For questions or problems join discord: https://discord.com/invite/sEWwDGr

Instagram : https://www.instagram.com/ash_dev18

Watch the Demo Video first. From asset store.

Basics :

You can drag and drop any of the vehicle prefab from prefab folder in your scene and swap the body and tire meshes and adjust colliders and use it.

For vehicle customization use vehicle editor script and car controller script which is there on the main parent of vehicle gameobject. Change their parameters for your need.

All the scripts can be customized and you can modify them.

Vehicle editor script only works on 4 wheels so first adjust 4 wheels and the suspension and damper force then remove both the vehicle editor script and references script. The duplicate the wheels and position them manually.

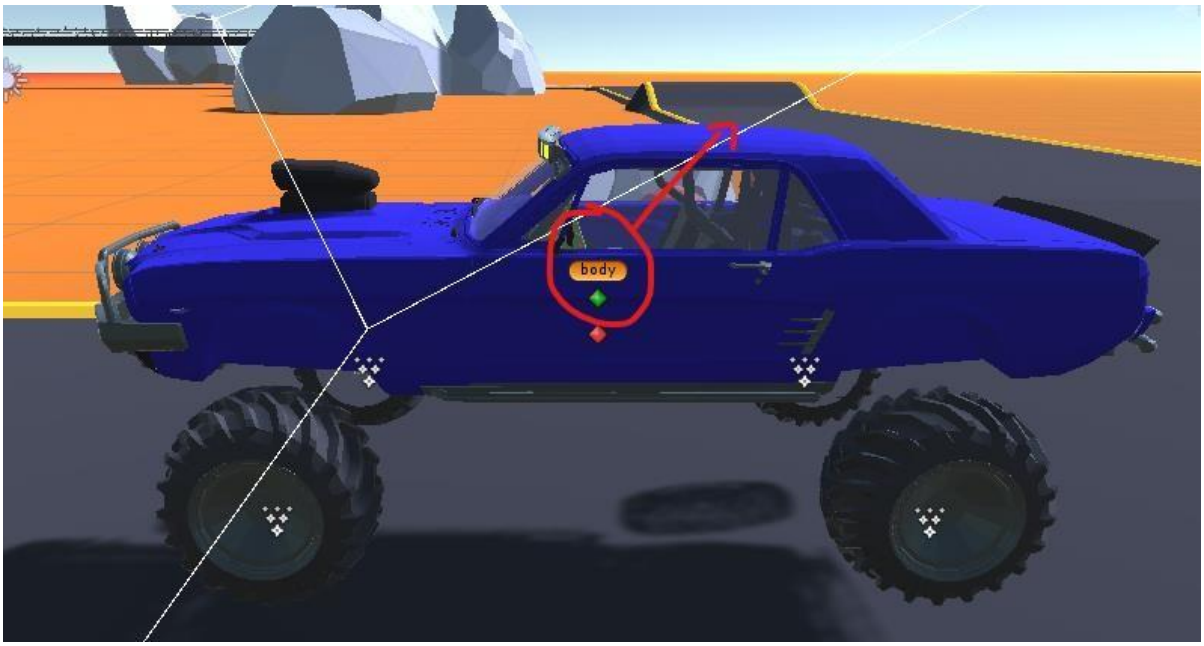
1.**Implimenting on other models**:

Download some vehicle models from internet. Then select any of the vehicle prefabs, from the prefab folder that you like, and want similar physics behaviour. You can also choose from the presets folder. Then drag and drop it in the scene. If your model is too small, or big, scale it to reasonable size. If you are going for RC racing or something, then you can use a small size, but then, you also have to reduce the mass of the body, and wheels, and the values in the car controller script.

Then you have to replace the meshes. For vehicle body, You have to replace the game object under mesh body and for wheels you have to replace the game object under WmeshFL, WmeshFR, WmeshRL, WmeshRR respectively. Replace and then set their position to 0,0,0 . Don't worry if your meshes are not in correct position For now.

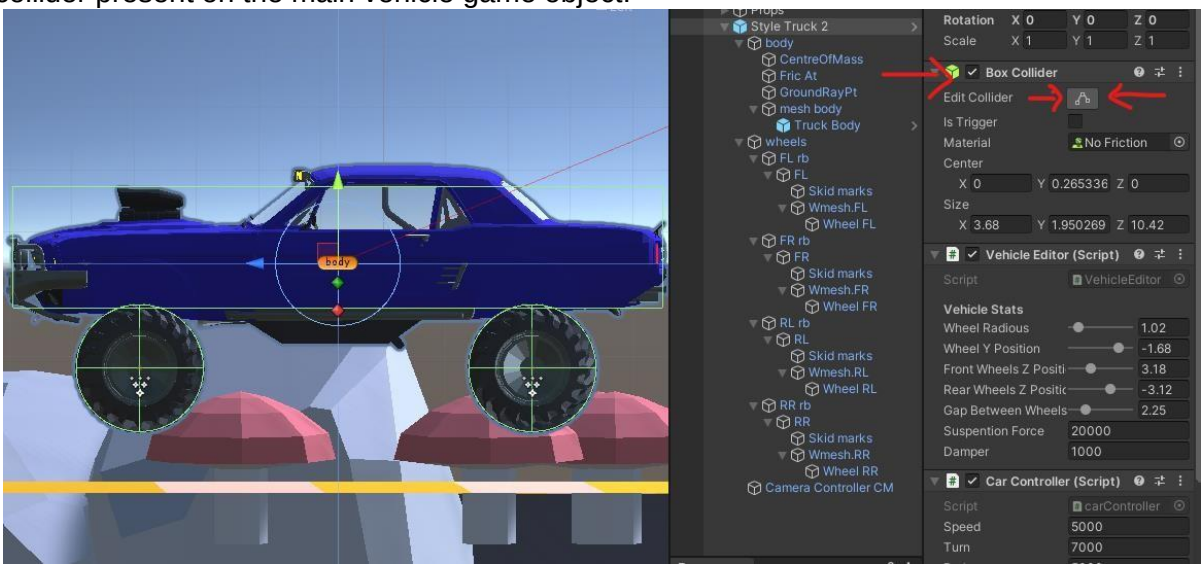
Figure 1 Structure of vehicle Gameobject

Once you replace them, select mesh body game object, and look for body name in the scene view.

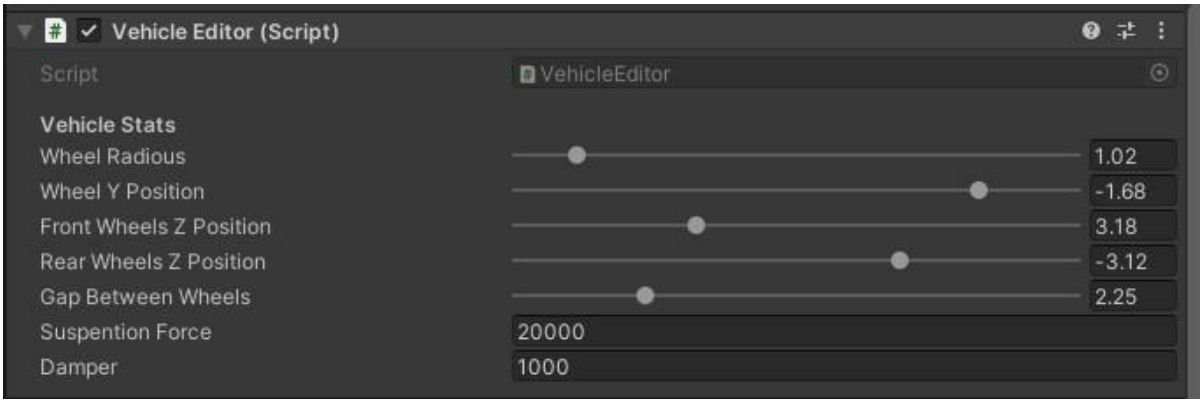


If you don't see it, enable gizmos and increase 3d icon size. Then Aline mesh body approximately at the centre of body icon. By going into isometric side view. So that your vehicle body mesh is approximately at the centre of body icon.

Then you can adjust colliders. You can also use a mesh collider for the body but it might affect performance, if your model is high poly. so, i recommend, adjusting the existing box collider present on the main vehicle game object.

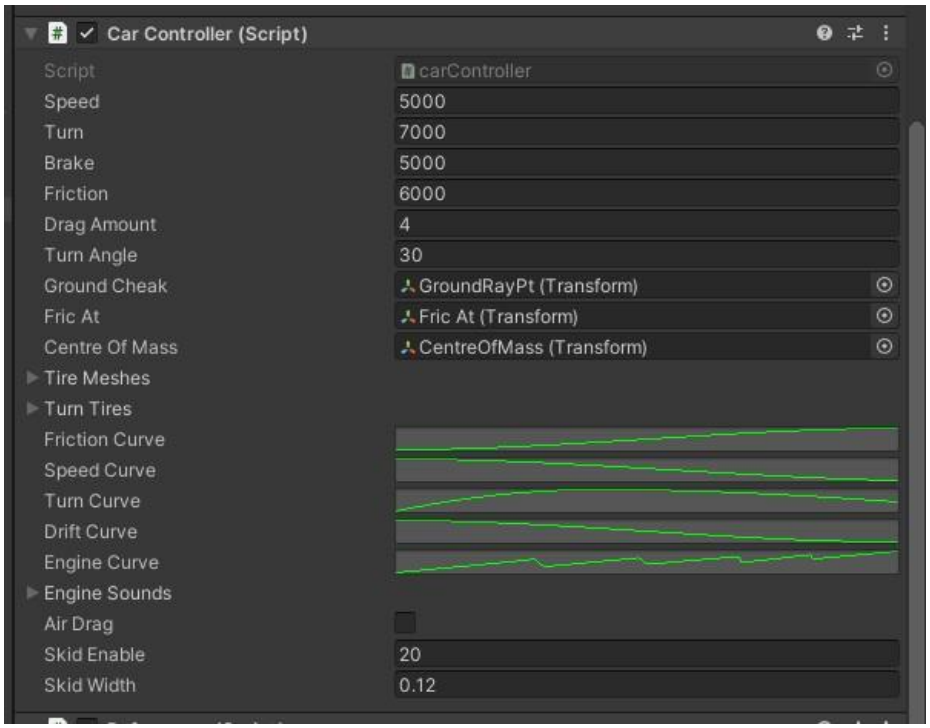


For wheels you don't have to set colliders one by one.



You can adjust the collider radius by vehicle editor script, present on the main vehicle game object. Wheel radius is the radius of sphere colliders, of wheels. You can expand the inspector window to get fine control on slider values, or manually put the values. Wheel y position is the position of wheels in y axis. Other values are also self explanatory. You can change them and adjust the position of wheels. You can also change the suspension and damper force from here. Basically the suspension force is the spring force. Once you adjust all of this, you can test the vehicle. There are still some things to adjust.

2**. Car Controller Script**:



First take a look at **curves**. These curves takes velocity of the car as an input, and gives the corresponding outputs. Lets see the **speed curve**. Open it by clicking on curve.

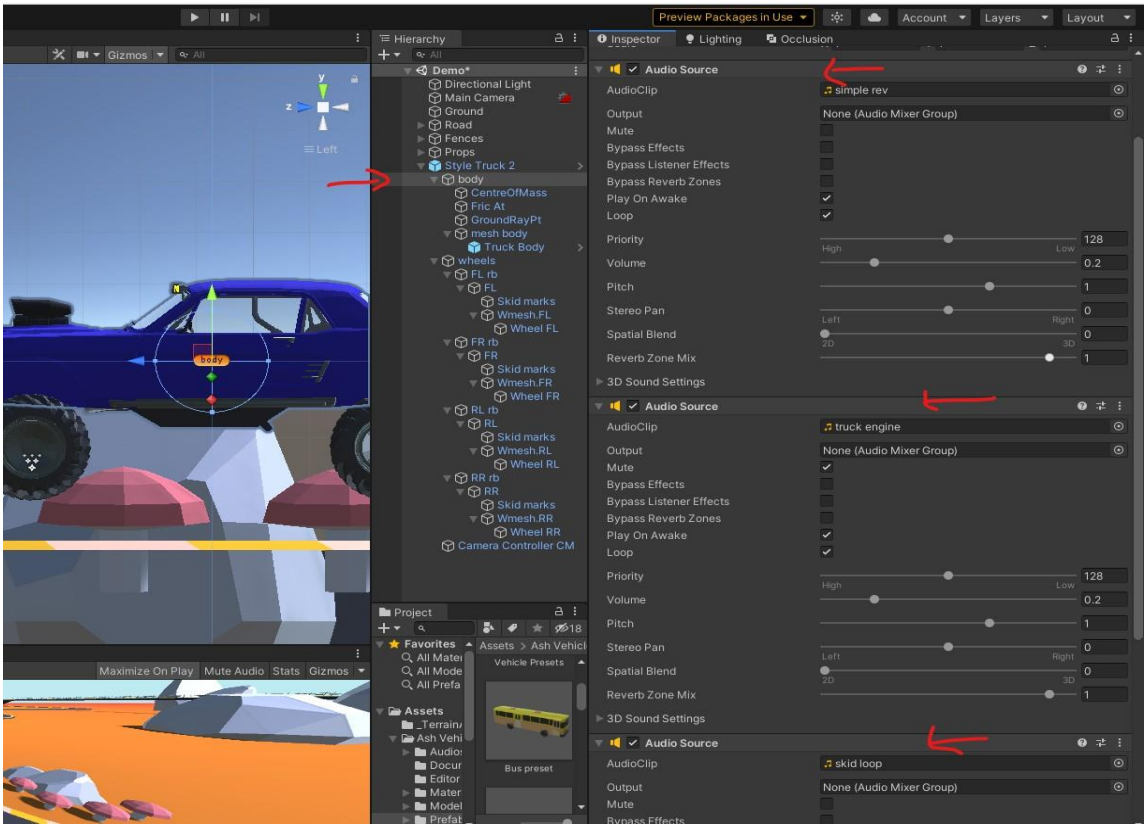
The horizontal axis is velocity of car and the vertical axis is the force, that should be applied on the car. In general, you don't want your vehicle to continuously gain speed. That's why the curve goes down as the velocity increases. The speed variable on the top, acts as a multiplier, That means when the curve value is 1, then the force value is same as the speed variable. So basically this speed value is not speed, its represents the force, but for convenience I named it speed. Because if you visualise it, it also represents the max speed. So this was the **speed curve**.

You can also edit this curves to get different behaviours. The other curves are also similar.
For **friction curve**, the multiplier value or you can say max value, is the **friction** value.
For **turn curve**, it is the **turn** value.
For **drift curve**, it is the **drag amount** value, and this curve determines the angular drag according to the speed. You don't need to change it. But if you want, you can.

Engine curve represents the pitch of engine sound according to the velocity, And the **spikes** in the curve represents the gear shift sound. You can change the gap between two gear shifts. You can also make the curve straight if you don't want gear shift sound.

If you expand **engine sounds**, you will see 3 audio sources.

Click on any of it. A game object will pop up (Named body). Click on the game object and you can see the 3 audio source components.

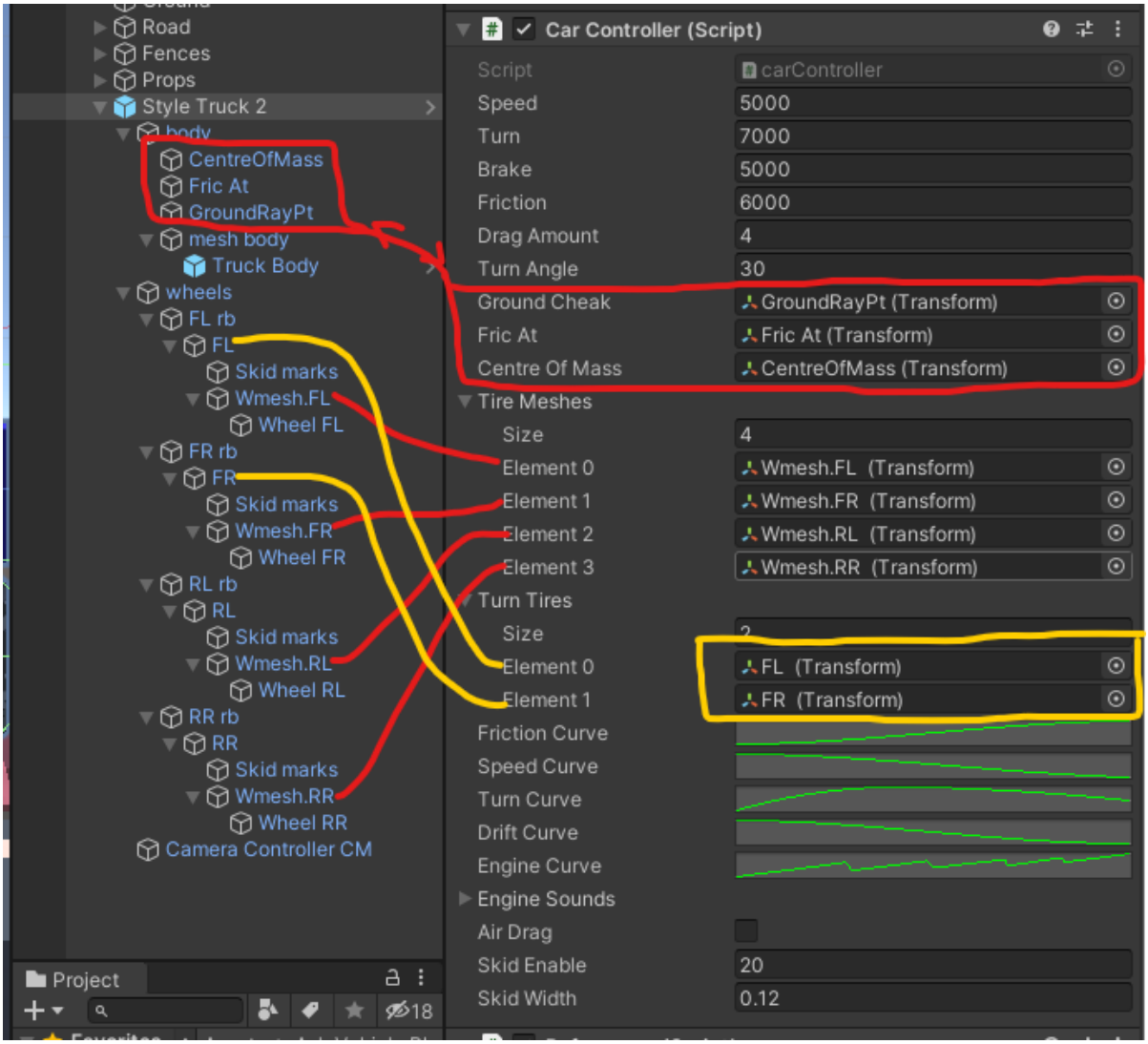


Here you can adjust the volume and all things of individual audios. The first 2 audios are engine sounds. And the last one is the skid, or drift sound. For engine sound, you can combine 2 audios, or mute one to play just one. You can change the audio file. You can use the provided sounds, or you can use your audio files. But they should be loopable.

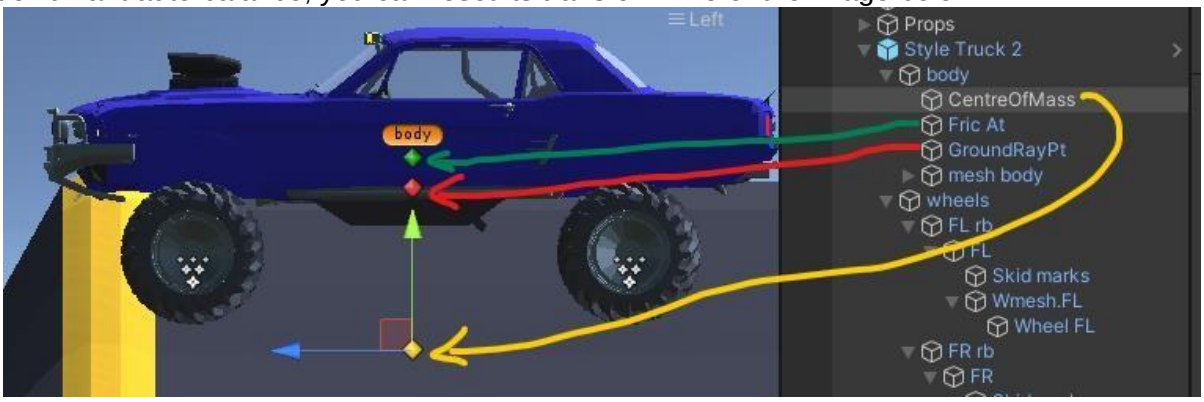
Turn angle(refer first image) is, at what angle the front tires should turn.

Air drag- whether you want drag in air or not. (enabling it will result in more realistic behaviour in air).

Now comes the relevant part.



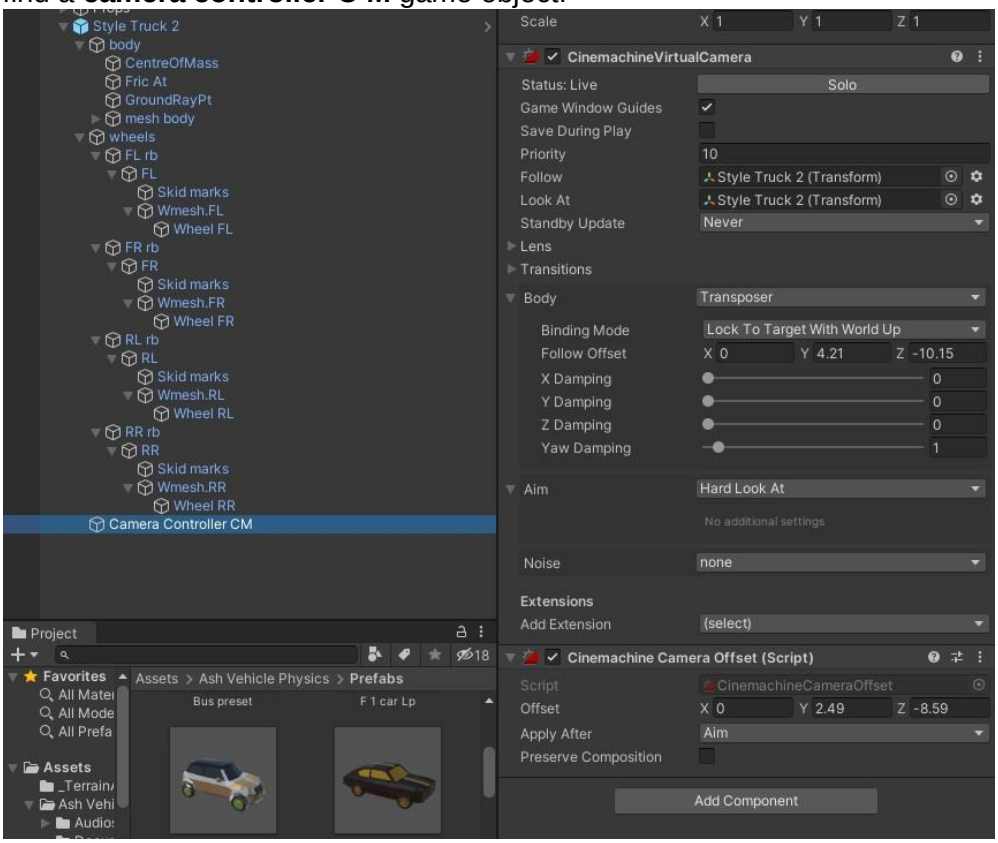
Ground check is the point from which, a ray will be shot in downward direction and it will check whether the vehicle is grounded or not. So this point should be above the wheels. But, it is also the point, where the accelerating and braking force will apply. So if you place it slightly below the **centre of body**, you will get this accelerating and braking motion. Next is **friction at**. This is the point where the sideways friction should be applied. Again, if you place this slightly below the centre of body, the vehicle will lean into the turns. But don't place it too much below the centre. As it can cause the vehicle to roll, or flip while turning. **Centre of mass** is the point where the centre of mass should be, when the car is in the air. So what happens here is, if you place this below the car, it works like auto balancing. If you don't want auto balance, you can reset its transform. Refer the image below.



tire meshes are for all wheel meshes, to rotate according to the speed. And turn tires are for front tires to turn. Each wheel has this many child game objects. So if you want more than 4 wheels you can duplicate the wheels and add their corresponding childs in **tire meshes**, or **turn tires** Array. For rotating tires according to speed, add **WmeshXX** (XX= FL, FR, RL, RR) game objects to **tire meshes** Array, and for turning front tires according to turning input, add **F L** or **F R** game object to **turn tires**. But do this after you implement the vehicle for 4 wheels because, the **vehicle editor script** Only supports 4 wheels. So first implement for 4 wheels, then duplicate the wheels, then do the steps. *And this steps are relevant to you only if you want more than 4 wheels. Also refer to the example prefabs.* There are some examples in the prefabs folder for more than 4 wheels.

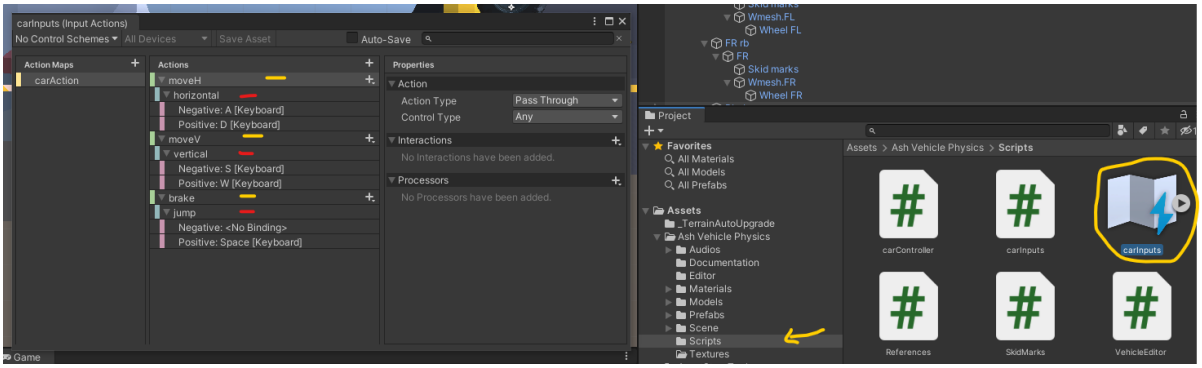
Skid Enable(in carController script) sideways velocity at which the skid mark and smoke starts.
Skid width is the width of skid marks.

Now I will quickly tell you about camera control and input system. For the camera you will find a **camera controller C M** game object.



You can set, how the camera should behave from here.

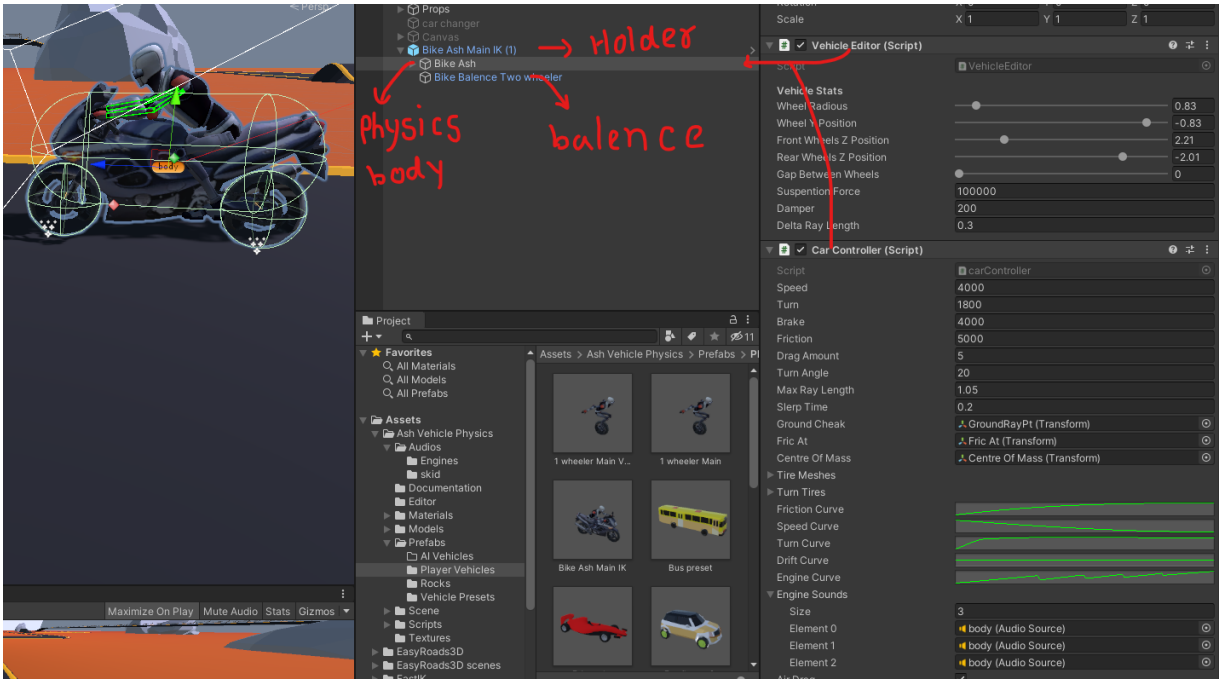
And for inputs, go to the **scripts folder** and open **car inputs**.



There are some actions here. Move h is the turning input. move v is the acceleration input. And brake is brake input. You can change the input buttons from here. This input system supports all types of inputs. So you can build the game for pretty much all devices such as p c, mobile, x box. Read unity documentation for more information about it.

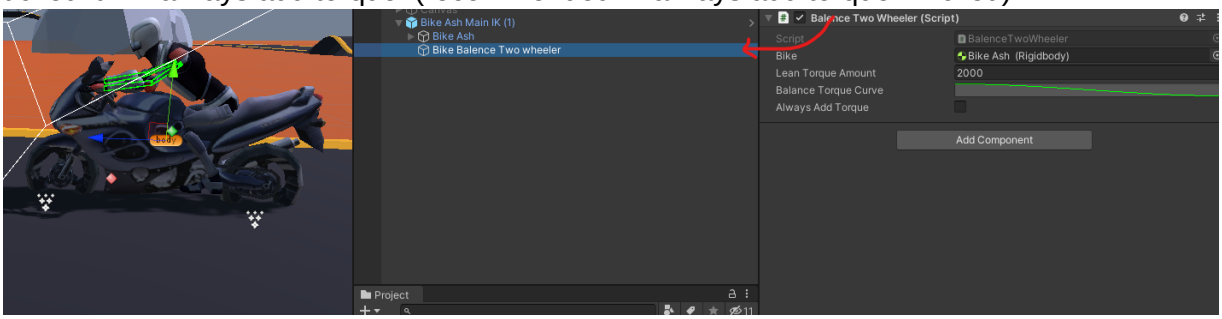
Bike (one and two wheeler)

All the things are same but the Structure of bike is same. The main parent gameobject of the bike is just for holding the bike and the balancing gameobject. So the physics body of bike is under the main gameobject of the bike. The scripts are attached to the gameobject under the main parent gameobject(named Bike Ash). This is the structure for the bike. So for changing physics of the bike you have to change the scripts values (same as above).



The 2nd gameobject under the main parent gameobject (Bike Balance Two Wheeler or One Wheeler) has a script attached which helps in balancing the vehicle. But you don't have to change anything in it as long as you don't change the mass of the vehicle. This script applies a torque when the bike leans in some angle opposite to the lean angle, to make the bike

stable. If Always Add Torque is not ticked , it will all torque only if the vehicle is grounded.If ticked it will always add torque. (recommended :– always add torque : Ticked)



This asset is for vehicle physics. To simulate the bikers hand body movements, you can use any Inverse Kinematics solution. IF you install the Fast IK package (free) from the asset store it will be better as I already implemented it in this package. So install it before installing my package as it can cause some errors. Or you can use other solution.

Gravity for Vehicles

There is a script in scripts folder called Custom Gravity. You can add this to any rigid body and the body will use the gravity stated in the script, not the unity's gravity.
For prefabs this script is added to the vehicle body and the wheels.