camera for X orbit around a look point and Y to change the pitch, at a certain distance from the follow. You can calculate your own X and Y based on speeds delta time and the length and travel of your animation.

5.14 Simple Character Controller

The *Simple Character Controller* script is an example script that works with the camera.

Its purpose is to add controls to the demo character to better illustrate camera functions.

It is not necessary to use this script for the camera to work.

The *Simple Character Controller* script contains examples for movement where the inputs for movement are rotated by the camera orientation so pressing forward will lead the character away from the camera, and vice versa.

There's also examples how to pass jump information to the camera script in order to use the *Stop Camera Following Y When Player Is Jumping* feature.

Its inadvisable to use the *Simple Character Controller* script in your project, but you can use its pieces and examples to get started.

6. Inspector properties

6.1 General

Follow

The transform of the gameobject that the camera will follow, simply drag and drop the gameobject from the scene to fill this area.

Inverse X Axis

Invert the horizontal axis controls

Inverse Y Axis

Invert the Vertical axis controls

6.2 Input

Note: input settings and names are found under Edit → Project Settings->Input

Mouse Input X

The input name of the mouses X axis, controls the horizontal orbit of the camera

Mouse Input Y

The input name of the mouses X axis, controls the vertical orbit of the camera

Right Stick X

Input name of the right joystick X axis, controls the horizontal orbit of the camera

Right Stick Y

The input name of the right joystick Y axis, controls the vertical orbit of the camera

Enable Lock On

Enable or disable the *lock on button*

Enable Cam Reset

Enable or disable the *camera reset button*

Enable Change Target

Enable or disable the *change target button*

Camera Input Type

Button inputs (not axis inputs) can be controlled in three ways:

via *key codes* directly referencing a key on the keyboard/controller.

via *Unity's Input Manager* found in Edit → Project Settings → Input.

Or with the new Unity Input System package.

If you are using the *input manager* make sure all the inputs are setup properly and the names are correctly spelled for the corresponding button.

Input System Use Script

If checked the camera will use the *CameraInputActions* script to read inputs. If you wish to set up your own inputs uncheck this and read section 3.3.2.

Camera Reset Key Codes

Array of Key codes to be used to activate the *camera reset* function.

Lock On Key Codes

Array of Key codes to be used to activate the *lock on* function.

Change Target Key Codes

Array of Key codes to be used to activate the *change target* unction.

Camera Reset Button

The input name of the *Reset* function, the *Reset* function centers the camera to its initial position on the *follow*'s back and to the *default Y angle*, only works during *Third Person mode*

Lock On Button

The input name for the *Lock On* function, pressing and holding this button engages the *Lock On mode*.

Change Targets Button

The input name for the *Change targets* function, the *change targets button* changes the current *lock on target* to the next, only works during *Lock On mode*

Dynamic Control Type Detection

Whether the camera will dynamically switch between mouse and controller when an input is detected.

This makes switching from mouse to controller instant during gameplay.

This variable does not work with the new Input system as the input system has its own way of detecting which control scheme is being used.

Using Controller

Reads controller inputs for camera rotation. This can be set manually in scene, set through code, or will be set automatically if *Dynamic Control Type Detection* is set to true.

Using Mouse

Reads mouse inputs for camera rotation. This can be set manually in scene, set through code, or will be set automatically if *Dynamic Control Type Detection* is set to true.

6.3 Third Person Properties

Default Y Angle

The *default Y angle* is the default angle the camera sits at and looks at the *follow*, this angle is also used to calculate *lock on angles* automatically and *reset* the camera to default position.

If *Lerp Camera To Default Angle* is set and *Use Controller* is set the camera will return to this angle when it detects no vertical input from the controller.

Distance

The distance the camera sits at away from the *follow* position. This distance is kept constant while the camera orbits around the *follow* in *Third Person mode*, its also used to automatically calculate distances when in *Lock On mode*.

X Speed

The speed at which the camera orbits horizontally.

Y Speed

The speed at which the camera orbits vertically.

Lock Camera Y During Collision

If set when colliding the camera will project to the point on the wall at the same height as the camera would be if there was no collision, this makes it so that the camera doesn't get close to the player when colliding but instead ends up with a top down view.

Lerp Camera To Default Angle

This feature only works with the controller input (*Using Controller* must be set to true), and is only present in the *Third Person mode* of the camera.

When the camera detects no vertical input the angle will smoothly return back to *Default Y Angle*.

This allows the player to manipulate the camera to look up and down but keeps the camera in an optimal angle when there's action on screen.

Y Lerp Speed

The speed at which the camera smoothly interpolates to default for the *Lerp Camera To Default Angle* option.

Y Min Limit

The lowest angle the camera can go vertically.

These limits are necessary so that the camera's look at axis doesn't coincide with the Z axis because that would result in a *gimbal lock*.

Y Max Limit

The highest angle the camera can go vertically.

These limits are necessary so that the cameras look at axis doesn't coincide with the Z axis because that would result in a *gimbal lock*.

Use Soft Limits

Soft limits define y angle limits on the controller where the player can manually look above/below them but once the input is released the angle interpolates back inside the *soft limit*. Used so that extreme angles can be viewed but but the camera wont remain in them.

Y Min Soft Limit

The lowest angle the camera will return to when no input is detected if *Use Soft Limits* is true.

Y Max Soft Limit

The highest angle the camera will return to when no input is detected if *Use Soft Limits* is true.

Lookoffset

Where the camera is looking relative to *follows* origin.

The *follow* will most likely have its position at its *root* which normally sits at the floor.

This allows the camera to look at the object body or over the shoulder while still only following the *root motion* of the *follow*.

Third Person Look Speed

Speed of the rotation of the camera towards its look target. Used to smooth the rotational movement in Third Person Mode.

Cam Smooth Damp Time

Camera movement time from the current to desired position, used to smooth camera movement, bigger value means faster camera, lower value means smoother camera.

Cam Clipping Smooth Damp Time

Camera movement time from the current to the desired position when the camera is avoiding walls. The camera should be fast in order to have minimal clipping.

Collision Margin

Controls the distance between the edges of the camera near plane and the box cast that handles camera collision (see figure 18).

Cam Obstacle

Unity layers that act as obstacles to the camera such as walls and props, the camera collides with them.

6.4 Lock On Properties

Lock On Manual Control

Lets the camera be orbited around the target when in lock on mode, when off the camera will stick behind the follow looking at the target and the Y angle will be *defaultYAngle*.

Lock On Targets Tag

The tag used to find target-able object for the *lock on mode*.

Lock On works using the tagging system in unity, make a tag that designates the object as target-able such as “*LockOnTarget*” assign it to something target-able. This would be something in the center of your enemy pawns.

And type the tag name in this parameter.

Lock On Toggle

If set the button for engaging *Lock On* will act as a toggle, if unset it will engage *Lock On* while held.

The toggle mode doesn’t use Lock On cool down and disengaging is instant.

Axis Target Change

Bind the Right stick and mouse input to switching targets when in Lock On Mode.

The direction of the axis input will determine which target will be switched to.

Axis Target Change Threshold

How strong does the axis input need to be to trigger target switching.

Axis Target Change Cone Angle

Limit the axis target switching to targets within a cone of the direction of the axis movement.

If there are no targets within that cone the switch will not execute.

Switch Target Cooldown Time

Cooldown time between switching targets.

Lock On Sort Algorithm Type

How locking on determines which enemy to lock on to when the lock on is engaged.

By Distance From Follow – picks the closest target to the object the camera is following.

By Camera Direction – lock on to the target closest to the center of the screen

Useful if the camera is used for directly aiming at targets.

By Camera Direction 2D – lock on to the target closest to camera direction ignoring the cameras pitch. The origin of the direction is from the follows position.

The direction of the camera is projected onto the XZ plane, so the target selection would be the same at different steepness of the camera.

Useful for more action adventure style games when picking targets in front of the character.

[Experimental] Turn Off Automatic Distance Calculation

The algorithm that calculated the camera distance in order to keep the follow and target both on screen will be turned off and the distance will simply be the default distance plus the distance to the target, may result in cutting off the follow at steep angles.

[Experimental] Lock On Disengage On Steep Angle

Makes the lock on disengage when the Y angle passes a threshold.

[Experimental] Lock On Disengage Min Angle

Minimum Y angle which the lock on mode is active. Only works if *Lock On Disengage On Steep Angle* is active.

[Experimental] Lock On Disengage Max Angle

Maximum Y angle which the lock on mode is active. Only works if *Lock On Disengage On Steep Angle* is active.

Lock On Follow To Target Ratio

Sets the look at point in between the follow and the target when in lock on mode. The Lower the value the closer the camera will look towards the follow and the higher the closer it will look towards the target (1 = looking at the target)

Lock On Distance Limit

Lock On will not be able to lock on to targets outside of this range, with 0 being infinite meaning all targets in the scene will be target-able.

Break Lock On When Out Of Range

If the *Lock On distance limit* is set this option controls whether the player automatically disengages *lock on* with a target out of range or keeps the target even when it goes out of range.

Disallow Locking To Target Outside View

Targets not in the camera frustum (offscreen) cannot be locked on to or target switched to.

Disallow Locking On To Target Behind Wall

Lock On wont engage if the target isn't visible.

Disallow Locking To Not In Player Line Of Sight

Target not in line of sight from the follow won't be locked on to even if visible in the camera

The player's line of sight starts from the point at which the camera orbits the player.

Optional Player Line Of Sight Start

If set this will be used for player line of sight instead of the look at point.

Lock On Reticle

The lock on graphic used as a reticle to target the locked on gameobject.

You can use the provided image in this package or make your own, see the demo for the canvas setup example.

Lock On Reticle Works With Off Mode

The *Lock On reticle* will show up when holding *Lock On* in *OFF mode*.

Lock On Anim Speed

The *lock on reticle* is animated to appear by enlarging and disappear by shrinking. This parameter controls the total speed of the animation.

Lock On Look Speed

The camera rotation speed when moving from the current *look at* direction to desired one. Used to smooth the camera movement by slowing the angle change in Lock on mode.

Lock On Camera Full Rotation Max Distance

The distance between the *target* and the *follow* at which the camera can orbit freely in *lock on mode*.

Once out of this range the camera automatically goes to the player's side of the orbiting circle and the orbiting is limited to an angle around the player. This angle is automatically calculated to make sure the extremes always keep the player a certain distance from the edges of the screen.

Far Cam Transition Speed

Controls the speed at which the camera's angle reduces when transitioning from full range to limited range (far cam) algorithms.

Lock On Cool Down Time

Once the player disengages *lock on* the camera will still remain in *lock on* for this time (without the lock on reticle) this prevents rapid camera movements when spamming the *lock on button*, and makes quick disengages and engages less disorienting.

This is only in effect when the Lock On mode is set to hold rather than a toggle using the *Lock On Toggle* variable.

Lock On Screen Bottom Margin

Minimal bottom margin from the characters feet to the screen bottom edge in *lock on*. This prevents the characters feet being cut off and makes the character remain on screen.

Lock On Rotation Range Percent

How much can the camera rotate in *far cam mode*. The value determines how far can the character go from the middle of the screen (0%) to the very edge of the screen (100%), the camera wont rotate if the character would leave these margins.

Stop Camera Following Y When Player Jumps

In *lock on mode* this ensures the camera wont jump with the player but will remain at ground.

To enable this feature it must be handled from code.

Use the function *UpdateJumping (bool isJumping)* when the player begins/ends jumping to update the camera.

Example can be found in the *SimpleCharacetrController.cs* script.

6.5 Camera Fade Objects

Camera Fade Objects

Set to true to turn on the camera fade functionality. *Camera fade objects* can automatically fade objects that are blocking the view between the player and the camera. Which objects are fade-able is determined by their layer and material.

To enable fading of an object set its layer to something from the Camera Fade Layer, and set its *material rendering mode* to *Fade*.

Camera Fade Layer

Layers which the camera fades when in between the camera and player such as, pillars, characters, physics objects...

Fade Objects Spherecast Radius

The radius of the *spherecast* that determines whether something is in between the player and the camera.

Always Fully Transparent

Faded objects will always have 0 transparency

Full Transparency Distance

At what distance from the camera will the faded object become fully transparent.

Fade In Speed

Transition speed from higher to lower transparency.

Fade Out Speed

Transition speed from lower to higher transparency.

Fade Player When Close

If set the camera will fade its follow gameobject when too close

Player Fade Distance

At what distance from the followed gameobject does the fading start.

Fade Shader

Shader used when fading objects, you can create your own or use the provided one.

6.6 Debug

Various Debug Logs and visualizations for parts of the script.

If you have your own debug mode implemented you toggle the *DebugOn* option when you turn on your debug mode and you can implement the rest of the debug options to work with your debug commands.