# Drive In VR

Drive in VR! Includes steering wheels, handlebars, gear shifters, and every other component you need to drive in virtual reality.

V 1.2.9

- Incomplete documentation parts will be improved over time.
- Get the most up to date documentation by clicking here.
- Remember you can hover over fields in the “Inspector” window in Unity’s editor to read tooltip explanations of each field.
- If you have any questions or need assistance email support at intuitivegamingsolutions@gmail.com.

## Table of Contents

*NOTE: See 'API Reference.pdf' if you are looking for source code documentation.*

# **How to:** Import the Package into a Unity Project

***Video tutorial:*** [Youtube] Drive In VR - Setup Blank Project

There are 2 ways to import the package.

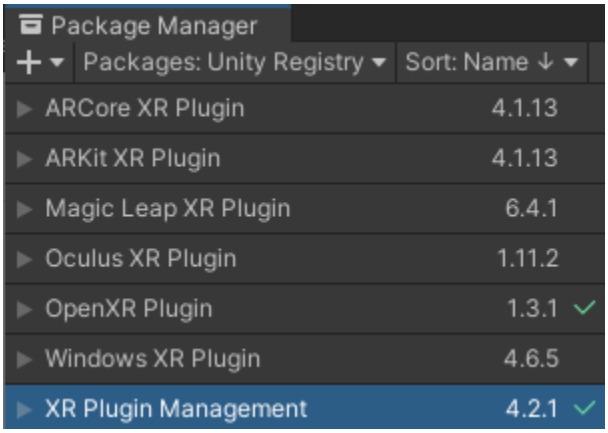
a. (Recommended) Using the Unity Editor ‘Package Manager’.
    i. Open the Windows→Package Manager using the Unity editor toolbar.
    ii. In the upper-left corner of the Package Manager window select ‘Packages: My Assets’.
    iii. Search for “Drive In VR”’ in the list or use the search bar in the window.
    iv. Select the asset in the package manager, select ‘Download’.
    v. After the package has finished downloading click ‘Import’ to import it into the project.
b. Importing DriveInVR.unitypackage
    i. Using the Unity Editor’s toolbar select Assets→Import Package
    ii. In the file explorer that opens navigate to DriveInVR.unitypackage
    iii. Double click the package and import it.

# Getting Started

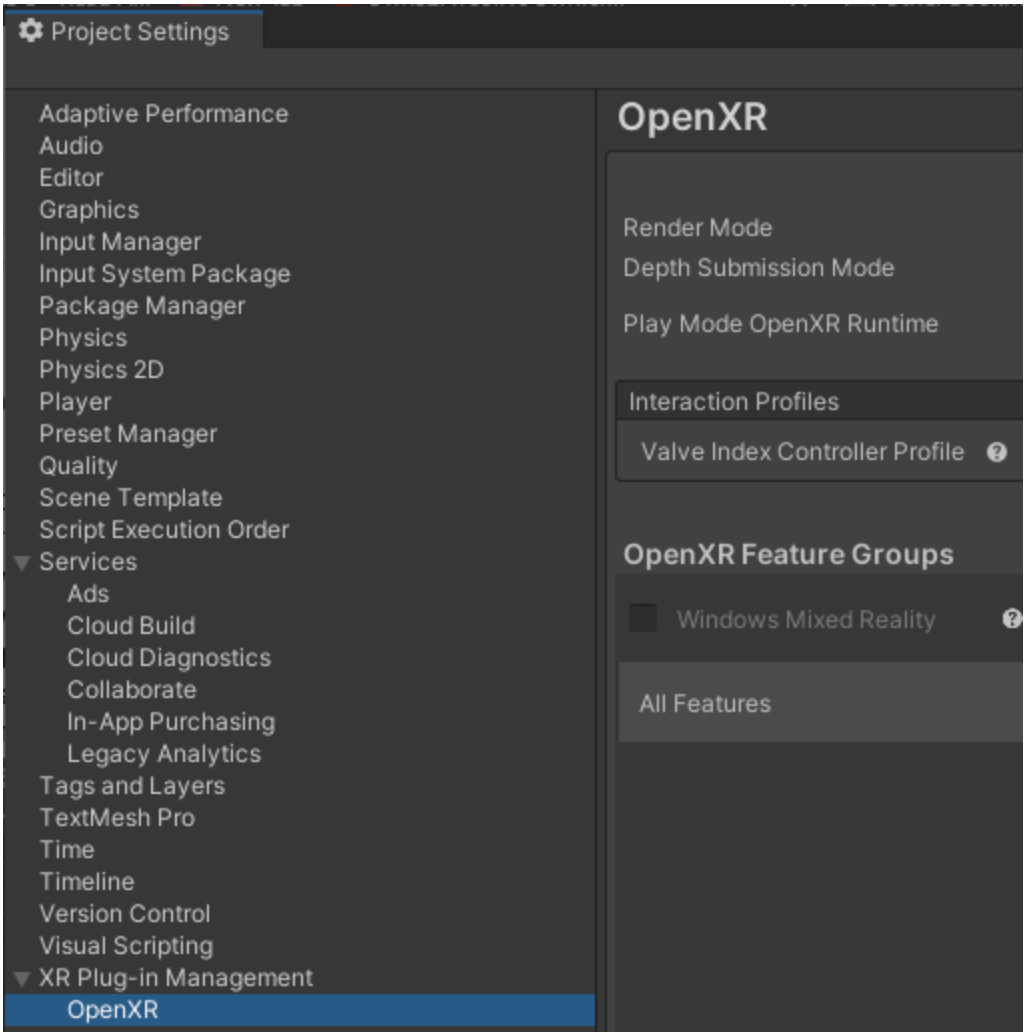
## 3.a. Importing The Asset & Required Packages

- It is really easy to get started using the demo scenes!

1. Start a new (or open an existing) Unity VR project.
    a. You can start a VR project using the 'New Project' feature in the Unity Hub.
    b. You can ***upgrade an existing non-VR project*** by opening the 'Package Manager' window and installing the '***XR Plugin Management***' package from the 'Unity Registry'.
        i. Install another package such as OpenXR (recommended), SteamVR, or similar.

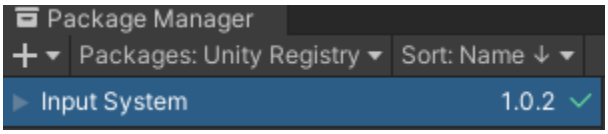



        ii. The last step to setting up 'XR Plugin Management' is to configure your chosen XR plugin – in the case of the screenshot below the 'OpenXR Plugin' was used.




iii. More detailed steps can be found on google regarding setting up VR in an existing Unity project.

2. If you want to use the provided demo content make sure your project has the **new unity input system** enabled *(it may be enabled by default in newer versions of Unity)* so make sure you have the '**Input System**' package installed in your project if you want to use the provided demo content.



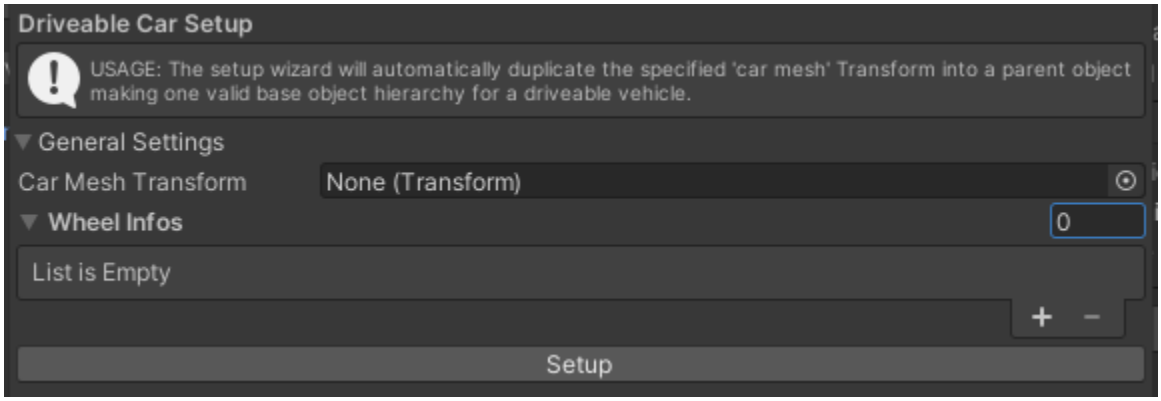

3. Import the 'Drive In VR' asset from the Unity Package manager (or a .unitypackage file) if you haven't done so already.
4. Connect your VR device and launch the demo scene(s) to get playing!

***If you already have 'Physics Hand' included in your project and want to use it in 'Drive In VR' you may now import the included 'Integration_PhysicsHand.unitypackage' file into your project.****

*Check out this Youtube Tutorial - 'Creating & Driving Your First Vehicle'*

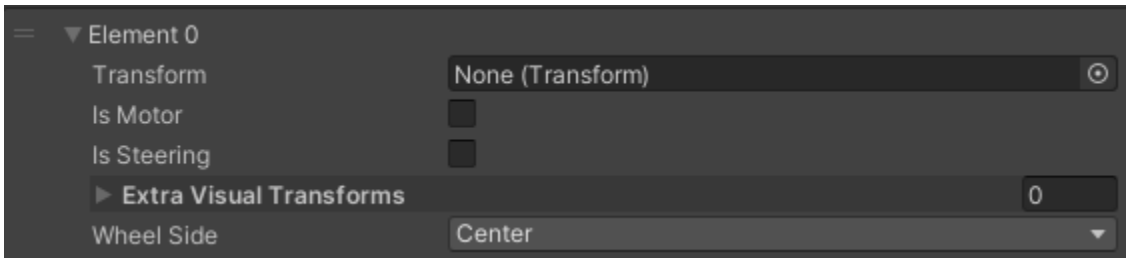
## 3.b. Creating Your First Vehicle

a. Open the 'Driveable Car Setup Wizard' by going to the Unity Editor toolbar and selecting 'Tools → VRDriving → Driveable Car Setup Wizard'. You will see a menu appear like the one shown in the screenshot below:




b. Specify the number of wheel colliders you want your vehicle to have by setting the # of 'Wheel Infos' in the box highlighted in blue in the screenshot above. ***Note that a single wheel info may drive multiple visual wheel geometries, this is useful for things like the 'duelie' style wheels on the demo semi truck.***

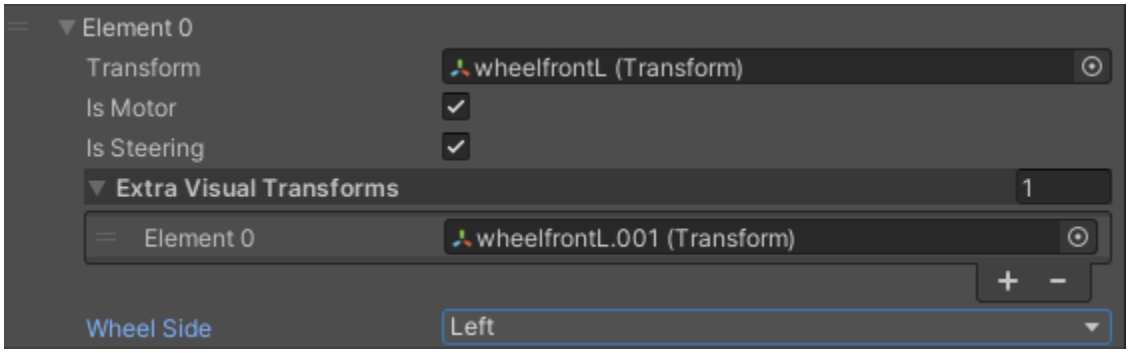
c. In this demo we are going to be setting up the included 'Mesh_VRDrivingDemoCar01' as a drivable vehicle. *Since the vehicle we are setting up has 4 wheels we enter '4' in the 'Wheel Infos' count box.* Four of the entries shown in the screenshot below will appear:




d. For each of the 'Wheel Info' entries you added in the last step specify the relevant references and values based on the descriptions from the table and screenshot below:

| | |
|---|---|
| Transform | A reference to the Transform where the wheel collider will be placed. ***Note that this will also be included as a visual wheel, you can manually remove it from the visual wheels array in the VehicleWheel component after generating the vehicle if you want.*** |
| Is Motor | A boolean value that controls whether or not the wheel is a 'driven' wheel that is powered |

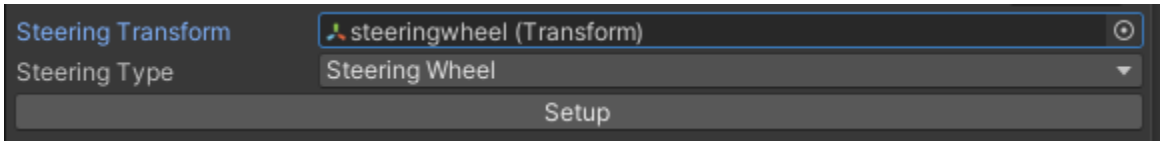
| | |
|---|---|
| | by a motor. If true it is motor-driven, if false it is just along for the ride. |
| Is Steering | A boolean value that controls whether or not this wheel is steerable, otherwise false. |
| Extra Visual Transforms | An array of extra Transforms that should have the WheelCollider's y-position and rotations applied to them. ***For example in the demo car's mesh the wheels have separate GameObjects for the front and back faces of the wheels, so the back face is added to this array.*** |
| Wheel Side | The VehicleWheel.Side the wheel is on. |




**A screenshot of the completed WheelInfo entry for the demo car being set up. Remember in the case of the demo the 'Extra Visual Transforms - Element 0' is a backface for the wheel mesh.**

*e.* *(Optional)* Set the 'Steering Transform' reference to the visual steering object you want to use.

i. After setting the 'Steering Transform' reference, in the case of the demo mesh to the reference shown in the screenshot below, the referenced equivalent Transform will be moved into the vehicle's interior Transform.

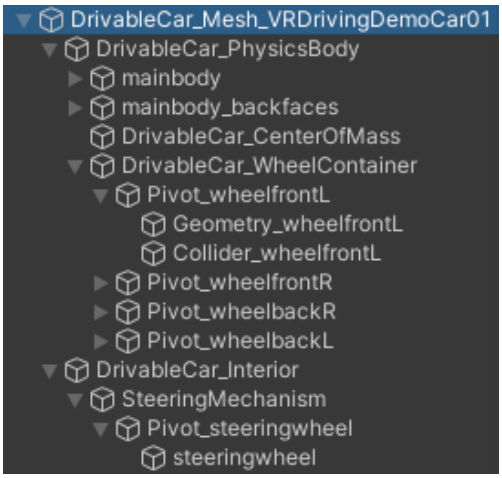



ii. After setting the 'Steering Transform' reference a 'Steering Type' dropdown will appear. If you want to manually set up your steering components select 'None'. ***In the case of the demo we want to use a 'Steering Wheel' so we select it in the dropdown menu.***

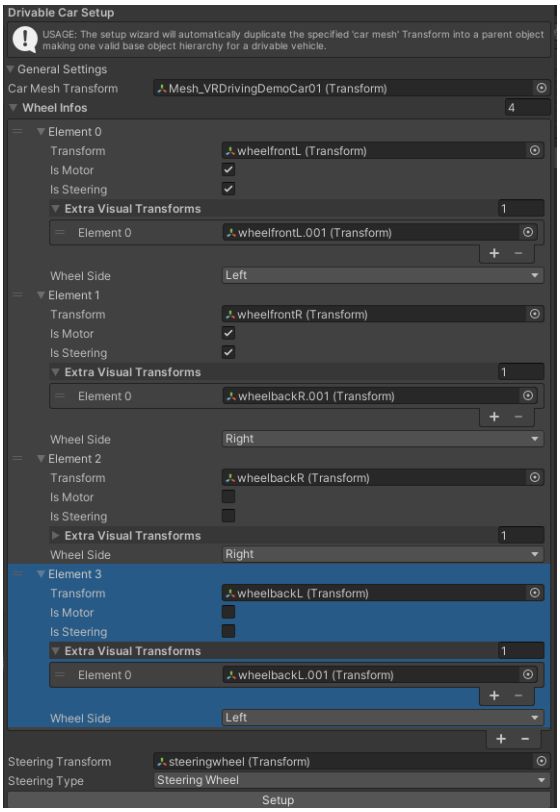
f. ***(Setup Complete)*** Finally, click 'Setup' and the setup wizard will automatically generate a new GameObject for your drivable vehicle. ***You can save the generated object as a prefab if you want.***

- After the set up you will likely need/want to modify settings, ***you can freely modify the generated drivable vehicle as desired.***

- *You will notice that certain objects, like the 'steering mechanism', were automatically put under parent objects with a 'Pivot_' prefix, this is useful for allowing you to adjust any weird rotations to make the 'axis' settings in the steering component you are using easier.*
- *The generated 'DrivableCar_CenterOfMass' object may be moved to change the vehicle's center of mass.*
- *You can now modify the generated VehicleWheel components directly, you can even remove any visual wheel transform (if you want to manually control the visual wheel transformations) or disable rotations or positioning on any axis.*
- ***Below are screenshots showing the hierarchy of the vehicle that was generated by the setup wizard (top) and the final 'Drivable Car Setup' wizard settings that were used (bottom).***



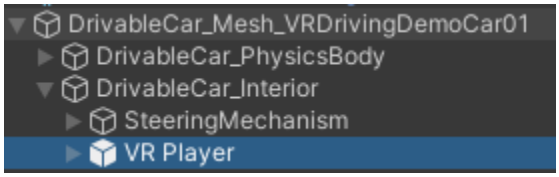

*A screenshot showing the (partial) hierarchy of the generated drivable vehicle.*



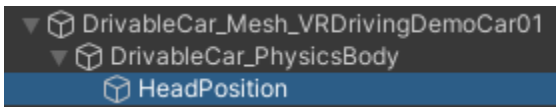
*A screenshot showing the final settings used when setting up the demo vehicle mesh using the steps described above.*

## 3.c. Driving Your Created Vehicle Using The Included Vehicle System

a. To drive your created Vehicle using the included vehicle system first drag the included 'VR Player' prefab into your vehicle's 'DrivableCar_Interior' GameObject as shown in the screenshot below.




b. Next, we need to make (and position) a reference GameObject to set up where the calibrated head position should be. To do this ***create an empty GameObject in the 'DriveableCar_PhysicsBody'*** GameObject, in this case we will name it 'HeadPosition'.

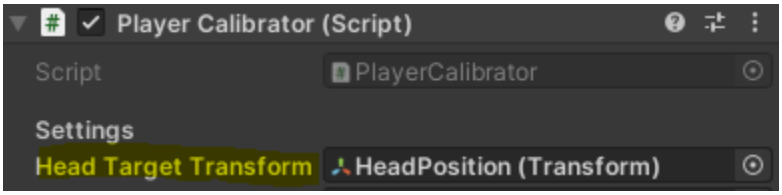



***NOTE: The demo vehicles use a 'Sphere' and a HideOnStart component to better visualize head placement, you can do this too if you want instead of using an empty GameObject.***

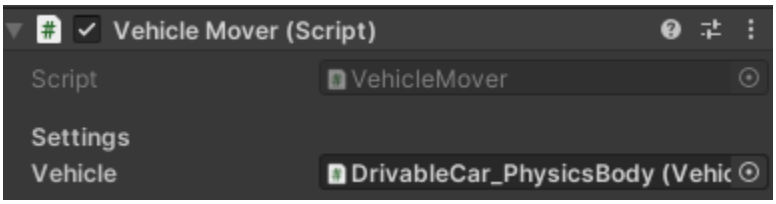
*A sample screenshot showing the 'HeadPosition' GameObject positioned in the 'Scene View' through the front windshield for the demo car.*

c. Lastly we need to set all relevant references in the 'VR Player' GameObject that we added in step (a).

   i. Set the 'Head Target Transform' reference in the PlayerCalibrator component to the newly created 'HeadPosition' GameObject from step (b).



ii. Set the 'Vehicle' reference in the VehicleMover component to the 'DriveableCar_PhysicsBody' GameObject.




iii. You have now completed the basic setup of your own custom vehicle, and set it up so it can be driven in VR! Click the 'Play' button to take it for a test drive.

# Integration

## 4.a. Physics Hand Integration

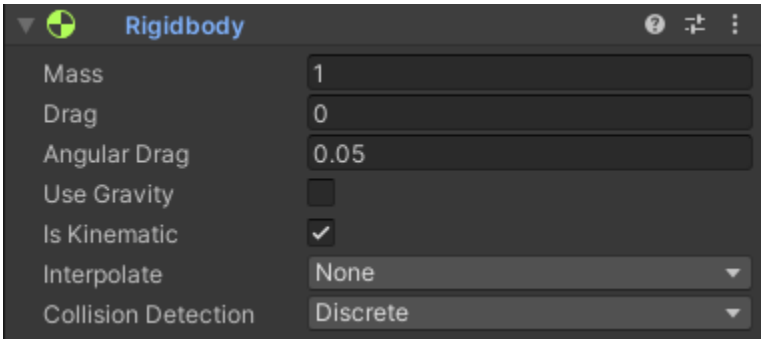
a. Ensure you already own the 'Physics Hand' asset and have imported 'Physics Hand' into your project.
b. Import the included 'Integration_PhysicsHand.unitypackage' file into your project.
c. Check out the 'Integration/Integration_PhysicsHand' folder to see the included C# source file(s) that provide PhysicsHand Integration.
d. Check out the included 'XR PhysicsHand Player' prefab for a reference.

- Physics Hand uses the same underlying grab system as Drive In VR and  because of this it works without any extra steps..

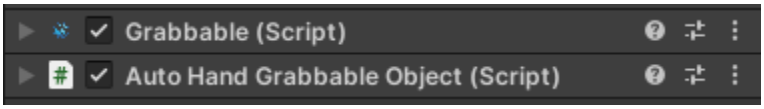
## 4.b. Auto Hand Integration

a. ***Note that the Auto Hand integration demo was made using Auto Hand v3.2***
b. Ensure you already own the 'Auto Hand' asset and have 'Auto Hand' imported it into your project.
c. Import the included 'Integration_AutoHand.unitypackage' file into your project.
d. Check out the 'Integration/AutoHandDemo' folder to see the included C# source files that provide AutoHand integration (useful to reference when adding support for other hand systems) and demo scene(s).
e. **Here are the steps to make your drivable vehicle work with AutoHand:**
    i. Add an Auto Hand 'Grabbable component to the same GameObject that holds the steering mechanism's GrabbableObject.


    ii. Auto Hand Grabbables require a Rigidbody component on the Grabbable to function… simply add a kinematic Rigidbody to your steering mechanism if it does not already have one.




    iii. Add an AutoHandGrabbableObject component as well. The screenshot below shows the 2 components added to the steering mechanism's GameObject.




***You have now completed the setup required to use your drivable vehicle with Auto Hand!***

## 4.c. Adaptive Hands Integration

- 'Adaptive Hands' is included free with 'Drive In VR' in the 'Integration_AdaptiveHands.unitypackage' file included in the 'Integration' folder.
- After importing the integration package check out the provided 'VRPlayer_KinematicHands' prefab that contains 2 fully configured adaptive hands and a VR driving player.


- The documentation for 'Adaptive Hands' can be found here.
- The API reference for 'Adaptive Hands' can be found here.
- Here is a Tutorial Video showing How To Swap the 'VR Player' prefab in your scene.
- For a quick overview check out the provided prefabs and the options in 'Tools → Adaptive Hands' in the Unity editor toolbar.

# Steering Components

## 5.a. The VehicleHandRelativeSteering Component

- Implements kinematic steering wheel control for VR.
- The VehicleHandRelativeSteering component is responsible for kinematically determining the steering wheel angle based on the position of VR controllers grabbing the wheel relative to the pivot point of the steering wheel/
- Like all VehicleSteeringBase components the VehicleHandRelativeSteering component provides a 'SteeringAngleMulitplier' that can be multiplied by the maximum steering radius of your vehicle to give the steering angle for your vehicle's wheels.
    - The VehicleKinematicSteering component provided in the 'Demo' folder provides a code example of how this can be done.
- This component provides two steering modes:
    1. **Offset** - A steering mode that uses offsets from the initial grab location of controller(s) to compute steering.
    2. **Delta** - A steering mode that uses deltas between frames for controller rotations around the steering mechanism to compute steering.

## 5.b. The VehicleHandRelativeHandlebarSteering Component

- Implements kinematic handlebar control for VR.
- Like all VehicleSteeringBase components the VehicleHandRelativeHandlebarSteering component provides a 'SteeringAngleMulitplier' that can be multiplied by the maximum steering radius of your vehicle to give the steering angle for your vehicle's wheels.
    - The VehicleKinematicSteering component provided in the 'Demo' folder provides a code example of how this can be done.

## 5.c. The VehicleWheelchairSteering Component

- Implements kinematic wheelchair movement for VR.
- Like all VehicleSteeringBase components the VehicleWheelchairSteering component provides a 'SteeringAngleMulitplier' that can be multiplied by the maximum steering radius of your vehicle to give the steering angle for your vehicle's wheels.
    - The VehicleKinematicSteering component provided in the 'Demo' folder provides a code example of how this can be done.

## 5.d. The VehicleJointAngleSteering Component

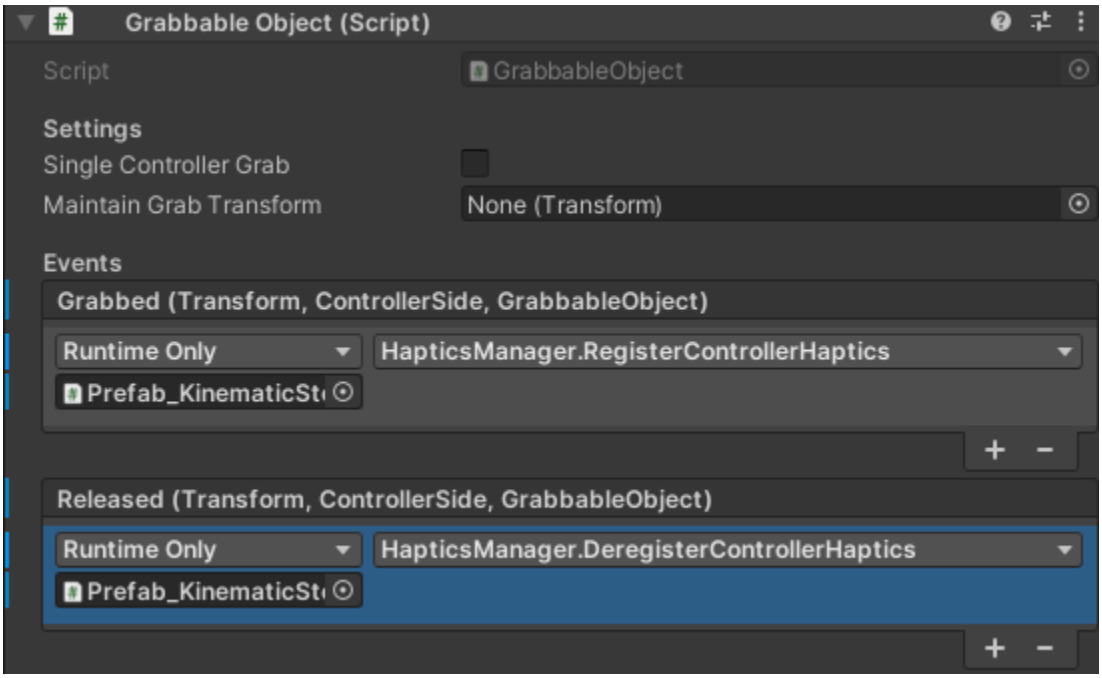
- Implements physics angle reader-based joint angle steering.
- This component requires a physics based hand addon, such as Auto Hand, to work.
- Like all VehicleSteeringBase components the VehicleHandRelativeSteering component provides a 'SteeringAngleMulitplier' that can be multiplied by the maximum steering radius of your vehicle to give the steering angle for your vehicle's wheels.

- The VehicleKinematicSteering component provided in the 'Demo' folder provides a code example of how this can be done.

# Extra Features

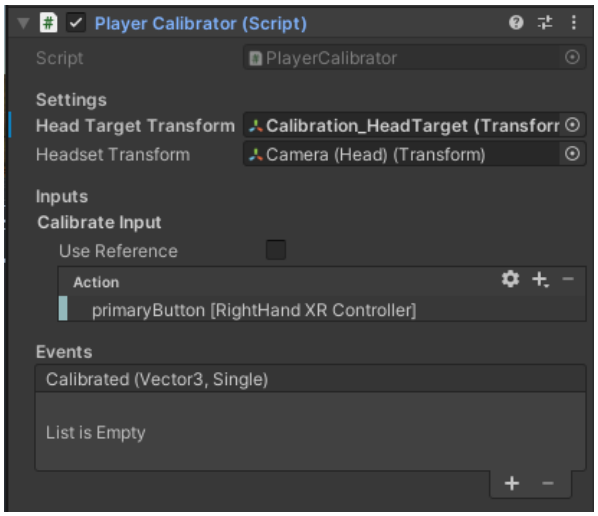
## 6.a. The HapticsManager Component

- The HapticsManager component allows haptics to be played easily through Unity editor events or other components via the scripting API.
- ***For the quickest, simplest setup you can simply add a HapticsManager and PlayHapticsOnCollision component to your vehicle!***
    - For more complex behavior such as only playing haptics on hands grabbing the steering mechanism use Unity events as described below *(or as shown in the vespa & semi truck demos)*.
- The HapticsManager component provides a handy setting to 'register hands on start'. When enabled the manager will automatically register left and right hand XR devices as haptics devices.
    - Alternatively, the component provides useful public methods like the ones shown in the screenshot below that allow you to easily register and deregister haptic devices as a hand grabs or releases a GrabbableObject or via Unity events that use ControllerInfo as their only argument.



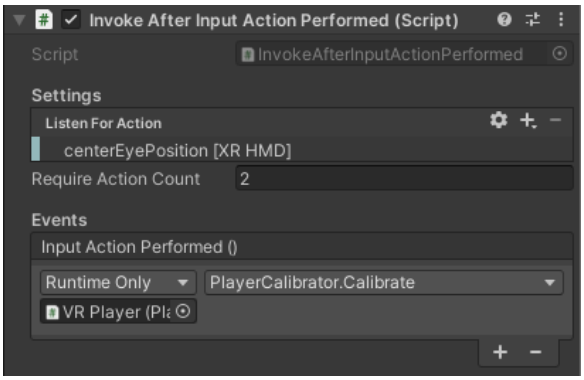

- Use the HapticsManager.HapticImpulse(...) method(s) to perform a haptic impulse via the scripting API or use any of the provided helper components like the PlayHapticsOnCollision component.

## 6.b. The PlayerCalibrator Component




**A screenshot showing the PlayerCalibrator component Inspector pane.**

- The PlayerCalibrator component is responsible for placing the player at the appropriate position, positioning their head to match the 'Head Target Transform' reference and the rest of their body to maintain their offset relative to the head.
- This component should generally be attached to your player prefab.
- **IMPORTANT!**
    - It is generally recommended you use an InvokeAfterInputActionPerformed component to calibrate the player once after the 'centerEyePosition' is received twice. Initially the player's VR headset and controllers will be at world origin, and for a single frame after that… this allows the player to be spawned in the correct spawn initially.




**A screenshot showing the InvokeAfterInputActionPerformed component to initially position the player in the correct spot relative to their 'Head Transform' reference.**

- Below is a table of the fields that make up the PlayerCalibrator component:

| **Field Name** | **Description** | **Default Value** |
|---|---|---|
| Head Target Transform | A reference to the Transform that specifies the position and orientation of the VR player's head | Null ***(Transform)*** |

| | | |
|---|---|---|
| | after they calibrate. | |
| Headset Transform | A reference to the Transform for your VR headset. | Null<br>***(Transform)*** |
| Calibrate Input | A InputActionProperty that allows inputs to be specified as calibration inputs. When a calibration input is fired the VR player's head will be positioned on the 'Head Target Transform' with the rest of the body remaining their offset relative to the head. | Null<br>***(Transform)*** |
| Calibrated | An event that is invoked whenever the player is calibrated. | Null<br>***(Transform)*** |

## 6.c. The DriveHandSwapper Component

- An overridable component that swaps driving controls between controllers whenever something (that is not explicitly ignored) is grabbed by the hand currently controlling the driving controls.

## 6.d. The VehicleSFX Component

- The VehicleSFX component provides engine, braking, and drifting sounds. It also allows you to configure various engine noises based on the current driving conditions.

# Tutorial Videos

## 7.a. Importing Into a Blank Project (~3 minutes)

- A tutorial video that demonstrates how to import 'Drive In VR' into a blank project.
- Youtube Link

## 7.b. Swapping the 'VR Player' Prefab In Your Scene (~2.5 minutes)

- A tutorial video that demonstrates how to swap the 'VR Player' prefab in your scene with another.
- Specifically this demo shows the steps to swap the 'VR Player' prefab in a scene with the 'VR Player - Kinematic Hand' prefab from the 'Adaptive Hands' integration package.
- Youtube Link

## 7.c. Creating & Driving Your First VR Vehicle (~3.75 minutes)

- A tutorial video that demonstrates how to create a drivable VR vehicle from a vehicle mesh.
- Also shows you how to add a 'VR Player' to your drivable vehicle and get driving.
- Youtube Link

## 7.d. Fixing Incorrect Steering Wheel Pivots (~1.25 minutes)

- A tutorial video that demonstrates how to fix an improperly positioned steering wheel pivot point from within the Unity Editor.
- Youtube Link

## 7.e. Grouping Separated Wheel Meshes (~1.5 minutes)

- A tutorial video that demonstrates how to group separated wheel meshes that should rotate with the wheel from within the Unity Editor.
- Youtube Link

# Tuning Your Vehicles Physics

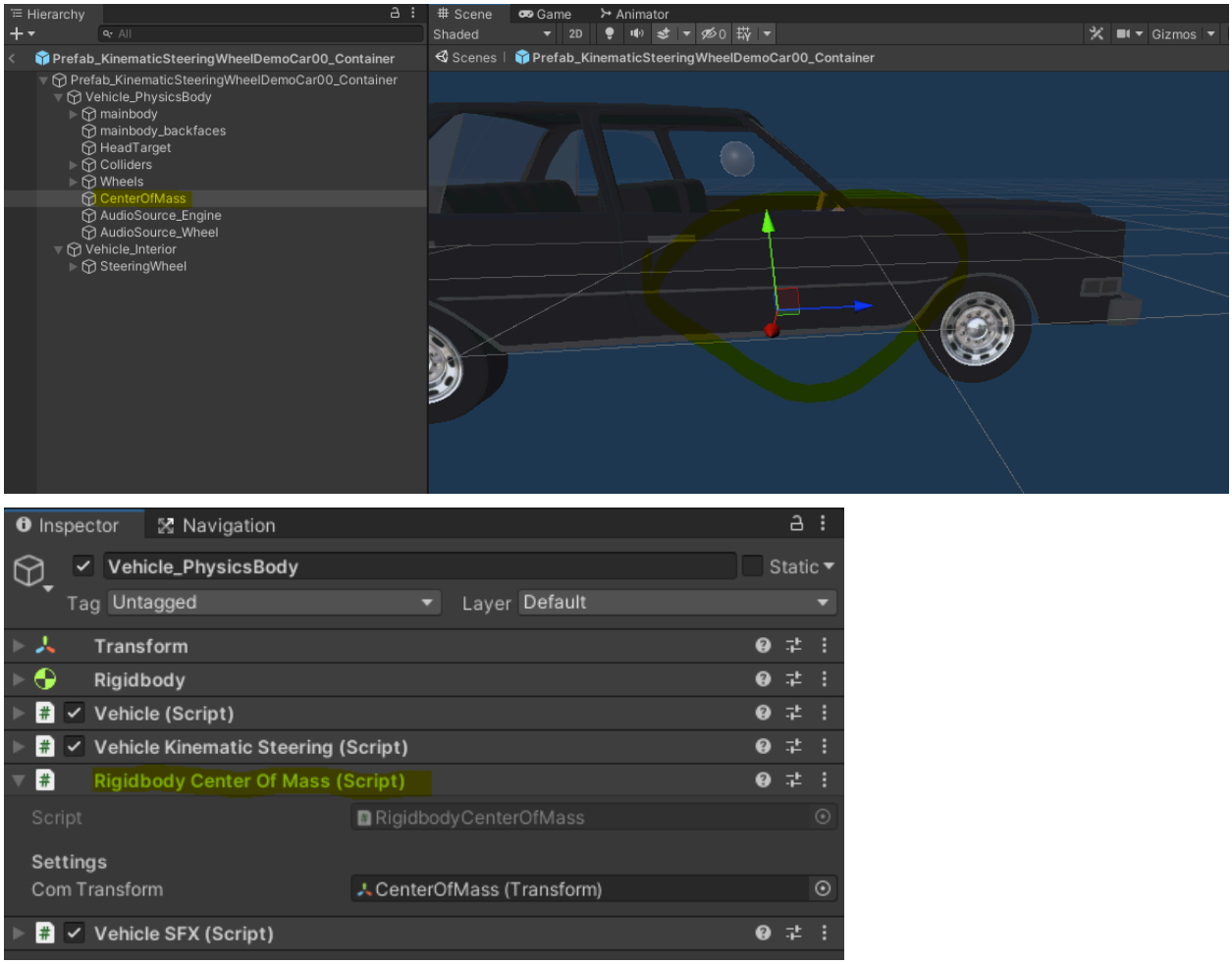
## 8.a. The WheelCollider Component

- Vehicles use Unity's built-in WheelCollider components for the actual 'physics' that drive the vehicle. These are highly customizable and there are many tutorials online for adjusting their 'feeling' and behavior and we recommend googling some of these specific tutorials around WheelCollider components.
- In general though:
    i. The 'Forward Slip' values of the WheelCollider component control forward 'slideyness'/wheel adhesion.
    ii. The 'Sideways Slip' values of the WheelCollider component control horizontal 'slideyness'/wheel adhesion.
    iii. The 'stiffness' affects the wheel's adhesion to the road and is one of the most important factors in controlling 'slideyness' and top speed.


**A screenshot of the Inspector for a WheelCollider component in the Unity Editor.**

## 8.b. The Vehicle's Center Of Mass

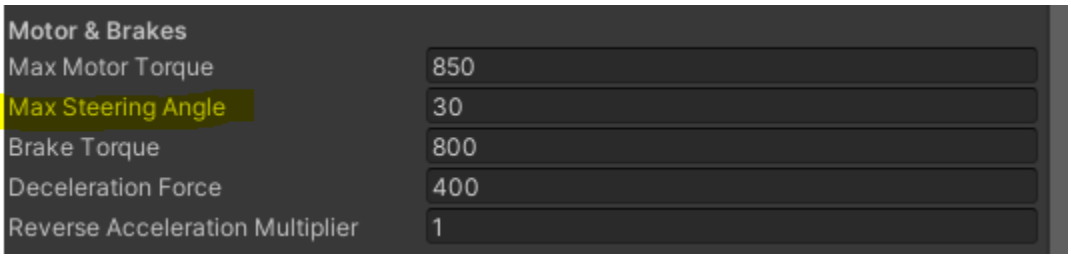
- The vehicle's performance can also be adjusted by modifying the vehicle's Center Of Gravity using a RigidbodyCenterOfGravity component (like used in the demo vehicles) and a reference GameObject. ***This has a huge impact on the vehicle's steering and stability.***




**Screenshots showing the configuration of a custom vehicle center of mass.**

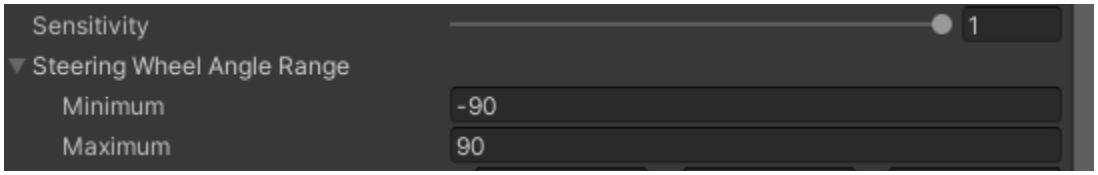
## 8.c. The Vehicle's Steering Settings

- The vehicle's steering radius, or the maximum wheel steering angle (or handlebar steering angle) can also be used (but is recommended to be used as a last fine-tuning resort as these should be more turned around your games style).
- Setting the 'Max Steering Angle' in the Vehicle component allows you to modify how much the steering wheel will be turned (how much the WheelCollider will be turned) when the steering mechanism is fully turned to one side.




**A screenshot showing the 'Max Steering Angle' field in a Vehicle component's Inspector.**

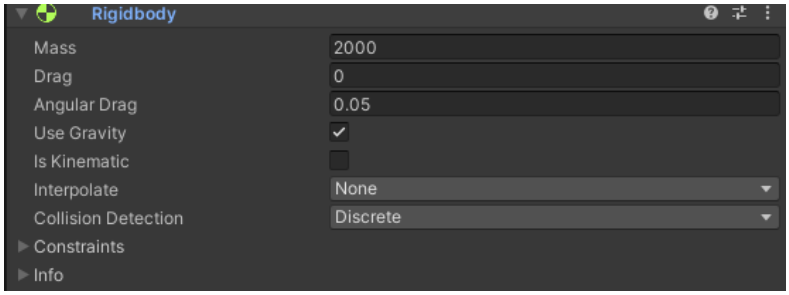
- Most steering mechanism components like the VehicleHandRelativeSteering component have settings such as 'Sensitivity' and 'Angle Range' that allow you to modify input sensitivity and steering mechanism limits easily. This is not generally recommended for adjusting the 'physics' behavior of the vehicle however as these settings more relate to your game.




**A screenshot of some settings from a VehicleHandRelativeSteering component's Inspector pane.**

## 8.d. The Vehicle's Rigidbody

- Finally the vehicle's Rigidbody can be used to adjust components by adjusting the vehicle's mass, drag, and angular drag settings.




**A screenshot of a Vehicle's Rigidbody component in the Inspector.**

# Modules

## 9.a. The Grab System Module

- The 'GrabSystem' module is used for grabbing in the demo content for 'Drive In VR'.
- Documentation: Google Drive
- API Reference: Google Drive

# Preventing VR Sickness

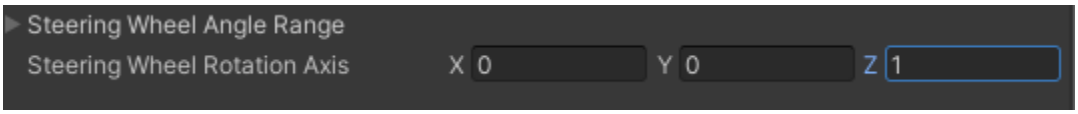
- Unfortunately some VR users are prone to motion sickness. This section of the documentation provides you with some advice on how you can reduce the risk of VR sickness or eliminate it altogether in your game.

1. ***Ensure your game is not laggy.***
    a. A low framerate or stuttering is the #1 cause of VR sickness.
    b. Use Unity's built-in profiler tools to figure out what is causing your project's low framerate or stuttering to fix it.
2. ***Ensure your game has variations in terrain and art content.***
    a. Users tend to experience VR sickness more in large, open, and flat areas with repeating art or a lack of unique reference points.
    b. Increasing terrain height variations can hugely reduce VR sickness for users.
3. ***Reduce the player's view distance.***
    a. Moving over large flat terrains in virtual reality is one of the leading causes of VR sickness. This is because it makes it hard for the player to focus on any unique reference point.
    b. Generally while moving in virtual reality it helps for users to have unique points of reference. Studies suggest having unique reference points largely eliminates VR sickness.
    c. Use things such as fog, hills, foliage, and miscellaneous props to ensure the player can not look too far away.
4. ***Make the player sit higher in your vehicle.***
    a. Studies show that seeing the ground (especially repetitive/similar ground) move by you at a fast rate in virtual reality greatly increases the risk of VR sickness.
    b. Making the player sit higher so they see less of the ground move past in their peripheral vision greatly reduces VR sickness for many people who experience it.
5. ***Reduce the 'slideyness' of your vehicles.***
    a. Making it more difficult for a vehicle to lose traction can help reduce VR sickness for users. See the 'Tuning your Vehicle Physics' section

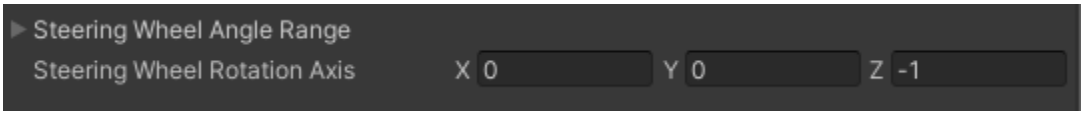
# FAQ

(Frequently Asked Questions)

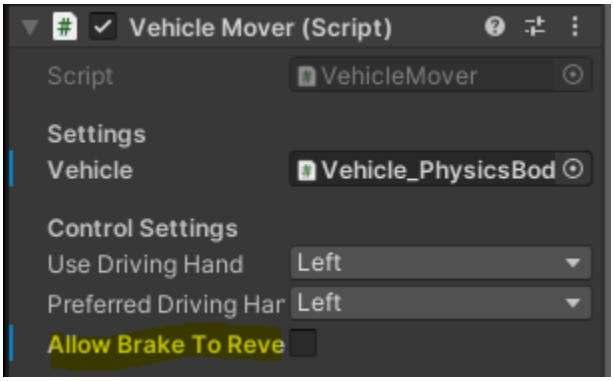
1. My custom steering wheel setup turns the opposite way of the vehicle but the steering still seems to work normally when using the VehicleHandRelativeSteering component. How do I fix this?
    b. This is caused when you have the incorrect signedness 'Steering Wheel Rotation Axis' setting. For example in the below screenshot '0, 0, 1' would be changed to '0, 0, -1'.
        i. Changes From:




        ii. To:



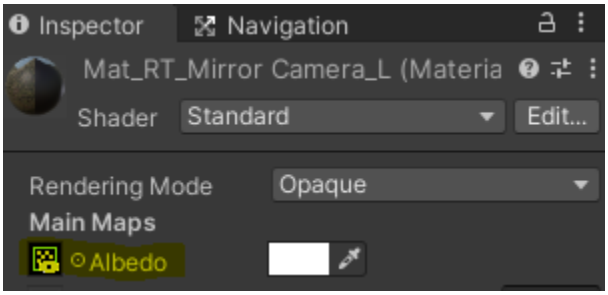

2. How can I disable using the brake button to reverse in favor of a gear shifter?
    - In your player prefab (or the demo player prefab) you will find a VehicleMover component as shown in the screenshot below. Simply uncheck the 'Allow Brake To Reverse' checkbox.




    - The gear shifter demo can be found in the '02_Demo_Semitruck' scene provided with the package.

3. How can I add working cameras & rear view mirrors like seen in the '02_Demo_Semitruck' demo scene?
    - To do this we made use of render textures and Cameras.
    - Check out the Unity documentation surrounding this as well as the '02_Demo_Semitruck' demo scene to see how we implemented this.
    - The basic steps are as follows:
        a. Right click in the 'Assets' pane and click 'Create → Render Texture'.
        b. Create a new Camera that will render to your newly created 'Render Texture', reference the 'Render Texture 'in your Camera by setting the 'target texture' field.


        c. Create a Material that uses your 'Render Texture' in the albedo channel.



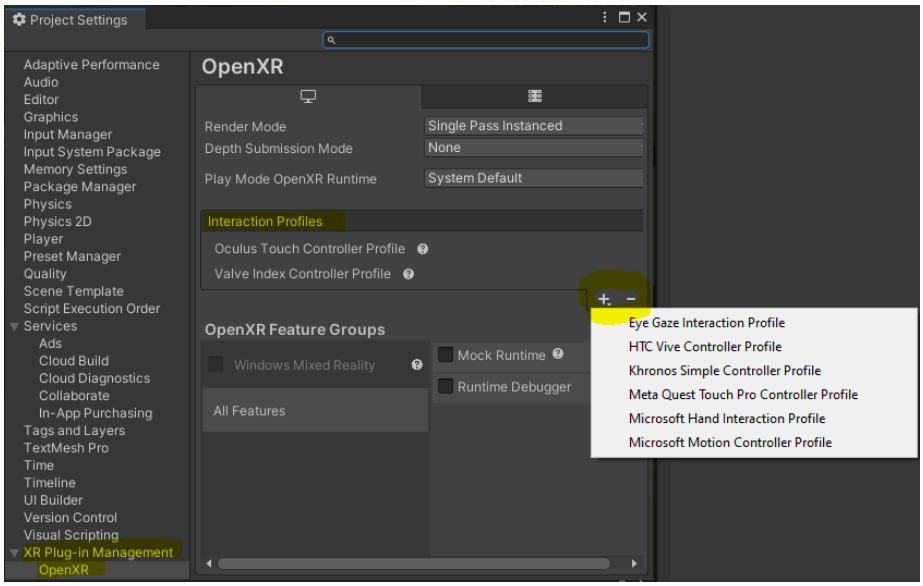

        d. Simply apply your Material to whatever you want to render your camera to.

4. The vehicles feel too 'slidey', or not 'slidey' enough for my liking. How can I adjust this?
    a. Vehicles use Unity's built-in WheelCollider components for the actual 'physics' that drive the vehicle. These are highly customizable and there are many tutorials online for adjusting their 'feeling' and behavior and we recommend googling some of these specific tutorials around WheelCollider components.
    b. In general though:
        i. The 'Forward Slip' values of the WheelCollider component control forward 'slideyness'/wheel adhesion.
        ii. The 'Sideways Slip' values of the WheelCollider component control horizontal 'slideyness'/wheel adhesion.
        iii. The 'stiffness' affects the wheel's adhesion to the road and is one of the most important factors in controlling 'slideyness' and top speed.

    c. The vehicle's steering radius, or the maximum wheel steering angle (or handlebar steering angle) can also be used (but is recommended to be used as a last fine-tuning resort as these should be more turned around your games style).
    d. The vehicle's performance can also be adjusted by modifying the vehicle's Center Of Gravity using a RigidbodyCenterOfGravity component (like used in the demo vehicles) and a reference GameObject. ***This has a huge impact on the vehicle's steering and stability.***
    e. Finally the vehicle's Rigidbody can be used to adjust components by adjusting the vehicle's mass, drag, and angular drag settings.
    f. View the 'Tuning Your Vehicles Physics' section for more details.

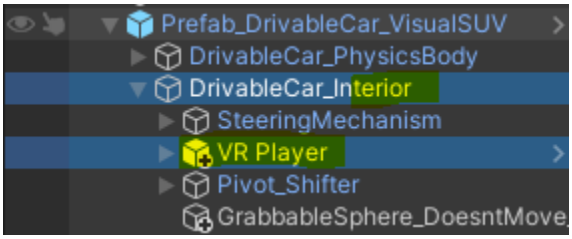
5. My VR driving inputs are swapping hands, how can I stop this?
    - This happens when you are using the provided DriveHandSwapper component. This component allows the driving hand to be automatically swapped using Unity events or through the scripting API and allows you to swap driving inputs to allow the 'free' hand to do things like shoot a gun, or perform other actions that require their inputs while driving.
    - To fix this simply remove the DriveHandSwapper from your player, it is not necessary.

6. When I grab the steering wheel and move the controllers it does not turn.
    - Users have reported this occurring when they are missing the relevant 'Interaction Profile' for their XR device.
    - Navigate in the Unity Editor Toolbar to 'Edit → Project Settings → XR Plugin-in Management → OpenXR' and use the '+' or '-' button for the 'Interaction Profiles' field in the relevant window.




**A screenshot showing the 'Interaction Profiles' setting in 'Project Settings' for OpenXR.**

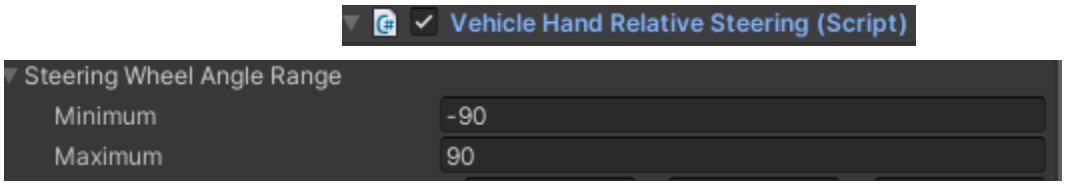
7. When I drive the player/hands do not follow the vehicle.
    - This occurs when you have accidentally placed the 'VR Player' GameObject as a child of the wrong object.
    - Your 'VR Player' GameObject should be a child of the 'interior' GameObject for your vehicle, in the case of the provided demo SUV this is the 'Prefab_DrivableCar_VisualSUV'.
    - The screenshot below shows the proper hierarchy location to place the 'VR Player' GameObject when using the demo SUV:



*A screenshot showing the 'Interaction Profiles' setting in 'Project Settings' for OpenXR.*

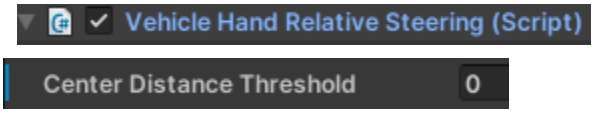
- Optionally see this Youtube Tutorial on Swapping the 'VR Player' Prefab In Your Scene.

8. How can I reduce the sensitivity of the VehicleHandRelativeSteering component?
    - While you cannot directly set the sensitivity of the VehicleHandRelativeSteering component as it would cause desyncing between the real-life and XR hand positions the best way to ***reduce steering sensitivity*** is to simply increase the minimum and maximum 'Steering Wheel Angle Range' setting of the VehicleHandRelativeSteering component.
        - Increasing this value will make a car handle more realistically, decreasing this range will make a car handle with a more slidey/arcadey style as it takes a smaller turn of the steering wheel to turn the vehicle more.




- For more information check out "The Vehicle's Steering Settings" section of this documentation.

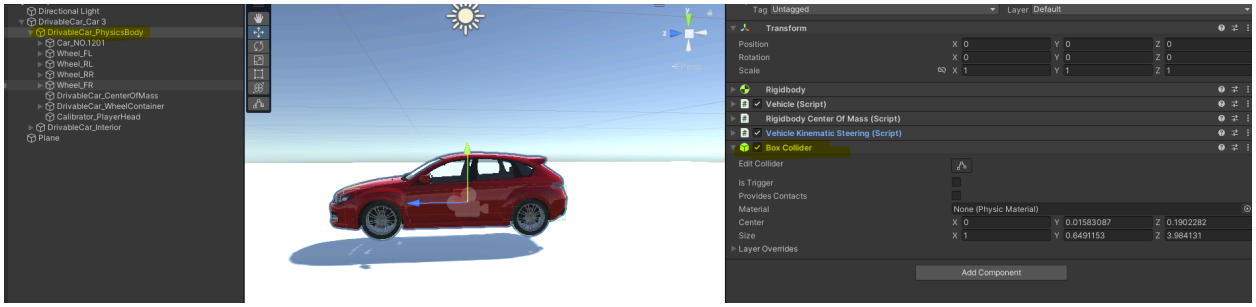
9. How can I prevent an abrupt turn when an XR controller drifts past the center-point of a steering wheel?
    - Although there is no perfect way to handle a player accidentally moving their hand past the steering wheel by accident you can give the user some leeway by using the 'Center Distance Threshold' setting of the VehicleHandRelativeSteering component to simply ignore steering inputs from XR devices that are too close to the center-point of the wheel.



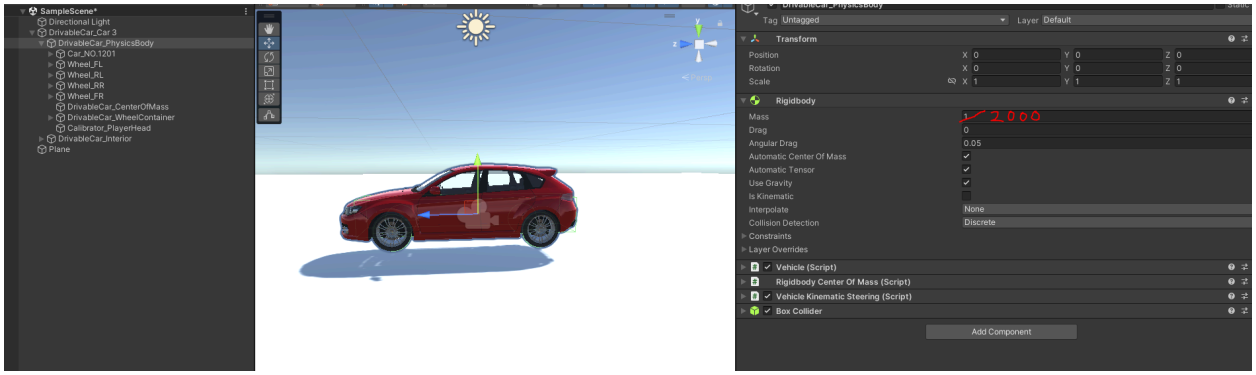

    - The 'Center Distance Threshold' setting is the minimum number of distance units the XR device must be from the steering wheel center-point before steering inputs from the XR device are included in the steering calculation.

## 10. My vehicle is bouncing around uncontrollably when entering 'Play' mode.

- This is most often caused by one of two things:
    a. Your vehicle has no Colliders. Add at least one Collider (like a BoxCollider) to the Rigidbody or its children.



    b. Your vehicle has too light of a mass. In some versions of Unity the mass is forced to '1'. Ensure the mass is higher (like 2000 for example).



.

## 11. My vehicle's steering wheel rotates around the incorrect pivot.

- This issue happens when the 'Pivot' point of your car mesh is incorrectly placed as illustrated by the screenshots below:

*A screenshot showing an incorrectly placed steering wheel pivot point.*

*A screenshot showing an correctly placed steering wheel pivot point.*

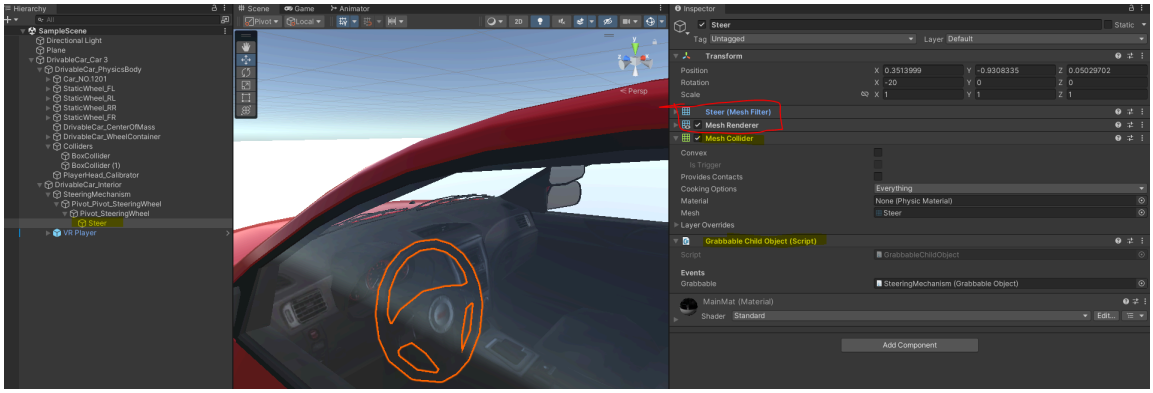
- You can fix can fix an incorrectly positioned steering wheel pivot by following the steps specified in the video here: https://youtu.be/qUySRUq0cmQ

## 12. Not all of, or too many of my vehicle's wheel parts rotate with (or do not rotate with) the tire.

- Ensure that your vehicle's rotating wheel meshes are correctly grouped under a single parent GameObject as shown in the video here: https://youtu.be/POhuMFzRkz0

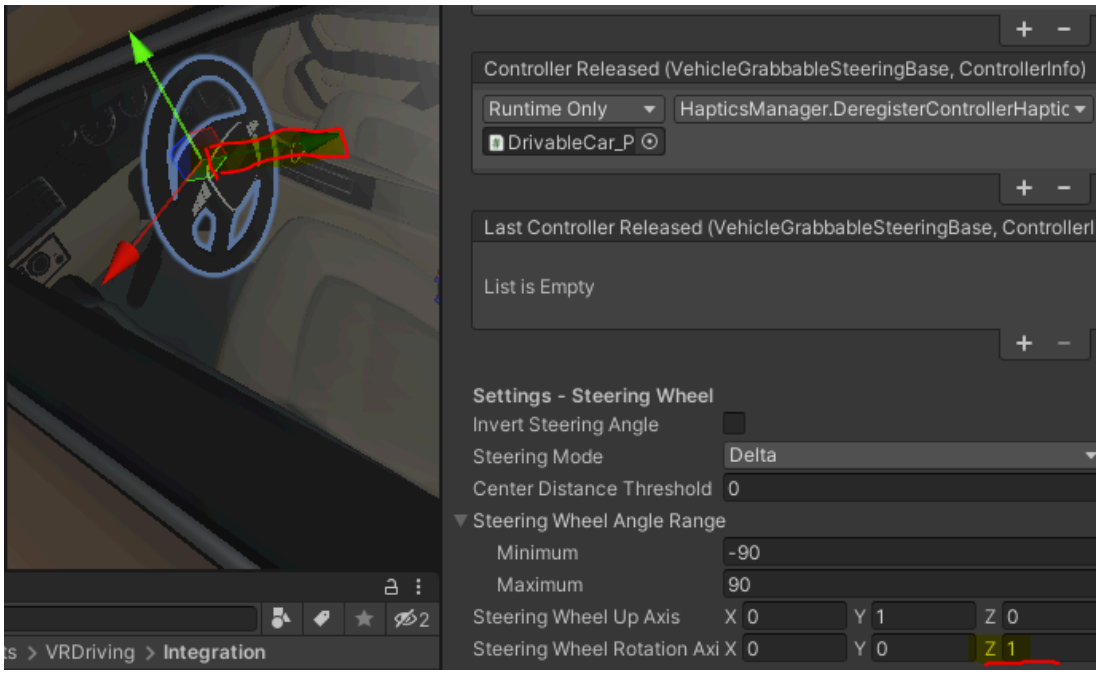
## 13. I cannot grab my steering wheel.

- Ensure that you've added a Collider (such as a MeshCollider) and a GrabbableChildObject component to your steering wheel's Mesh as shown in the screenshot below:



## 14. My steering wheel pivots weirdly when steering.

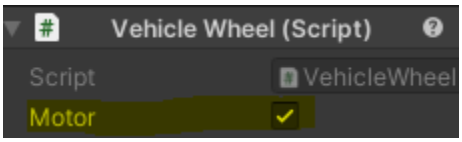
- Ensure your steering wheel's pivot point's 'rotation axis' is aligned straight through the steering wheels center. The axis that defines the steering wheel rotation is 'Steering Wheel Rotation Axis' in the VehicleHandRelativeSteering component.
- If your steering wheel's pivot point does not match like the screenshot below (or at least if the 'Steering Wheel Rotation Axis' doesn't match the visually expected pivot arrows see this youtube tutorial on fixing it from within the editor.



## 15. The VehicleSFX component is only playing the idle sound on my custom vehicle.


**A screenshot of the 'Torque Ref Wheels' field in the VehicleSFX component Inspector in the Unity Editor.**

- This happens when there are no 'Torque Ref Wheels' referenced in the VehicleSFX component.
- You can resolve this by dragging the WheelCollider components from motor-enabled wheels of your vehicle (at least 1) into the 'Torque Ref Wheels' setting of the VehicleSFX component.




**A screenshot of the 'Motor' setting enabled on a VehicleWheel, the child WheelCollider of these wheels should be referenced in the 'Torque Ref Wheels' setting.**