EXOA

# TOUCH CAMERA PRO - MANUAL

**28/03/2023**

Latest online Version is accessible here

# INTRODUCTION

Touch Camera PRO is a really easy to use mobile and desktop camera controller with perspective switching!
It's working on both desktop and mobile devices! It supports translation, rotation around center, rotation around point, Zoom In/Out, Object Focus/Follow etc, on every camera type.

The plugin also supports scene boundaries (square or circle), and an API to Move To Position/Rotation/Distance animated or instantly.

# HOW TO INSTALL

- Download from the asset store and import in your project. There is no dependency anymore to install!
- If you upgrade, make sure to remove the previous folder before importing.

# BASIC USE

1. Add both prefabs TouchCamera & TouchCameraInputs inside your scene

2. If you want to add boundaries to your camera, fill the CameraBoundaries component on the TouchCamera gameObject with a another gameObject (having a collider, see demos)

3. Edit parameters on the CameraPerspective & CameraTopDown components if needed.

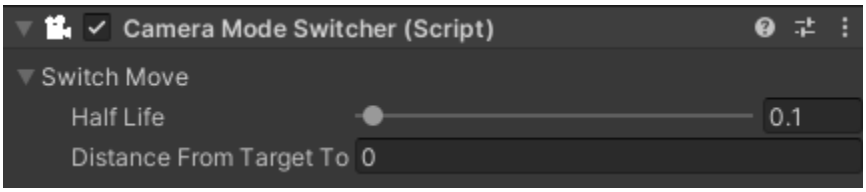
## CAMERA TYPES

Depending on the camera type you need, you can choose between:

- **Camera Perspective**: a 3D perspective camera focusing on the floor.
- **Camera Top Down Ortho**: a 2D top down orthographic  camera working on the XZ plane
- **Camera Isometric Ortho**: a 3D isometric camera focusing on the floor.
- **Camera Side Scroll**: a 2D orthographic camera working on the XY plane
- **Camera Free**: a beta 360 perspective camera working with colliders

# COMPONENTS

## Camera Mode Switcher



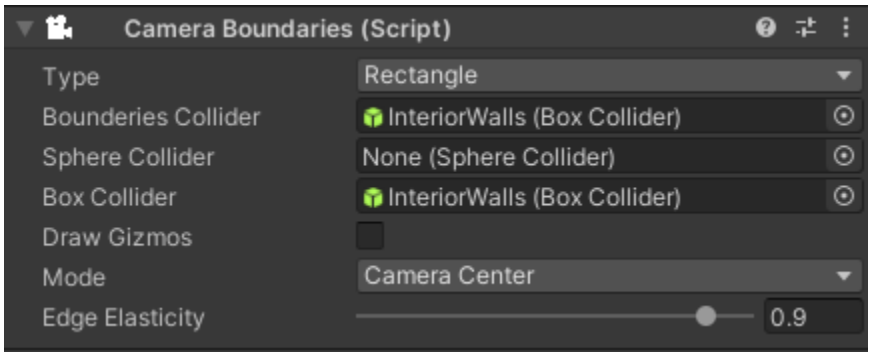

This component is needed only if you need to have two camera modes on your camera.

You can add a perspective camera mode (CameraPerspective) and an orthographic camera mode (CameraTopDownOrtho or CameraIsometricOrtho) and use the perspective switch feature with the CameraModeSwitcher component.

With 2 camera modes you need to specify which one is the default one by checking "default mode". This checkbox only appears when there's 2 modes on the camera.

The "Switch Move" settings allows you to tweak how fast the transition between the 2 modes goes. The higher the "Half Life" is, the slower the transition will be.

## Camera Boundaries




Boundaries are meant to restrict your camera's position in the scene. Two shape types are supported, Rectangle and Circle. For Rectangle, you will have to fill the box collider field and for a circle, you will have to fill the sphereCollider field. Make sure that your sphere collider Y position is on the ground.

You can check the "draw gizmo" option to see how the boundaries are in the end calculated based on your collider.

The next important option is the boundary mode. "Camera Center" will restrict the camera view center on the ground inside the boundaries. "Camera Edges" will restrict the edges of the camera view projected on the ground inside the boundaries.

The second mode works great when the camera looks down (60° to 90° pitch) (top down or isometric modes) all four corners will be restricted. For the perspective mode, if you allow the camera to have an angle < 60° (Pitch Rotation) then only the bottom corners of the camera will be restricted in the boundaries.

The first mode (Camera Center) is calculated before the camera applies the movement, whereas the Edge Mode is calculated only after the camera has moved, as it's based on the frustum final state. This is why there is an elastic effect in that case.

## Camera Perspective


The perspective camera is a mode where the camera can rotate above the ground in X (pitch rotation), Y (yaw rotation), move  and zoom in/out regarding a point on the ground. The pitch angle always needs to be greater than 0 and lower than 90° as the camera uses the floor to calculate all movements (using Raycasting).

The 3D position on the floor where the camera is looking at is called the “Offset” from (0,0,0) in the world coordinates. This is not the position of the camera itself. The camera position is calculated using this offset, the rotation, and distance from that offset point.

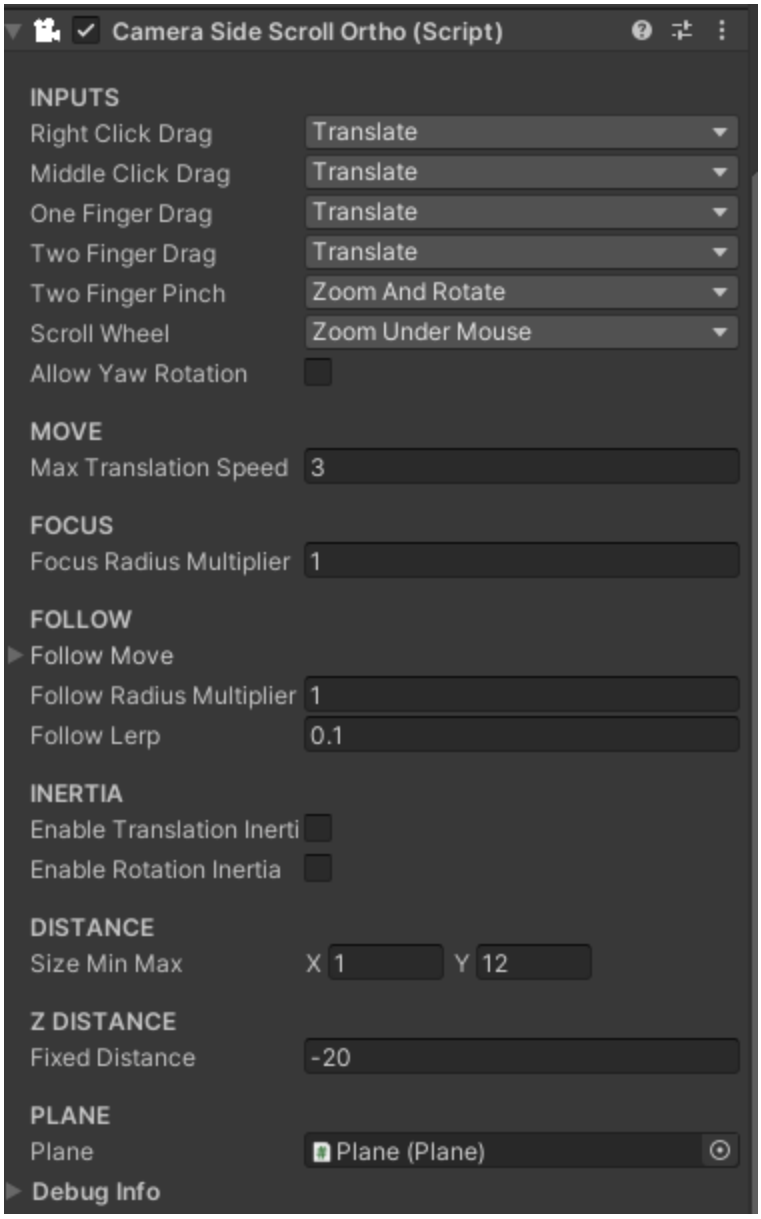
## Camera Top Down Ortho


This mode is an orthographic mode where the camera rotation X (the pitch) is blocked at 90° to look straight down to the floor. This mode is perfect for plans, mini maps and such. In this mode the camera can still be moved, rotated on Y, and Zoom In/Out (using the size property instead of the actual distance).

## Camera Isometric Ortho


The isometric mode blocks the camera in an isometric X angle  (pitch) that can be customized (By default 45°). This mode is perfect for isometric games. In this mode the camera can still be moved, rotated on Y, and Zoom In/Out (using the size property).

## Camera Side Scroll Ortho




The side scroller mode blocks the camera in a XY plane with a 0° pitch angle. This mode is perfect for side scrolling games. In this mode the camera can still be moved, rotated on Z„ and Zoom In/Out (using the size property).

## Camera Free


This beta mode allows for 360° movements around an object with colliders. The camera doesn't have to look down the floor but will rely on the object collider instead to calculate all movements.

# COMMON PARAMETERS

### Inputs

Here you can link the mouse buttons as well as mobile device finger gestures to actions. The trackpad on a computer is not considered as “fingers” but as a mouse, so using 2 fingers to pinch on a trackpad for example, will not work as a 2 fingers gesture but can work as a “scroll wheel” depending on the computer settings.

Here is a quick description of each action·

- **Translate**:  The camera offset position will move on the XZ plane.
- **Rotate Around**: The camera will rotate around the offset position on the ground on X (pitch) and Y (Yaw) axis.
- **Rotate Head**: This option is only working in the CameraFree component, it allows the camera to look around, without having any ground.
- **Zoom And Rotate**: Controls both the distance between the camera and the offset position on the ground, and the Y rotation around it.
- **Zoom Only**: Only the camera distance from the offset point on the ground will change.
- **Rotate Only**: Only the camera rotation around the offset point on the ground will change.
- **None**: No action will be performed

### Ground Height

By default the ground is at y=0, but you can specify other values, and even change it at runtime using SetGroundHeight(). Check the “Change Ground Height“ demo for an example!

### Focus

These parameters are used when calling the FocusCameraOnGameObject() to move and focus on an object, you can tweak how fast the camera moves and the distance regarding the object bounding box size.

### Distance

This is specifying the default distance of the camera from the ground. You can also clamp that distance with a min/max value.

The distance changes when the user zooms in/out using the mouse scrollwheel or the pinch gesture.

### Follow

These parameters are used when calling FollowGameObject() to move and follow an object, you can tweak how fast the camera moves and the distance regarding the object bounding box size.

### Rotation

Yaw rotation is the rotation around the world Y axis, so there is no clamping as the camera can rotate all around Y.

The pitch rotation is around the X axis. 0° means that the camera is parallel to the ground, so you need to set a minimum greater that 0, and 90° means that the camera is looking straight to the ground.

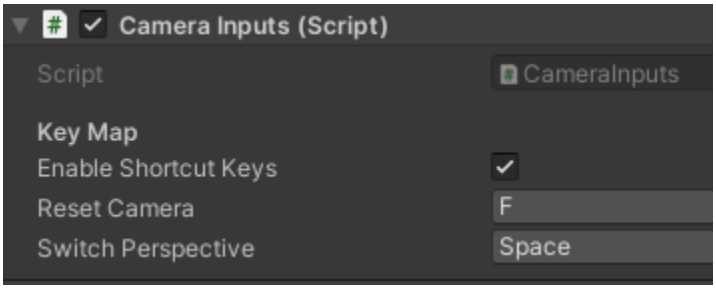
## API DOCUMENTATION

The API doc is available here: [http://monitor.exoa.fr/tcp-doc](http://monitor.exoa.fr/tcp-doc)

The sections below will describe the main methods.

## SWITCHING PERSPECTIVE

There are two ways to switch perspective in the demos, by pressing the “Space bar” or by clicking the 3D icon. To change the key you can click on the TouchCameraInputs game object and change the keys:




The other way is done by triggering an event as follow :

***CameraEvents.OnRequestButtonAction?.Invoke( CameraEvents.Action.SwitchPerspective, true);***

If you only need one mode in your scene you can remove either the “CameraPerspective” or “CameraTopDownOrtho” component and the CameraModeSwitcher component.

## MOVE CAMERA TO

You can call MoveCameraTo() or MoveCameraToInstant() to move the camera with or without an animation. You can pass to these methods a position, a rotation and a distance. Here are the different variations you can call. (See the API Documentation)

void **MoveCameraToInstant** (Vector3 targetPosition)

Moves the Camera offset position on ground to a new position, in 1 frame More...

void **MoveCameraToInstant** (float targetDistanceOrSize)

Changes the Camera distance from the position on ground, in 1 frame More...

void **MoveCameraToInstant** (Quaternion targetRotation)

Changes the Camera rotation, in 1 frame

void **MoveCameraToInstant** (Vector3 targetPosition, float targetDistance)

Moves the Camera offset position on ground and distance from it, in 1 frame

void **MoveCameraToInstant** (Vector3 targetPosition, Quaternion targetRotation)

Moves the Camera offset position on ground and rotation, in 1 frame

void **MoveCameraToInstant** (Vector3 targetPosition, Vector2 targetRotation)

Moves the Camera offset position on ground and rotation, in 1 frame

void **MoveCameraToInstant** (Vector3 targetPosition, float targetDistance, Vector2 targetRotation)

Moves the Camera offset position on ground, distance from it and rotation, in 1 frame

void **MoveCameraToInstant** (Vector3 targetPosition, float targetDistance, Quaternion targetRotation)

Moves the Camera offset position on ground, distance from it and rotation, in 1 frame

void **MoveCameraTo** (Vector3 targetPosition)

Moves the Camera offset position on ground, animated

void **MoveCameraTo** (Quaternion targetRotation)

Changes the Camera rotation, animated

void **MoveCameraTo** (float targetDistanceOrSize)

Moves the Camera distance from the ground, animated

void **MoveCameraTo** (Vector3 targetPosition, float targetDistance)

Moves the Camera offset position on ground, distance from it, animated.

void **MoveCameraTo** (Vector3 targetPosition, Quaternion targetRotation)

Moves the Camera offset position on ground, and rotation, animated

void **MoveCameraTo** (Vector3 targetPosition, Vector2 targetRotation)

Moves the Camera offset position on ground, and rotation, animated

void **MoveCameraTo** (Vector3 targetPosition, float targetDistance, Vector2 targetRotation)

Moves the Camera offset position on ground, distance from it and rotation, animated

void **MoveCameraTo** (Vector3 targetPosition, float targetDistance, Quaternion targetRotation)

Moves the Camera offset position on ground, distance from it and rotation, animated

## FOCUSING ON A GAME OBJECT

In the demos, clicking on a cube will trigger a focus on that object. Check out the "FocusOnClick.cs" script to see how it's done.

You basically just have to trigger an event like so :

***CameraEvents.OnRequestObjectFocus?.Invoke(gameObject);***

You can also call a method on a camera instance:

void **FocusCameraOnGameObject** (GameObject go, bool allowYOffsetFromGround=false)
Focus the camera on a GameObject (distance animation)

void **FollowGameObject** (GameObject go, bool doFocus, bool allowYOffsetFromGround=false)
Follow a game object

void **StopFollow** ()
Stop follow/focus/moveto animations

## FOLLOWING A GAME OBJECT

In the demos, clicking on a moving cube will trigger a follow on that object. Check out the “FocusOnClick.cs” script to see how it’s done.

You basically just have to trigger an event like so :

```
CameraEvents.OnRequestObjectFollow?.Invoke(gameObject, focusOnFollow);
```

You can also call a method on a camera instance:

```
void FollowGameObject (GameObject go, bool focusOnFollow, bool
     allowYOffsetFromGround=false)
```

The focusOnFollow parameter is a boolean. A value at false, is to follow only the position of the object, keeping the same distance that you can control with inputs. Setting “true” would also lock the distance regarding the object size on screen. You can control the focus multiplier on the camera’s component properties.

## RESET THE CAMERA POSITION

The new “Camera Reset” feature, helps to put the camera back in its initial position and rotation. In the demos it is done by clicking the “R” icon.

```
CameraEvents.OnRequestButtonAction?.Invoke( CameraEvents.Action.ResetCamera, true);
```

You can also call a method on a camera instance:

```
void ResetCamera ()
```

The camera default position/rotation/ditance is calculated from where you placed the camera in the scene.

To change the camera default position/rotation/distance for the reset feature, you can use this method :

```
void SetResetValues (Vector3 offset, Quaternion rotation, float distance, float size)
```

## CALLBACK EVENTS

In the class CameraEvents, a few callbacks are available:

**OnFocusComplete / OnFocusStart**: triggered when a follow/focus/moveTo animation starts and ends.

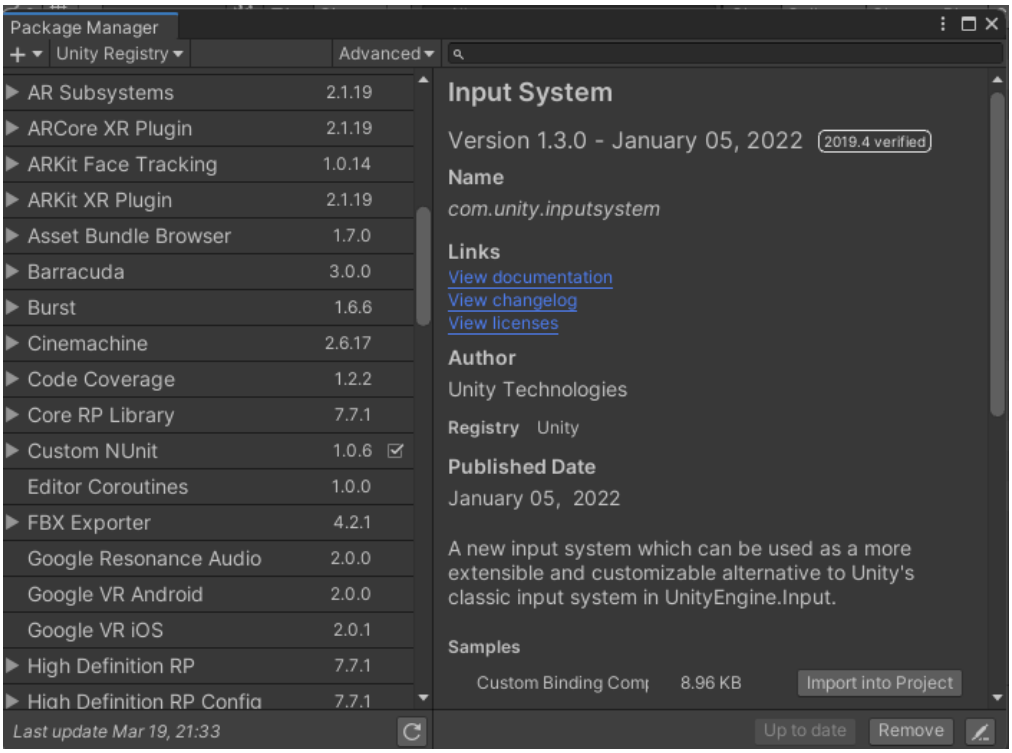
**OnBeforeSwitchPerspective / OnAfterSwitchPerspective**: triggered before an after the switch perspective animation happens.

## SIMULATING FINGERS TWIST AND PINCH

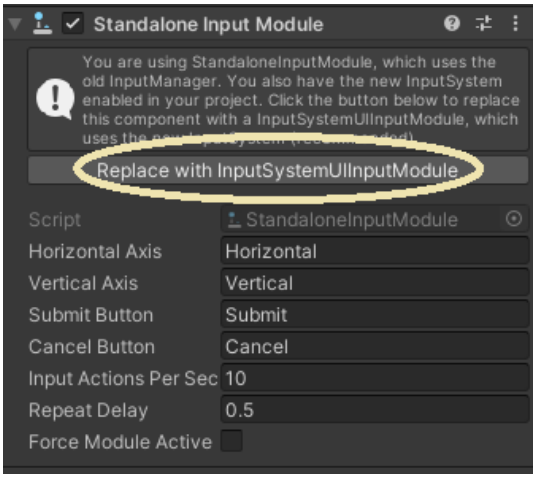
Thanks to Lean Touch, you can simulate the finger inputs. Pressing ALT + CLick on the ground will place a virtual “fingers center point” then holding the Ctrl key + clicking and dragging will let you simulate the pinch (scale) and twist (rotation) around a point on screen.

## SETTING UP WITH THE NEW INPUT SYSTEM

Make sure you have installed the Input System from the package manager:



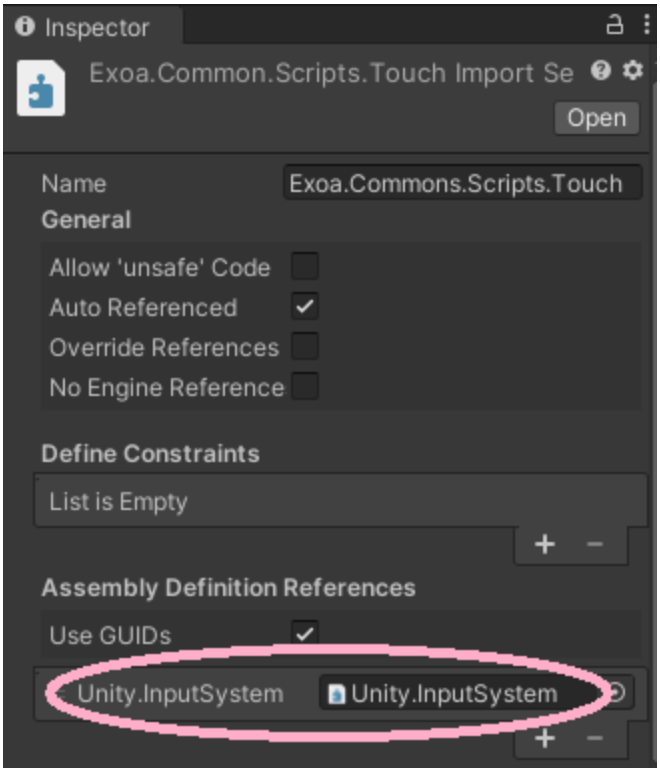

In your scene (or in the demo scenes) locate the EventSystem game object and click on “Replace with Input System”




Change the input handling in **Edit > Project Settings > Player**, under **Active Input Handling**, set “New Input SYstem”


the project will recompile and you should be using the new input system.

If you see any compile errors related to the Unity.InputSystem namespace make sure the definition file is referenced in every .asmdel file in the project like so:



# SHORTCUTS

### SWITCH PERSPECTIVE

Press "Space Bar", or press the top right "camera" button

### IN TOP DOWN ORTHOGRAPHIC MODE

Mouse Wheel : Zoom In/out

Left/Right/Middle Mouse Button Press & Drag : Drag Camera

### IN PERSPECTIVE MODE

Mouse Wheel : Zoom In/out

Left/Middle Mouse Button Press & Drag : Drag Camera

Right Mouse Button Press & Drag : Rotate Around Center

### TOUCH SIMULATION IN BOTH MODES

Alt+Left Mouse Button Click : Set the center point of the simulated fingers

Alt+Left Mouse Button Press & Drag : Drag Camera

Ctrl+Left Mouse button Press & Drag : Two fingers simulation for Pinch (Zoom In/Out) and Twist (Rotate around fingers center point)

# DEMOS

Android and PC Demos are accessible here

-

## OTHER PLUGINS

- Home Designer
- Floor Map Designer
- Level Designer
- Assets Manager Pro
- Packages Manager Free
- Tutorial Engine

## SUPPORT

Please post your questions and issues on the forum : https://support.exoa.fr/