MS Vehicle System

Created by: Marcos Ismael Schultz
Site: www.schultzgames.com
Forum: https://forum.unity.com/threads/ms-vehicle-system-vehicle-controller.544282/
Email: marcos11-24@hotmail.com

'MS Vehicle System' is a complete controller for vehicles that use the wheel collider component. The system includes the following features listed below:

- Scene controller, allowing you to get out and get into vehicles in a simple way.
- Support for vehicles to be controlled by other codes (AI).
- Automatic and manual gears system for the vehicle.
- Joystick, steering wheel and control buttons for mobile devices.
- Controller for cameras, including 9 different modes of handling.
- Complete torque distribution system for the vehicle, allowing precise adjustment.
- Fully adjustable brake system.
- Realistic and adjustable suspension system.
- Simulators of trawl and air friction, gravity and rotation.
- Completely complete lighting system including main lights, flashing lights, warning lights, brake lights, reverse lights, among others.
- Steering wheel system for the vehicle, allowing separate control of each wheel.
- Fuel System.
- Complete mirror system.
- Damage system to the vehicle, causing it to be kneaded.
- Complete system of sounds for the vehicle, including wind sounds, brakes, engine, lights, horn, reverse gear, beats, bumps, terrain, among others.
- System that allows to adjust the friction and the torque of the vehicle in each terrain separately.
- Complete system of skid marks, allowing you to adjust the trail width, opacity, and color depending on each terrain.
- Complete particle system, including wheel dust, exhaust fumes, smoke damage, among others.
- And much more.

'MS Vehicle System' also includes several sample scenes and includes some demonstration vehicles, which allows you to test all systems.

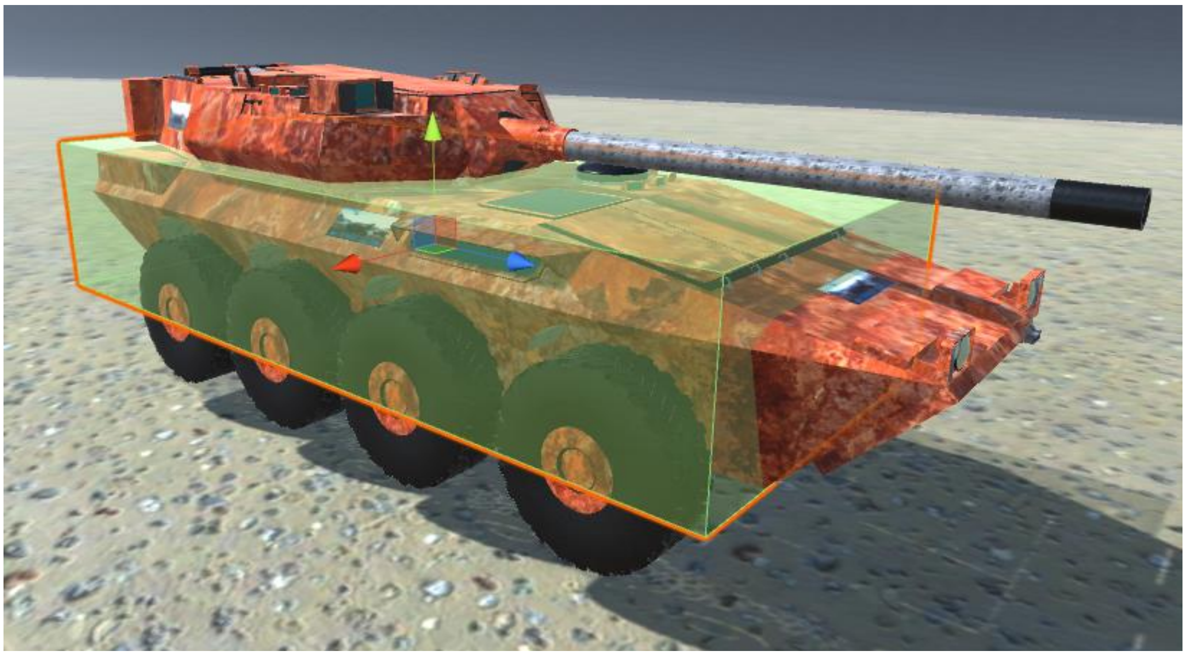
The asset also comes with several pre-fabricated objects (prefabs), making it easy to use the system with objects already configured.

How to use:

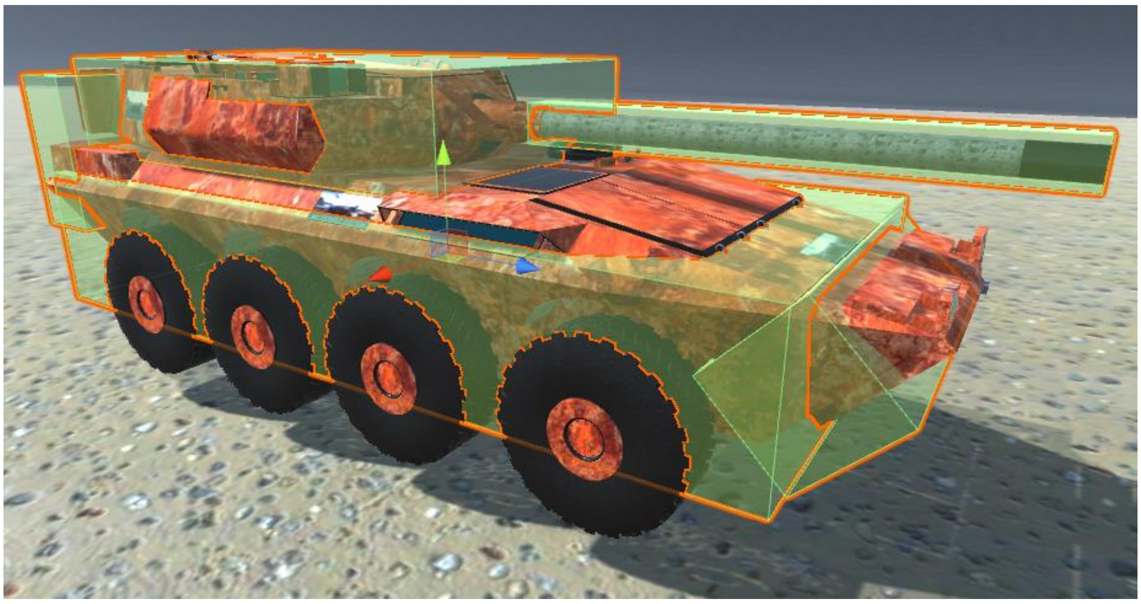
To use the system, you must first have a 3D model of a properly rotated and staggered vehicle, that is to say that the vehicle's rotation must be pointed towards the Z axis of the vehicle's main transform. Likewise, the wheels of the vehicle must also be rotated so that the Z axis of its transform is in the same direction as the Z axis of the main transform of the vehicle.

It is also good that the vehicle has scale 1 in all its meshes and transform. This implies in the main transform of the vehicle to have the scale (1,1,1), as well as all other transform of the vehicle, to the wheels ... This makes the simulation of physics more realistic.

Having the vehicle properly adjusted in rotation and scale, it is necessary to add a collider to the vehicle, to simulate physics and collisions. It is recommended that the collider cover the wheels of the vehicle to avoid surreal impacts. The image below illustrates the ideal position of the collider.

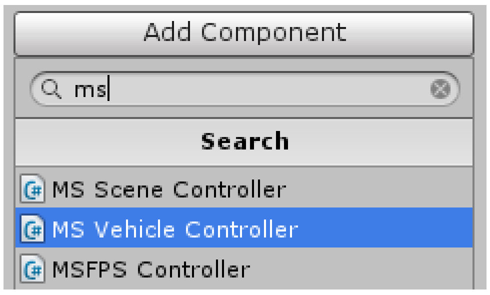


For greater realism, several collisions attached to the vehicle can be used, adjusted to mimic the shape of the vehicle. The following image demonstrates this process.

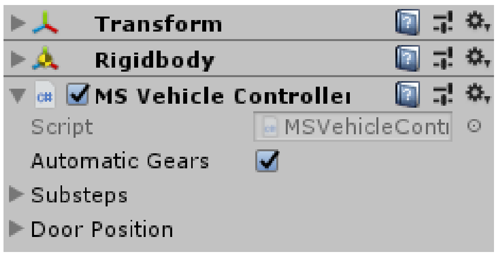


The "mesh collider" can also be used, but it is not recommended because it causes bottlenecks when the object that has the attached collider is enabled or disabled in the hierarchy.

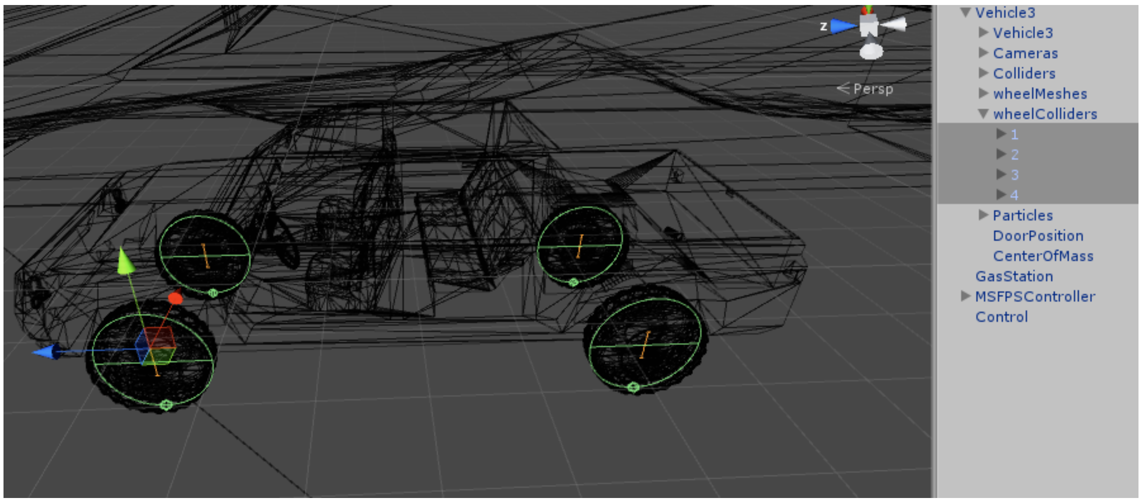
After correctly setting the vehicle's collisions, the 'MS Vehicle Controller' component must be added to the vehicle main transform.



By doing this, the script will automatically add the 'Rigidbody' component to the main vehicle's transform, to simulate the physics of the vehicle.



After doing all this, it's time to add the 'WheelCollider' component to each wheel of the vehicle by simply creating an empty object in the position of the center of the wheel, and adding to this object the 'WheelCollider'.



Now click on the main transform of the vehicle, go to the 'MS Vehicle Controller' component and configure the vehicle as you want, associating the wheels in their places, configuring the torques, among other things.

Having your vehicle set up, you also need to have a Controller for the scene, which is responsible for informing the vehicle of the player's preferences and inputs.

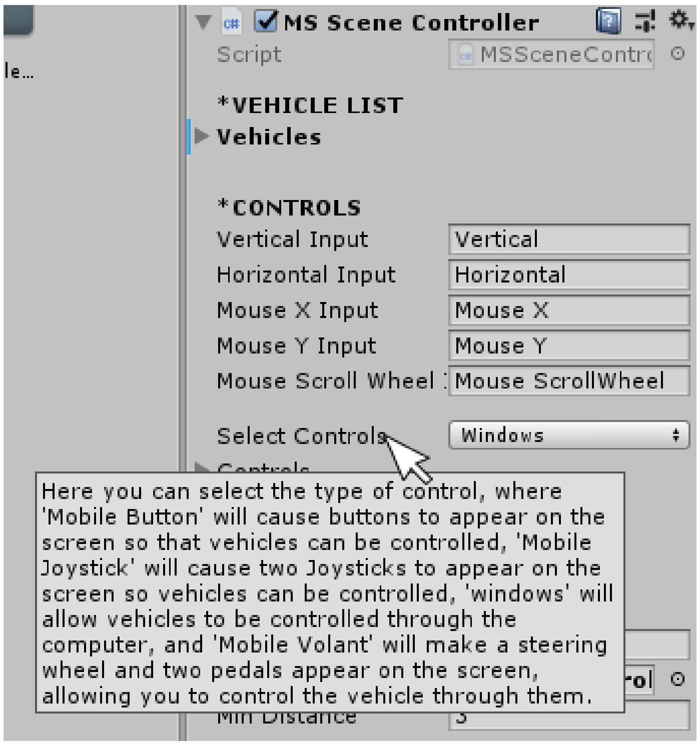
A scene controller named **'Main Control**' should be created automatically when the script '**MSVehicleController**' is added to a vehicle.

If the controller is not present in the scene, you can manually insert it into the scene. The controller is at: **Marcos Schultz > MSVehicleSystem > Prefabs > Controller > MainControl**.

Caution! Only one controller is allowed per scene, so if the controller is already present in the scene, there is no need to add another one.

After that, click the **'Main Control**' that was added to the scene and configure it in the inspector, according to each variable.

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.


MS Scene Controller

Script MSSceneContr

*VEHICLE LIST

Vehicles

*CONTROLS

Vertical Input	Vertical
Horizontal Input	Horizontal
Mouse X Input	Mouse X
Mouse Y Input	Mouse Y
Mouse Scroll Wheel	Mouse ScrollWheel
Select Controls	Windows

Here you can select the type of control, where 'Mobile Button' will cause buttons to appear on the screen so that vehicles can be controlled, 'Mobile Joystick' will cause two Joysticks to appear on the screen so vehicles can be controlled, 'windows' will allow vehicles to be controlled through the computer, and 'Mobile Volant' will make a steering wheel and two pedals appear on the screen, allowing you to control the vehicle through them.

Min Distance 3

Setting up the vehicle correctly as the 'Tooltips' indicate, and configuring the 'Control' object as their Tooltips indicate, the system should already work properly. If something is wrong, the system will provide alert warnings indicating what is wrong.

For the vehicle to have sounds, particles, lights, among other things, just follow the tips that the script provides in each variable, through the 'Tooltips'.

If necessary, there is also a video showing how to configure a complete vehicle using version 3.4.0 of this driver:

https://www.youtube.com/watch?v=piYSD1N5zgE&

The scene controller also has an enum called "SelectControls", which allows you to choose between four types of control, being them, buttons (Mobile Button), joystick (Mobile Joystick), mobile steering wheel (Mobile Volant) and keyboard keys (windows), so you can implement games for computer as well as for mobile devices.

To help understand how the entire system works, the feature contains several demo scenes, with examples of how the system works.

The asset also features a day and night system, allowing you to better see how the vehicle's lights work.

In the 'Scenario' object of each scene, there is a component called 'ExampleScene', and there is a variable called 'UI Visualizer'. When this variable is checked, instructions on how to control the vehicle will appear on the screen.

The asset carries other prefabs besides the vehicles, the scenario and the main controller. In addition, it brings an object called 'GasStation', which is a trigger, every time a vehicle passes through it, the vehicle gains fuel.

Another prefab that the asset brings is an FPS controller, to represent a player.

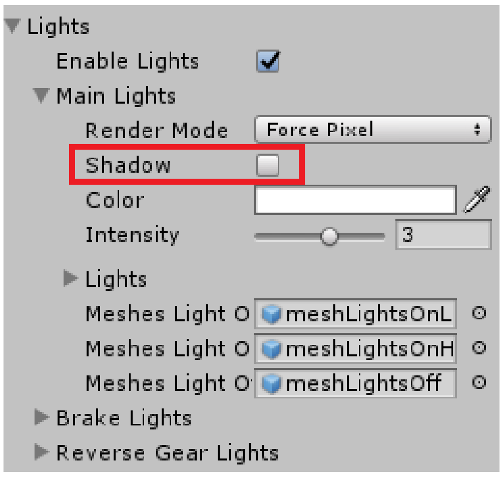
Note:

If you need to instantiate vehicles only after the scene has started, you should at least insert the 'MainControl' prefab in the scene so that vehicles can be instantiated at play time without any problems.

Possible problems:

It is possible that the scene gets heavy when using the asset, and this happens due to the simulation of vehicle lights. This is totally normal because lights tend to weigh heavily, especially if they are calculating shadows in real time.

To alleviate this problem, the vehicle controller allows you to disable the shading and change the rendering mode, leaving the lights lighter.



▼ Lights	
Enable Lights	✓
▼ Main Lights	
Render Mode	Force Pixel
Shadow	☐
Color	
Intensity	3
▶ Lights	
Meshes Light O	meshLightsOnL
Meshes Light O	meshLightsOnH
Meshes Light O	meshLightsOff
▶ Brake Lights	
▶ Reverse Gear Lights	

Another feature that the 'MS Vehicle System' code has to make lights lighter, is to simulate the illumination of the meshes separately ... In this way, you can have meshes of lights in a vehicle without the need for lights.

In addition, some heavy system resources can be disabled by simply deactivating a controller variable within the class. As for example, the classes 'Damage', 'Sounds' and 'SkidMarks'.

The wheelCollider component does not present a very realistic simulation when using actual values for the wheels, so it is advisable to leave each wheel with the mass in (1/8) the mass value of the vehicle. For example, if the vehicle has mass of 1000, each wheel must have a mass of 125, to give more realistic results.