MS Advanced Camera Controller

Created by: Marcos Ismael Schultz
Fórum: www.schultzgames.com
Email: marcos11-24@hotmail.com
Asset Store page: https://assetstore.unity.com/publishers/23246

'MS Advanced Camera Controller' is a camera controller, which has several camera options to suit the most diverse situations. The cameras included in the code are:

- **LookAt The Player:** A camera that stands still, but always rotates toward the object containing the script.
- **FirstPerson:** A first-person camera that allows 360-degree rotation without moving the player. Ideal for FPS.
- **FollowPlayer:** A camera that follows the object that contains the script smoothly, and avoids obstacles, getting in front of them. This type of camera also makes a smooth rotation towards the object containing the code.
- **Orbital:** A simple orbital camera with configurable zoom, distance and motion speed options, and also contains a function that avoids obstacles and detects collisions.
- **Stop:** A totally dead camera with no action at all.
- **StraightStop:** A camera that stands still in its position, but keeps the horizon always straight.
- **OrbitalThatFollow:** A junction of the '**FollowPlayer**' and '**Orbital**' cameras.
- **ETS StyleCamera:** A '**FistPerson**' camera, with an additional sliding option when the player looks to the left, allowing the camera to move slightly, and come back in case the player looks to the right.
- **Fly Camera:** A free camera, which allows you to walk the scenery freely, without collisions or limitations.

'MS Advanced Camera Controller' also includes two demo scene, with all kinds of cameras listed above, to demonstrate the operation. One scene is set up to work on your computer, and the other is set to work on mobile devices.

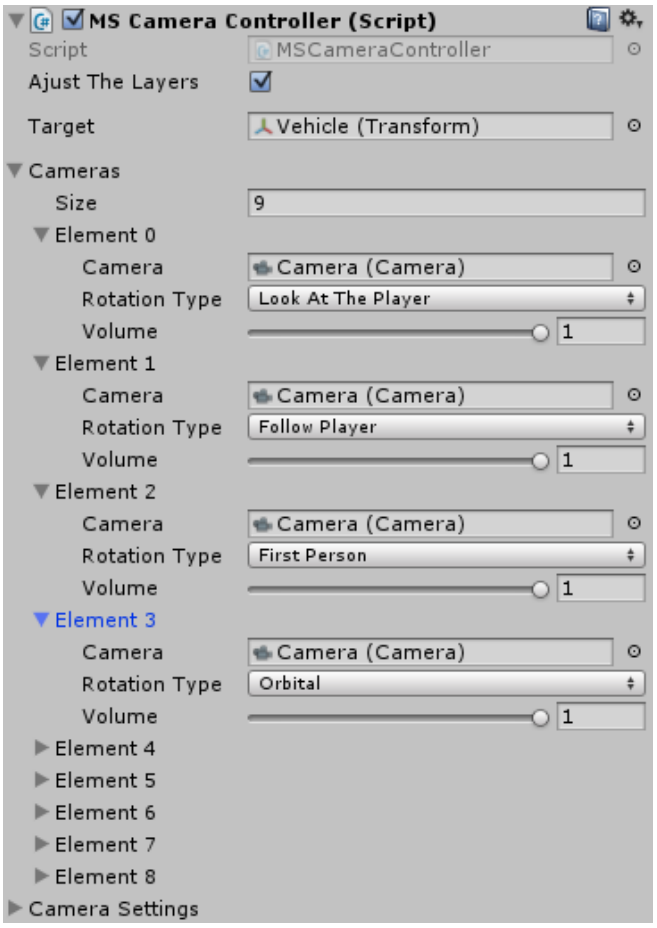
The system is extremely configurable, and can be implemented in virtually every situation. The input system is configurable, and can be used in the computer, mobile devices, joystick and other options.

How to use:

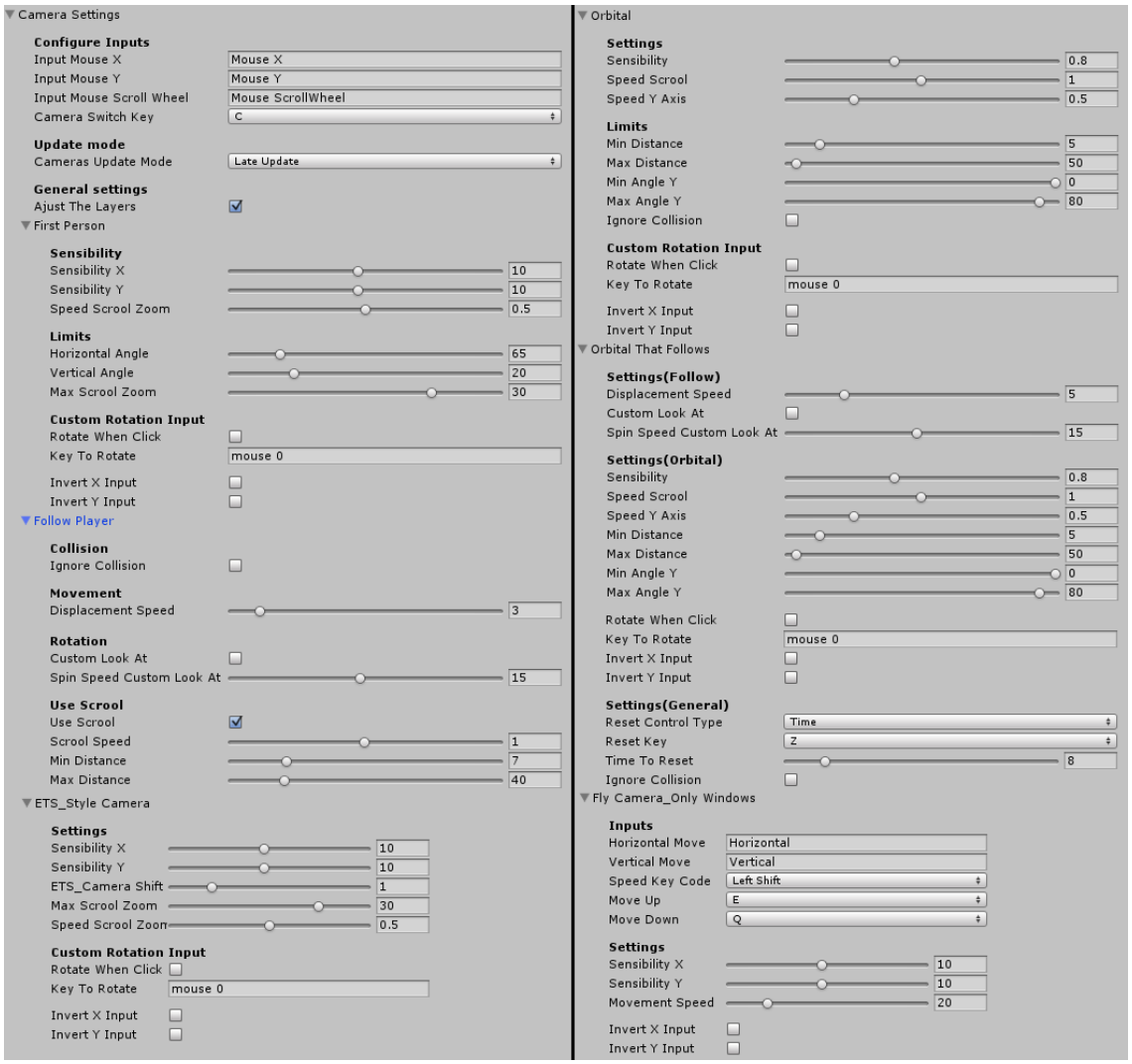
To use this feature, simply place the '**MSCameraController**' script on any object in your scene. It can be an empty object or even the player itself. Having done this, you must associate the player with the variable 'Target', or the target the cameras will follow. If you do not associate any objects in the 'Target' variable, the cameras will follow the very object on which the script is allocated. After that, just associate the cameras with their variables and set everything up.

In the **'Cameras'** array, you must define the number of cameras you have and press the **'Enter'** key to initialize the array of cameras.

Once you've done that, simply associate each camera with an index and set the type of rotation or movement that each camera should have, as the image below illustrates.



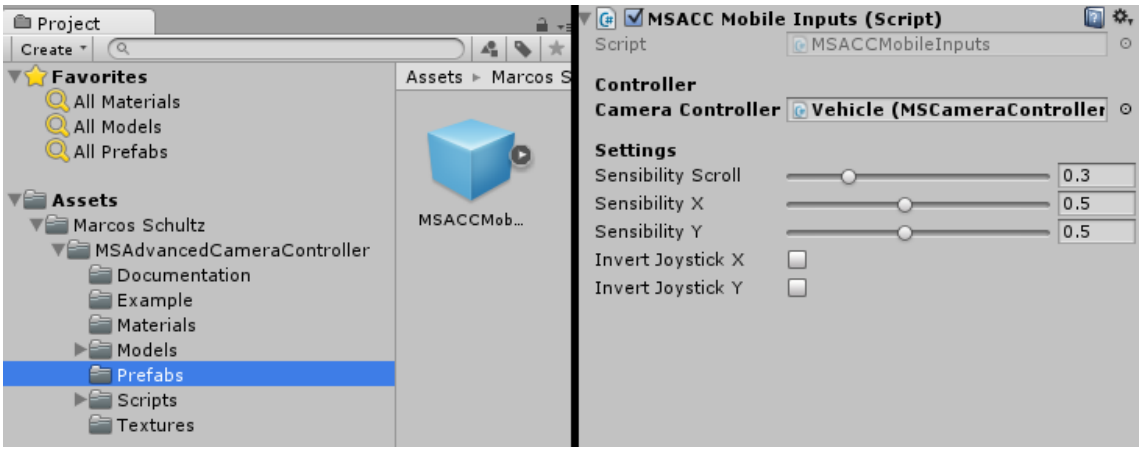
You can configure each type of Camera as well, by setting the limits of movement, speeds, rotations, among other things, as the image below demonstrates.



The entire system has 'Tooltips' in the variables, making the system easy to implement, and helping to avoid errors. These Tooltips provide warnings that help the user to implement the system. To see the warnings, just rest your mouse over some variable, and the warning will appear.

Mobile inputs:

For the system to work on mobile devices is very simple. In the **'MarcosSchultz > MSAdvancedCameraController > Prefabs'** folder there is a 'prefab' named 'MSACCMobileInputs (new)'. Just place this object in your scene and configure it.



With this object in your scene, a virtual joystick will be created automatically to allow the cameras to be controlled via touch.