Arcade Vehicle Controller

From Enigma 23
v1.8

Table of Contents

Arcade Vehicle Controller (AVC) is a simple vehicle controller asset allowing you to create any type of vehicle – car, hover car, motorbike, boat. The customizable arcade-y feel to the vehicles gives a fun, satisfying, player experience to your vehicles.

Included is a Vehicle Builder editor window, which allows you to setup a new vehicle in seconds! Just link the required objects and fill in the details, press a button and you've got your new vehicle!

The values for a vehicle (speed, turning, drifting, etc.) are saved in a ScriptableObject, meaning you can re-use the same values for hundreds of different vehicles, and only have to tweak them in 1 place.

Also included, are examples of how to interact with the vehicle controller for player input and AI.

What's Included

/Demo/ - In this folder is extra items that are used purely for the demo scene. Once you no longer require the demo scene, you may delete this folder without any issues.

/Examples/ - This folder is important, it holds examples for various parts of the asset for you.
/Examples/Effects/ - Quick, simple effects used to demonstrate smoke trails and tire trails from the vehicles. They're here to show setup, they're not production quality assets.
/Examples/VehicleModels/ - FBX files for the vehicles, and their prefabs with the correct hierarchy setup that the asset requires.
/Examples/VehiclePrefabs/ - Premade prefabs of each vehicle type. These are used in the demo scene. Use these as reference if you encounter any issues with your own vehicles.
/Examples/VehicleSettings/ - These ScriptableObjects hold the data for each example vehicle type. A vehicle requires a vehicle setting.

/Materials/ - Physics material used on the vehicle's collider.

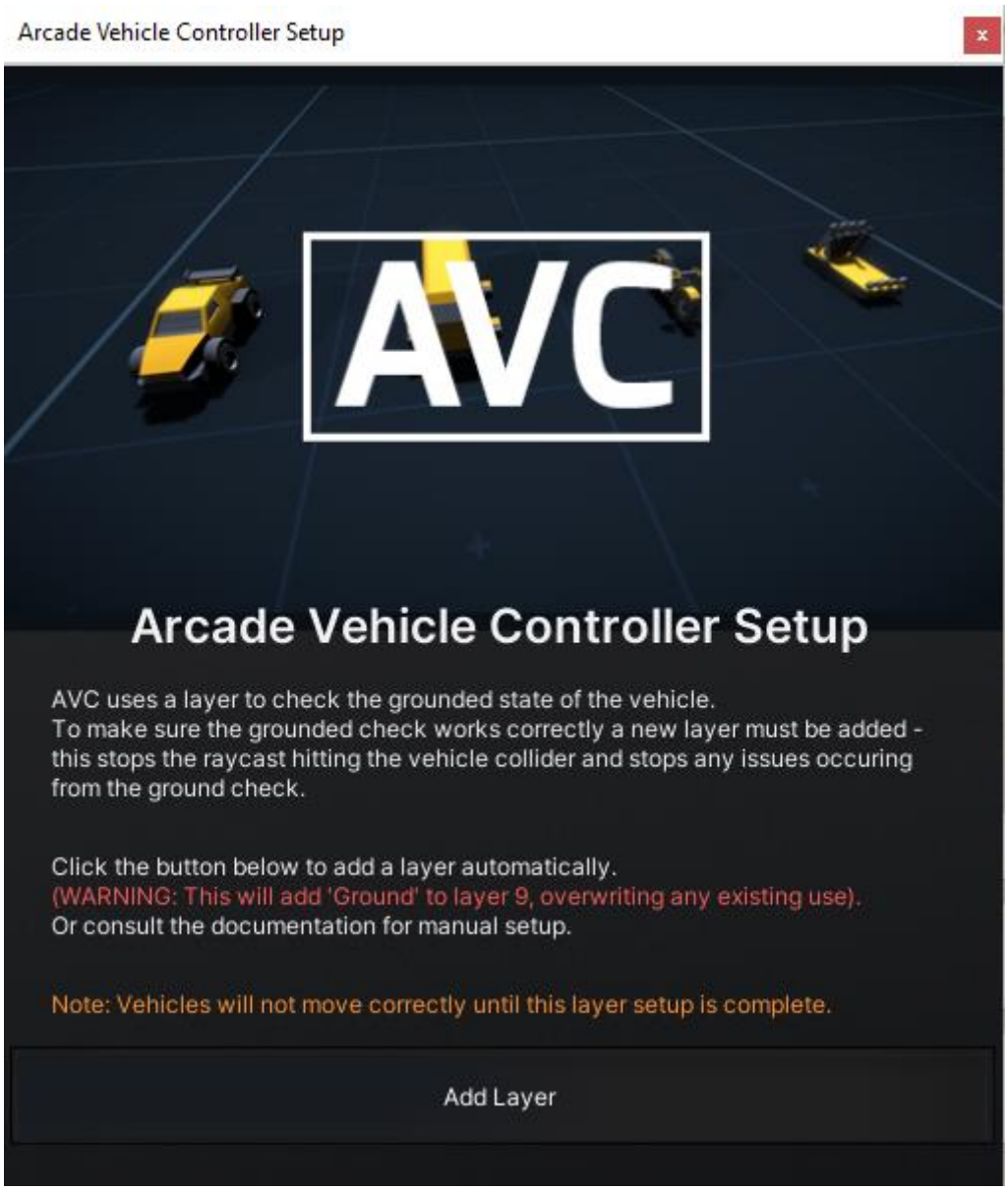
/Scripts/ - The scripts used to simulate the vehicle movement and control the effects.

/Scripts/Editor/ - Editor scripts, which give a custom inspector for VehicleBehaviour.cs, and the vehicle builder editor window scripts. In here there is also a ScriptableObject to save your builder window data.
/Scripts/Examples/ - Example scripts to show you how you might want your own classes to communicate with VehicleBehaviour.cs. These are optional, however, they are used on the example prefabs, deleting these scripts will give you missing component references on those.

Layer Setup

To work correctly, AVC requires a ground layer to be set and used – this stops the ground check raycast from hitting the vehicle collider and causing issues with the movement. You can use multiple layers (eg: ground, road, sand, water, gravel, grass, etc)

After importing AVC, you should be greeted with the layer setup window:



If that window doesn't automatically appear, select Tools -> e23 -> 'AVC Setup'. Clicking the 'Add Layer' button will automatically add 'Ground' into 'User Layer 9' (found in Edit -> Project Settings -> Tags and Layers). If layer 9 is not blank, then the setup will not continue, and you will be given an error.

Manual Setup:
1- Open the Tags and Layers window (Edit -> Project Settings -> Tags and Layers).
2- Find an empty user layer.
3- Enter 'Ground'.
4- Go to \Assets\e23\ArcadeVehicleController\Examples\VehicleSettings
5- For each settings asset set the 'Ground Mask' as the layer you created in step 3.
6- Load the demo scene.
7- Select the game object 'Tracks'.
8- Set the layer as the layer you created in step 3, and yes to change all children.

Demo Scene

After you've imported the package, take a look at the demo scene, here you can try 4 different vehicle types and see how you might control a vehicle with AI.

The demo scene is located in */e23/ArcadeVehicleController/Demo/Scene/*

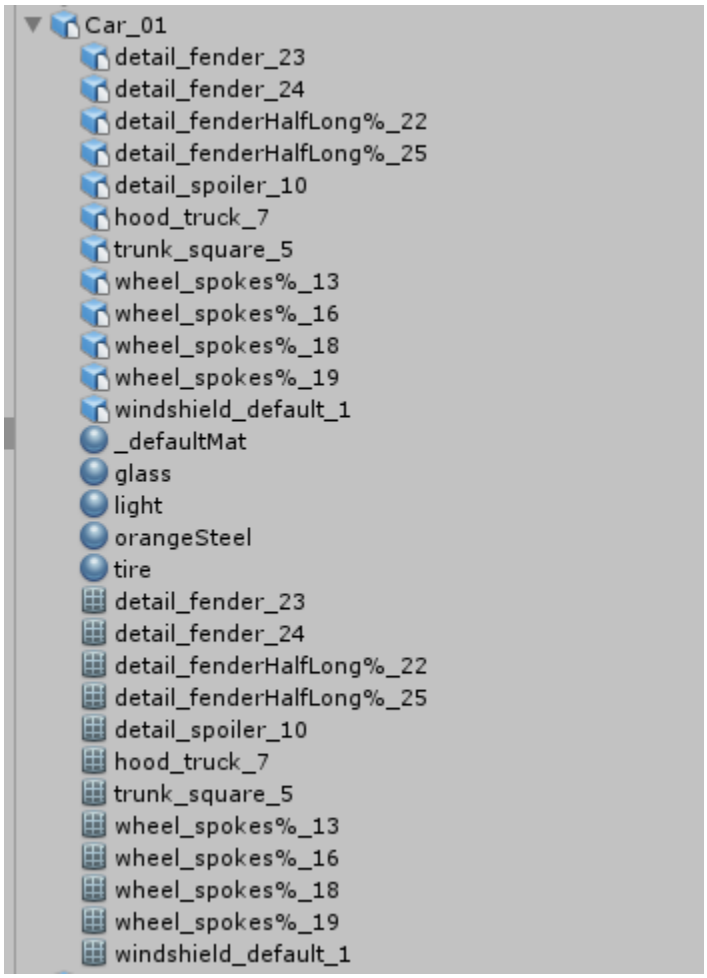
Load the scene and press play. The controls are:
- WSAD to move
- C to change the camera between top down and follow
- R to reset the vehicle position
- T to view the AI, or return to playable vehicles

Use the buttons along the bottom of the screen to swap between playable vehicles.

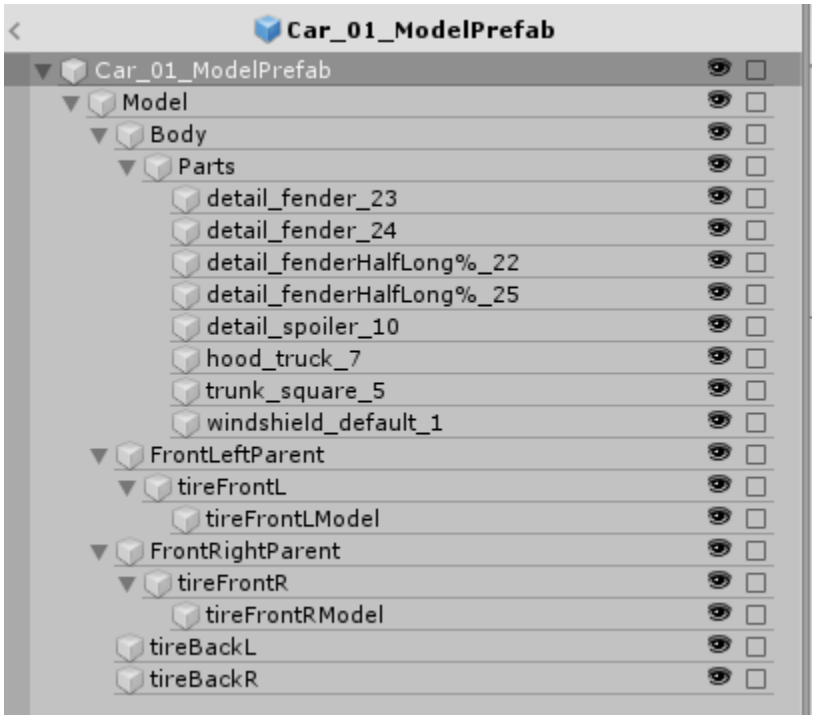
Car	Hover Car	Motorbike	Airboat	Helicopter

Preparing Your First Vehicle

1- For the vehicle to work correctly, the hierarchy needs to be setup in a certain way. Below is an image of how my vehicle model imported, this won't work.



2- Compare that with the hierarchy for the prefab.



Model holds everything, and should be at position 0,0,0.
Body has all the body parts underneath it, and should be at 0,0,0.
Parts is optional, required for this model because the body faces in the wrong direction. This GameObject is rotated 180 on Y.

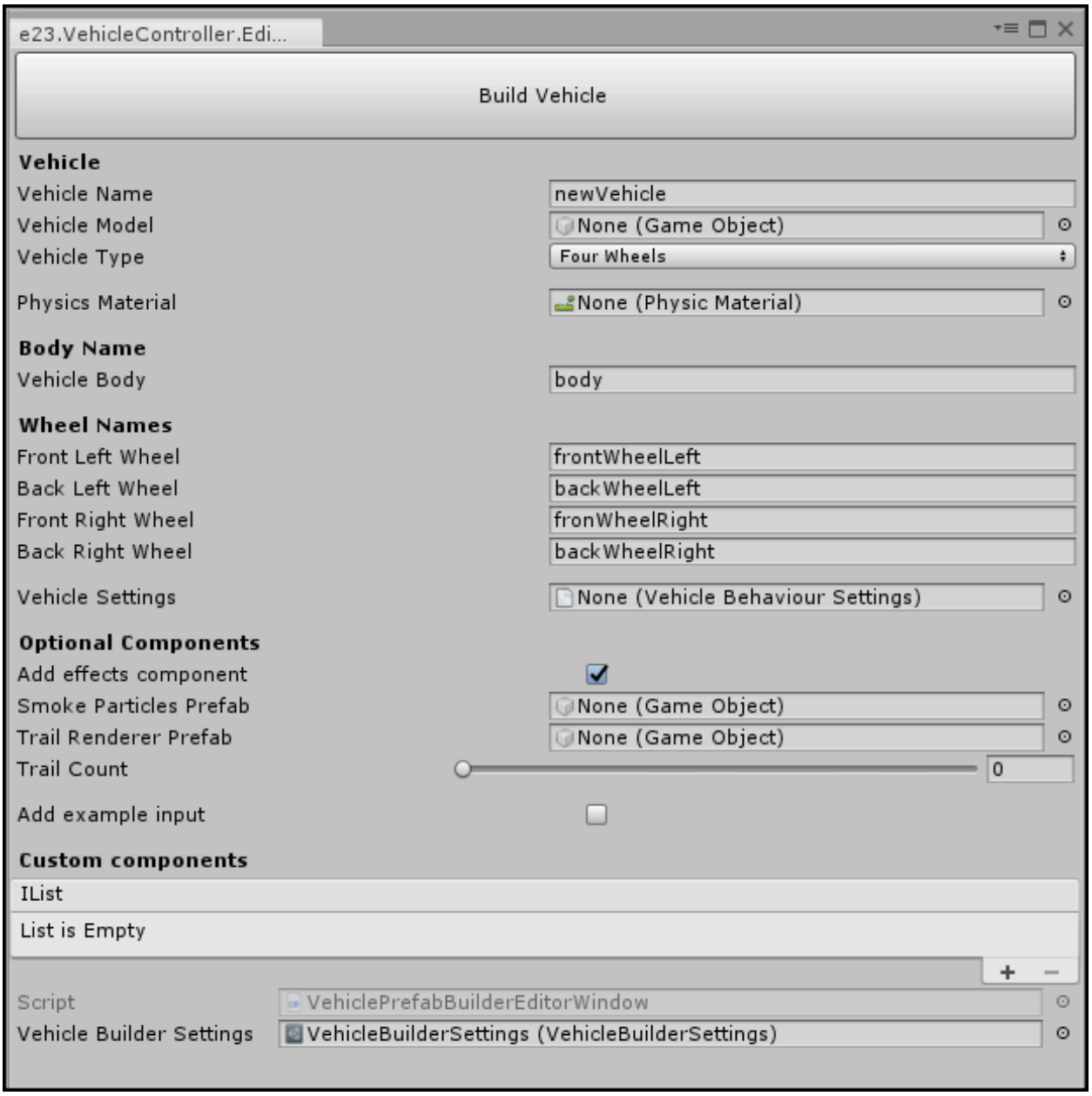
The two front tires require an empty GameObject (**FrontParent**) to deal with their turning rotation. Due to the pivot point on these wheels, they also require an additional empty GameObject (**tireFront**) to get the pivot to be at 0,0,0 for the correct rotation.

Parenting the front wheels:

3- Create an empty GameObject.
4- Copy the position of your front left wheel.
5- Select the empty GameObject and paste the position from the front left wheel.
6- Draw the front left wheel to be a child of the empty GameObject and rename to “FrontLeftParent”.
7- Repeat for the front right wheel.

Building Your First Vehicle

Now that your vehicle model is ready, you can open the vehicle builder and make your working vehicle. From the menu bar -> Tools/e23/Vehicle Prefab Builder.



Vehicle

Vehicle Name – the name you want your new vehicle to be called.
Vehicle Model – the model that you have just prepared to be used in the previous section.
Vehicle Type – how many wheels does the vehicle have.
Physics Material – which material to add to the vehicle collider – optional, but recommended to use the included physics material.

Body Name

Vehicle Body – the name of the body in the model.

Wheel Names

Wheel Names – the names of the wheels in the model. This is the name of the model GameObject for the wheel set at 0,0,0 (not the front wheel parents, they are found automatically).

Vehicle Settings – the settings you wish to use for the vehicle being created. For your first vehicle, just click the circle and select the Car settings.

Optional Components
Example Effects – if you wish to have smoke and/ or tire trails, enable this option and assign the prefabs. You can also select here how many wheels should get the trail.
Example Input – an example script that uses WSAD to move the vehicle will be added, optional – to add your own input component use the Custom Components list.

Custom Components
Any components that you have created yourself, that needed to be added to a vehicle, can be included in this list, they will be added to the vehicle too.

Clicking build now will create a vehicle in the scene with the correct hierarchy and should be playable straight away.

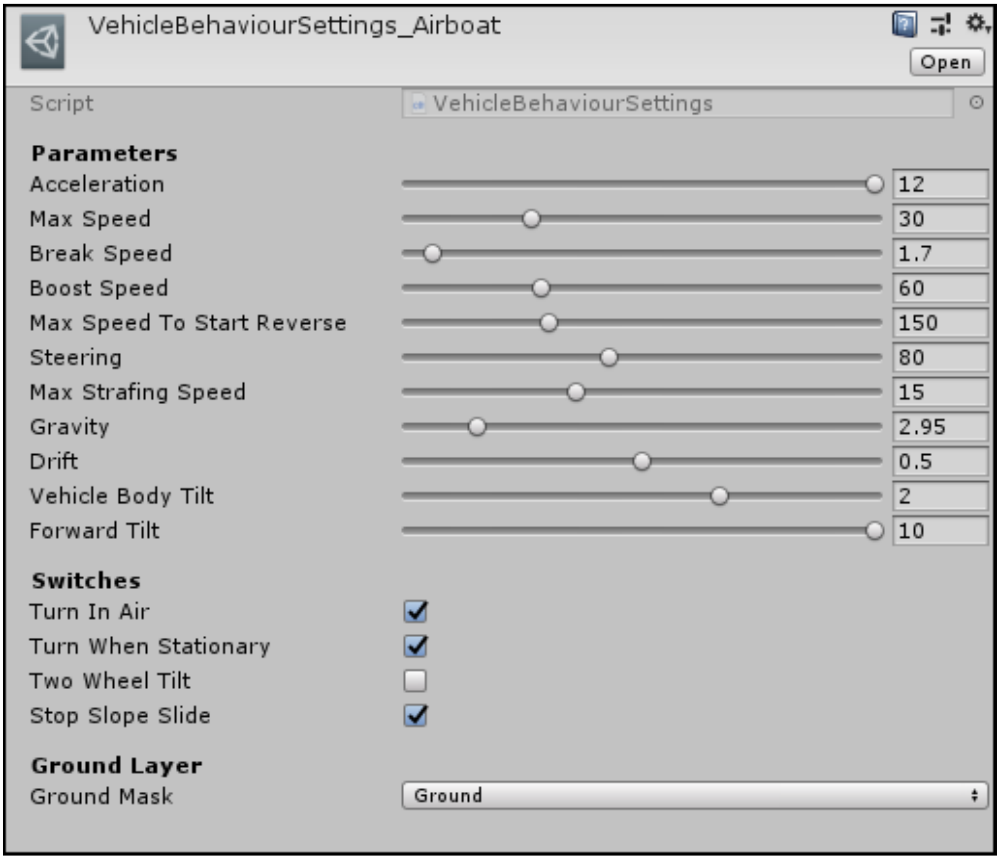
These steps are available on YouTube:
https://www.youtube.com/watch?v=9m1Am-BYoZo

Vehicle Settings

Vehicle settings store the data for a vehicle's movement. These settings can be used on multiple vehicles at the same time, making it easy to tweak values. You can have 50 vehicles and 1 settings object, or 50 vehicles each with their own settings.

To create a new settings object, navigate to the folder you wish to store it in:
right mouse click -> Create -> e23 -> Vehicle Settings

The below image is from the Airboat example settings that come included with the asset. Most of the values are self-explanatory.



Parameters

Acceleration – How fast to speed up the vehicle.
Max Speed – How fast the vehicle can drive (used for forward and reverse).
Break Speed – How fast to slow the vehicle down.
Boost Speed – How fast the vehicle will move when boosting
Max Speed To start Reversing – Breaking will slow the vehicle down (using break speed), once this value is reached the vehicle will stop breaking and start reversing.
Steering – Controls the turning angle of the vehicle, how tight the vehicle can turn. The lower the value, the larger the turning arch.
Gravity – A higher value will bring the vehicle down to the ground faster.
Drift – How easily the vehicle will slide around a corner, the lower the value the harder it is to drift.
Vehicle Body Tilt – How much the vehicle will tilt on the Z axis when turning.
Forward Tilt – The multiplier applied to work out how much the vehicle should tilt on the X axis, when moving forward. The higher the value, the smaller the tilt.
Turn in Air – Allow the vehicle to rotate when not grounded.
Two Wheel Tilt – Used for motorbikes, to give an extra tilt when turning corners.
Stop Slope Slide – Enabling this will stop the vehicle from sliding down a slope, when it is perpendicular to the slope.
Ground Layer – This is a LayerMask for you to set as your ground layers, which the vehicle will use to know if it is grounded.

Script Reference

The main class in Arcade Vehicle Controller is *VehicleBehaviour.cs*, which must be attached to any vehicle for it to move.

Properties		
Acceleration	Float	How fast your vehicle speeds up, Range value 1 – 12 (higher value means faster). Set via the vehicle settings.
MaxSpeed	Float	How fast your vehicle can move, Range value 5 – 100 (high value means faster). Set via the vehicle settings.
BreakSpeed	Float	How fast your vehicle brakes, Range value 5 – 40 (higher value means faster). Set via the vehicle settings.
BoostSpeed	Float	How fast your vehicle moves when boosting, Range value 5 – 200. Set via the vehicle settings.
MaxSpeedToStartReverse	Float	The speed your vehicle will be travelling when it stops braking and reverses instead, Range 1 – 500 (higher value means sooner). Set via the vehicle settings.
Steering	Float	Controls how tight of a turning circle your vehicle has, Range 20 – 160 (higher value means tighter cornering). Set via the vehicle settings.
Gravity	Float	A multiplier used to determine how quickly the vehicle falls to the ground, Range 0 – 20 (higher value means heavier vehicle). Set via the vehicle settings.
Drift	Float	Determines how slidey the vehicle is when turning, Range 0 – 1 (1 means very slidey). Set via the vehicle settings.

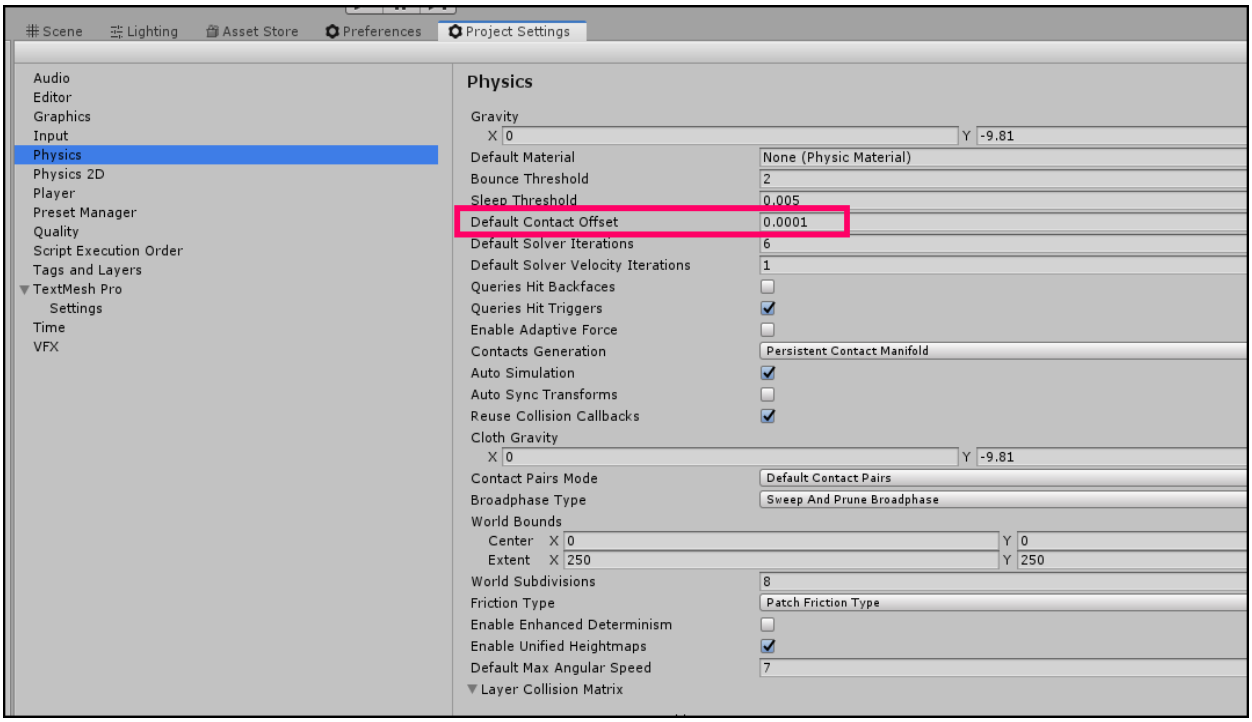
VehicleBodyTilt	Float	Sets how much the vehicle tilts when turning, Range 0 – 3 (higher value gives a bigger tilt) Set via the vehicle settings.
ForwardTilt	Float	Sets the multiplier for how much vehicle tilts when accelerating/ reversing, Range 1 – 10 (higher value means smaller tilt). Set via the vehicle settings.
TurnInAir	Bool	Allow the vehicle to turn in the air. Set via the vehicle settings.
TwoWheelTilt	Bool	Gives the vehicle an extra tilt when turning, this is for vehicles with 2 wheels. Set via the vehicle settings.
NearGround	Bool	Returns if the vehicle is close (2 Unity units) to the ground.
OnGround	Bool	Returns if the vehicle is grounded (1.1 Unity units).
GroundMask	LayerMask	Returns the layers that the vehicle can move on.
DefaultMaxSpeed	Float	Returns the MaxSpeed from the Vehicle settings, allowing you to manipulate the MaxSpeed and return to the default value.
DefaultSteering	Float	Returns Steering from the Vehicle settings.
GetVehicleVelocitySqrMagnitude	Float	Returns the velocity square magnitude of the vehicle as a float, used to work out when to emit smoke.
GetVehicleVelocity	Vector3	Returns the velocity of the vehicle as Vecto3, used to work out when to emit smoke.
VehicleWheelCount	Enum VehicleType	Changes how many wheels the vehicle has. 4 wheels. 2 wheels. No wheels.
VehicleModel	Transform	Parent transform of the vehicle model.
PhysicsSphere	Transform	Transform for the GameObject named ‘SphereRigidBody’.
VehicleBody	Transform	Transform for the ‘body’ GameObject, used for tilting.
FrontLeftWheel	Transform	Transform for the main wheel. This is used as the front left of a vehicle with 4 wheels, and the front of a 2 wheeled vehicle.
FrontRightWheel	Transform	Transform for the secondary wheel. This is used as the front right of a vehicle with 4 wheels, and the rear of a 2 wheeled vehicle.
BackLeftWheel	Transform	Back wheel.
BackRightWheel	Transform	Back wheel.
VehicleSettings	VehicleSettings	Link to the vehicle settings ScriptableObject the vehicle is using for the data.

Public Methods	
SetVehicleSettings()	Updates the vehicle settings from the VehicleSettings ScriptableObject. Called in Awake() and from the update button located on the component in the inspector.
ControlAcceleration()	Makes the vehicle move forward, use this to call forward movement from your input class.

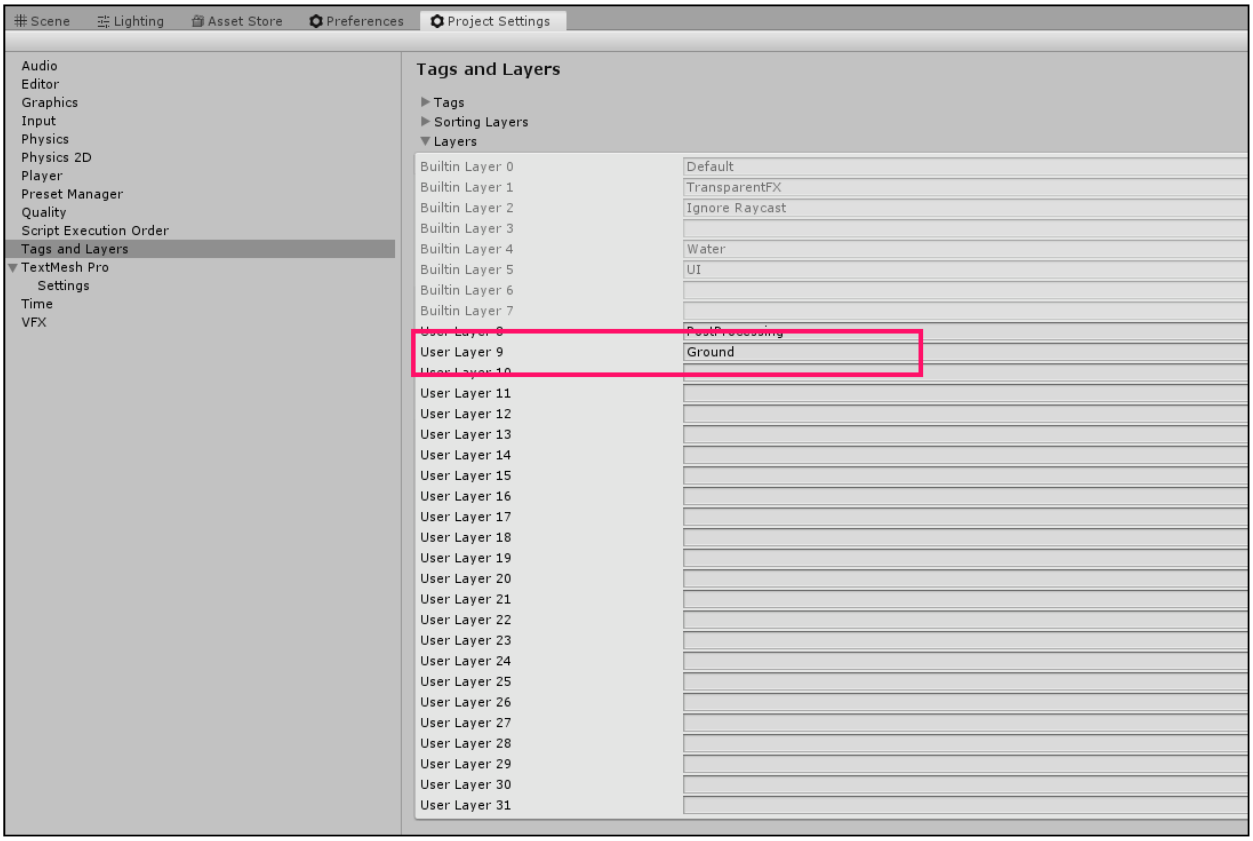
ControlBrake()	Makes the vehicle brake, or reverse, use this to call backwards movement from your input class.
ControlTurning(int)	Controls the turning of the vehicle, and takes in an int to determine the direction. -1 for left 1 for right
SetPosition(Vector3, Quaternion)	Used to reposition and rotate the vehicle, best used for respawning. Pass in a Vector3 for the new position and a Quaternion for the rotation.
MovementPenalty(float)	Used to change the MaxSpeed of the vehicle. Example use: slowing down on grass.
SteeringPenalty(float)	Used to change the Steering speed of the vehicle. Example use: making it harder to turn on grass.
Boost()	Used to set the vehicle boosting, changes *isBoosting* to true.
OneShotBoost(float)	Performs a boost over a set amount of time (in seconds).
StopBoost()	Changes *isBoosting* to false returning the vehicle to MaxSpeed.

Troubleshooting

1- Vehicle bounces on the join of two ground colliders. Change the ‘Default Contact Offset’ in 'Project Setings' -> 'Physics'. It runs great for me with the value set to 0.0001.



2- Vehicle moving very slowly or not at all. As of v1.5 AVC requires a LayerMask to be set for the ground, by default this should be set to “Default”. On your vehicle settings check what the ‘Ground Mask’ is set as and correct this to what your ground layer is set as.



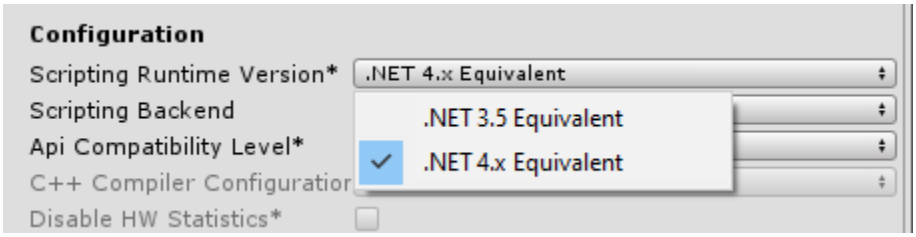
3- NullReferenceException in the console. There can sometimes be a null reference appear in the console, linked to the Vehicle Builder Window, this appears to be a Unity bug and can be ignored. (https://forum.unity.com/threads/assetdatabase-saveassets-is-giving-a-nullreferenceexception.778691/). Closing the Vehicle Builder Window when you're done with it should stop the error appearing.

```
NullReferenceException: Object reference not set to an instance of an object
UnityEditor.GameObjectInspector.ClearPreviewCache () (at
/Users/builduser/buildslave/unity/build/Editor/Mono/Inspector/GameObjectInspector.cs:211)
UnityEditor.GameObjectInspector.OnDisable () (at
/Users/builduser/buildslave/unity/build/Editor/Mono/Inspector/GameObjectInspector.cs:199)
UnityEditor.AssetDatabase:SaveAssets()
e23.VehicleController.Editor.VehiclePrefabBuilderEditorWindow:SavePrefabSetup() (at
Assets/e23/SimpleCarController/Scripts/Editor/VehiclePrefabBuilderEditorWindow.cs:475)
e23.VehicleController.Editor.VehiclePrefabBuilderEditorWindow:OnDisable() (at
Assets/e23/SimpleCarController/Scripts/Editor/VehiclePrefabBuilderEditorWindow.cs:63)
```

4- Vehicle wheels not rotating correctly. This can happen when the pivot of the wheel model isn't set in the way the AVC is expecting it. Create an empty GameObject, position it in the same place as the wheel, and then place the model as a child of the GameObject – renaming the new parent as your search term.
If you are correcting an already built vehicle, make sure to assign the new parent object to the correct wheel transform on the VehicleBehaviour component.

5- If the below error is showing in the console, you need to change the projects 'Scripting Runtime Version' to '.NET 4.x Equivalent'. This can be found in Player Settings -> Other Settings -> Configuration.

```
Feature `nameof operator' cannot be used because it is not part of the C# 4.0 language specification
```



Social stuff:
http://enigma23games.com
https://twitter.com/en9g131
https://www.facebook.com/en9g131/
https://forum.unity.com/threads/avc-arcade-vehicle-controller.781895/