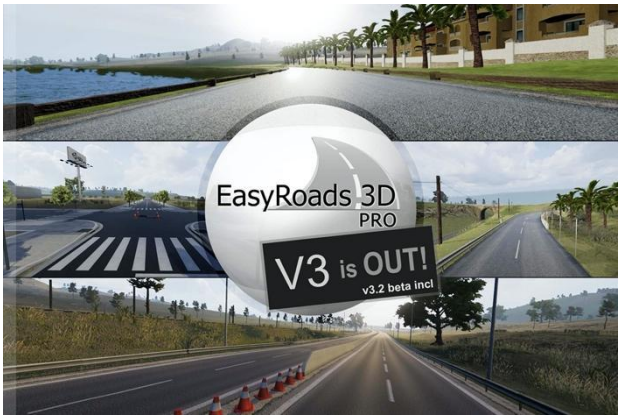
Universal Vehicle Controller

Documentation

Perfect Games 2022

Content:

Additional links.



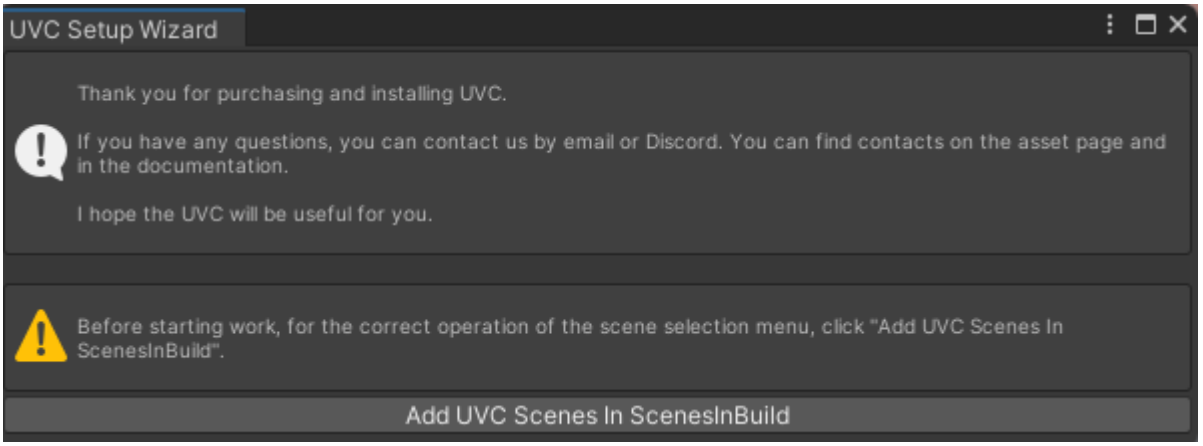
Easy Roads Pro v3 – create roads, paths and road props. Easy to use tool. This is an essential tool for creating a Racing game.

First Racer - A game based on UVC.

UVC users can request a free key for First Racer, just contact us in any convenient way in the message, specify InvoiceNO UVC to confirm the purchase.

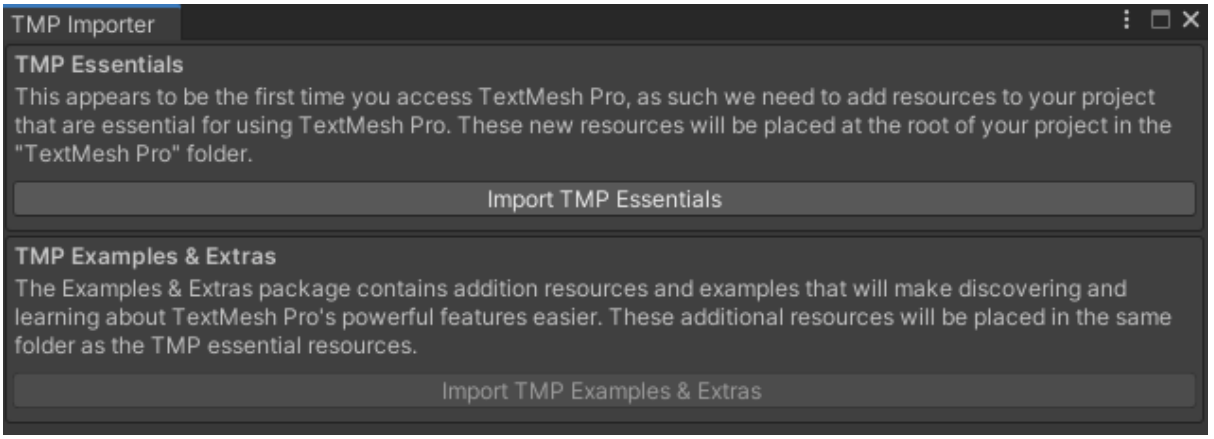
!!! Instructions for importing an asset !!!

After importing the asset, the "UVC Setup Wizard" window will open, the window opens automatically only after the asset is imported, the window can also be opened using the "Window/Perfect Games/UVC Setup Wizard" command:



If you plan to use UVC's Scenes menu, then click the "Add UVC Scenes In ScenesInBuild" button. Otherwise you can ignore this step

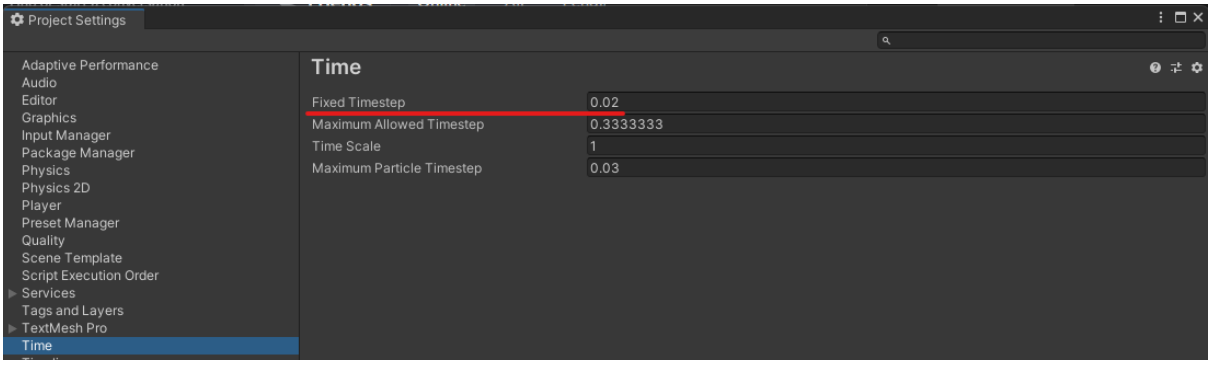
If you do not have TextMeshPro imported, then after opening any UVC scene, a message from TextMeshPro will appear:



Here you need to click the "Import TMP Essentials" button, after that you can close this window.

!!! Recommendations for Project Settings !!!

A very important parameter "ProjectSettings->Time->Fixed Timestep", greatly affects the physics of vehicles and the accuracy of damage, the lower this parameter, the better the physics will work. **I recommend using a value of 0.01.**



(ProjectSettings is opened via the Edit->ProjectSettings menu)

UVC has the ability to connect additional plugins: InputSystem, FMOD, URP.

For mobile versions, it's best not to install InputSystem. Touchscreens with InputSystem often have problems with buttons sticking or not working.

You can install other dependencies, but URP degrades performance on weak devices.

For PCs, Game Consoles and other high-performance devices, you can install dependencies at your discretion. I advise you to install InputSystem, with it setting up control from devices is much more convenient than in the standard input system.

Introduction

First of all, I want to thank you for purchasing this asset!

I tried to write the code in the most simple and understandable manner, not to overload it with unnecessary details and comments distracting readability. I also tried to make as few connections between classes as possible, I think even absolute Unity beginners will understand this code. But if you have any issues you can contact me in any suitable way. All the contacts and listed at the end of this document. Usually I answer in approx. 24 hours (sometimes a bit longer).

This asset is intended to help you create physics for a variety of different vehicles with a bit arcade-style control. Asset uses WheelColliders, I think they do their job perfectly.

Code style

Throughout the entire project I followed one style of coding:

All classes are inside the "PG" namespace to avoid class names overlapping.

Any declaration (public or private), class names, method names begin with Capital letter:

```
public Wheel[] Wheels;
float MaxMotorTorque;
float FixedUpdateBrakeLogic ()
public class EngineConfig
```

Except variables used to check these variables inside properties. These variables begin with "_" or "m_" and they are private:

```
MeshFilter _MeshFilter;
public MeshFilter MeshFilter
{
    get
    {
        if (!_MeshFilter)
        {
            _MeshFilter = GetComponent<MeshFilter> ();
        }
        return _MeshFilter;
    }
}
```

All curly braces begin on a separate line (like in the previous picture). This increases readability. Exceptions are some properties which are perfectly readable in one line and cases where division in separate lines looks inappropriate:

```
public int CurrentGearIndex { get { return CurrentGear + 1; } }
```

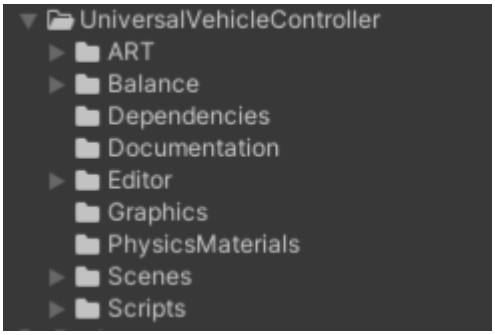
```
WheelSounds.Add(WheelsEffectEmitterRef.Event, new WheelSoundData() { Emitter = WheelsEffectEmitterRef
    });
```

With small letters begin variables which are parameters or declared inside a method:

```
public void SetDamageForce (float force)
bool forwardIsSlip = false;
bool anyWheelIsGrounded = false;
```

All possible unobvious cases have been commented. I tried to name methods and variables in the most understandable and logical manner. Exceptions may be some formulae the explanation of which will be as big, as the formula itself.

Project structure



The project is divided into directories:

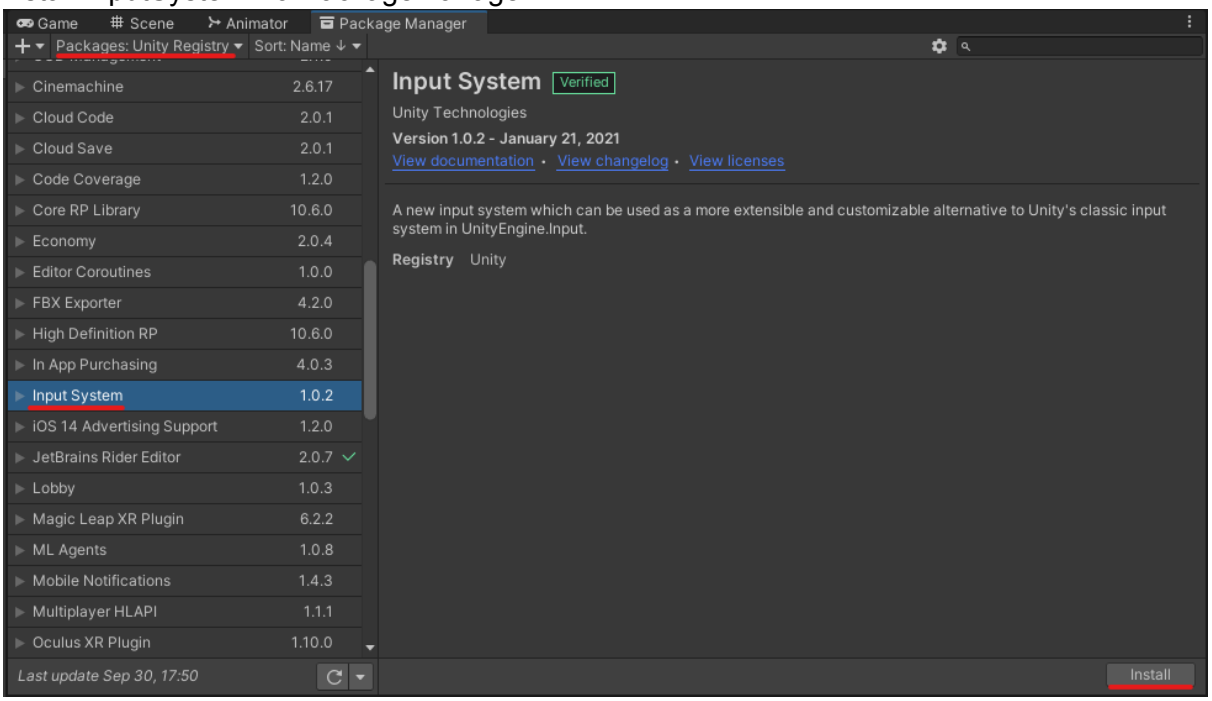
- ART – Textures, Sprites, .fbx files, materials etc.
- Balance – Files related to balance (they are not much now, just asset settings)
- Dependencies: packages with FMOD, InputSystem, URP.
- Editor: Files for editing the project (For example, the logic of creating cars).
- Graphics: Graphics prefabs, lighting, post effects settings, etc. all files are modified when the URP is imported.
- PhysicsMaterials – Physics materials
- Prefabs – UI prefabs and some GameObject used in different scenes
- RenderSettings - Settings related to rendering
- Scenes - Scenes
- Scripts - Scripts

Add-ons

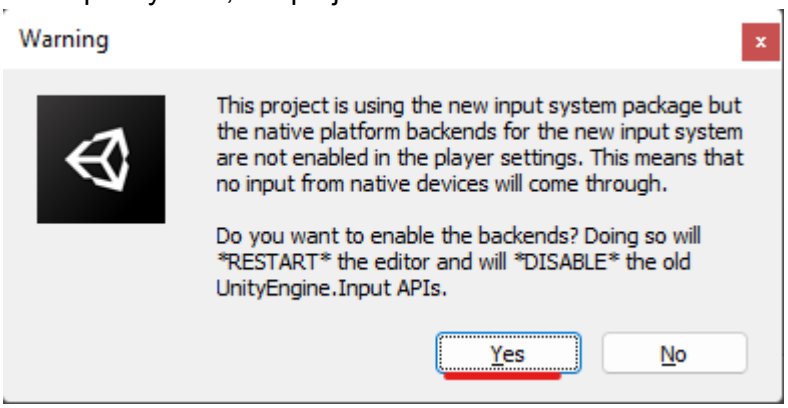
InputSystem

Installing InputSystem:

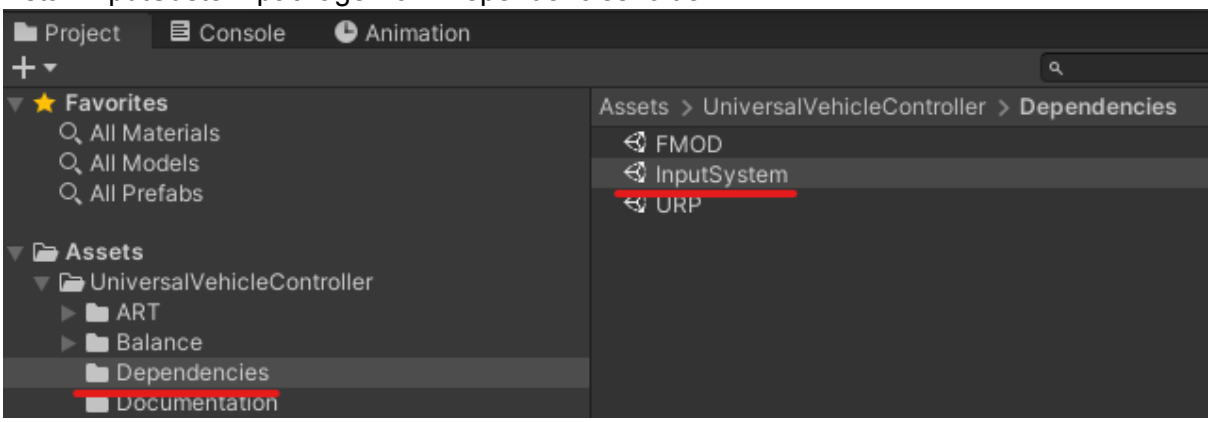
- Install InputSystem via PackageManager.

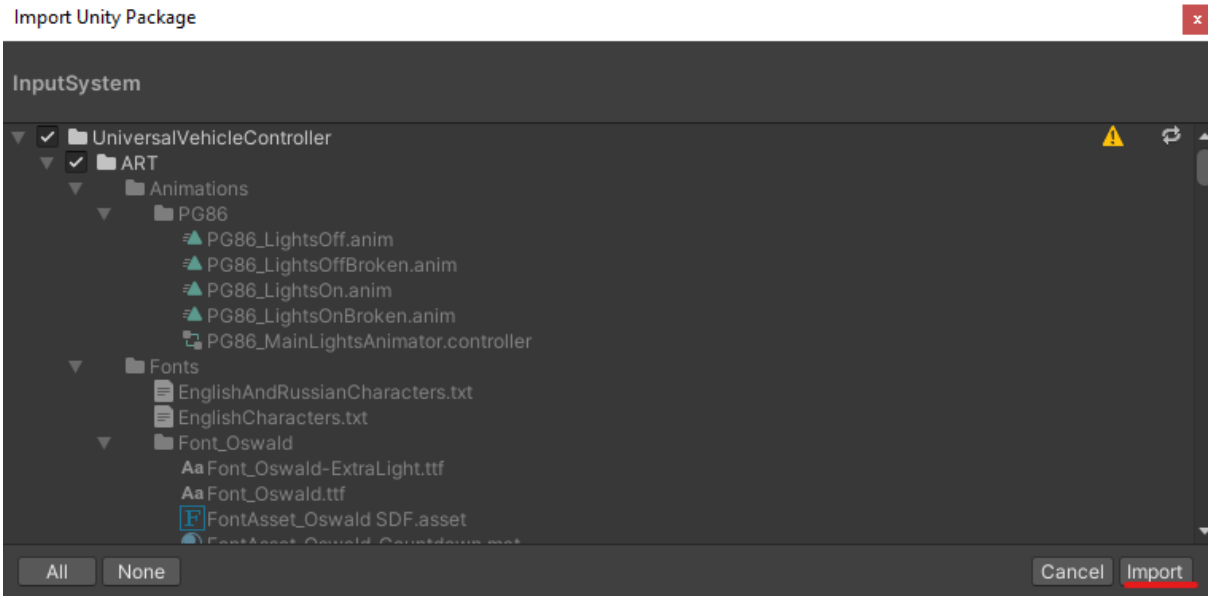


- After installation, the message "Warning" will appear, click "Yes" this will switch the project to the new input system, the project will be reloaded.



- Install InputSustem.package from Dependencies folder





When importing this file, the project files will be changed and new ones will be added:

InputHelper.cs

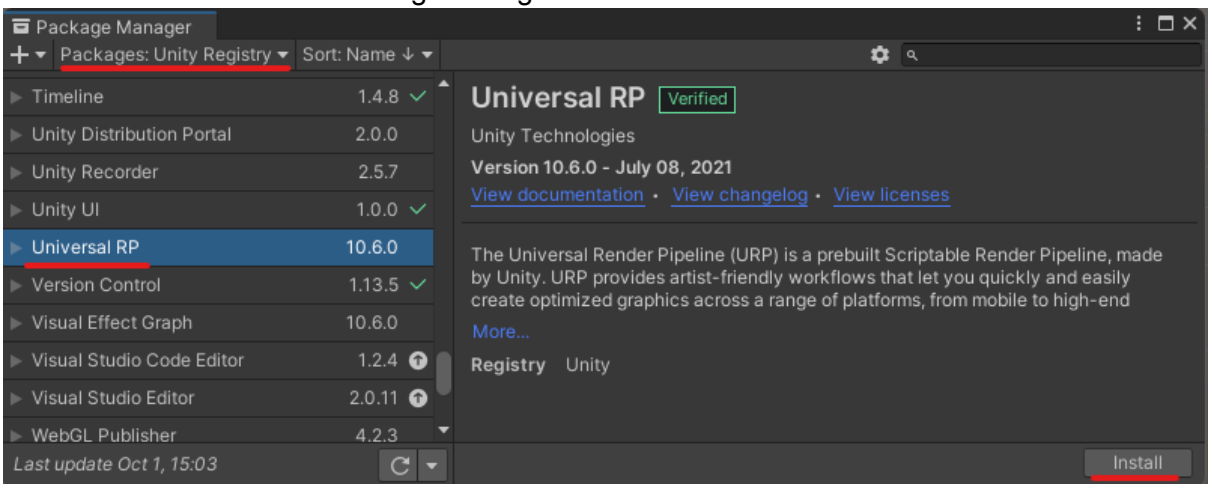
UserInput.cs

The settings are in P1Input.asset (For 1st player) and P2Input.asset (For 2nd player)

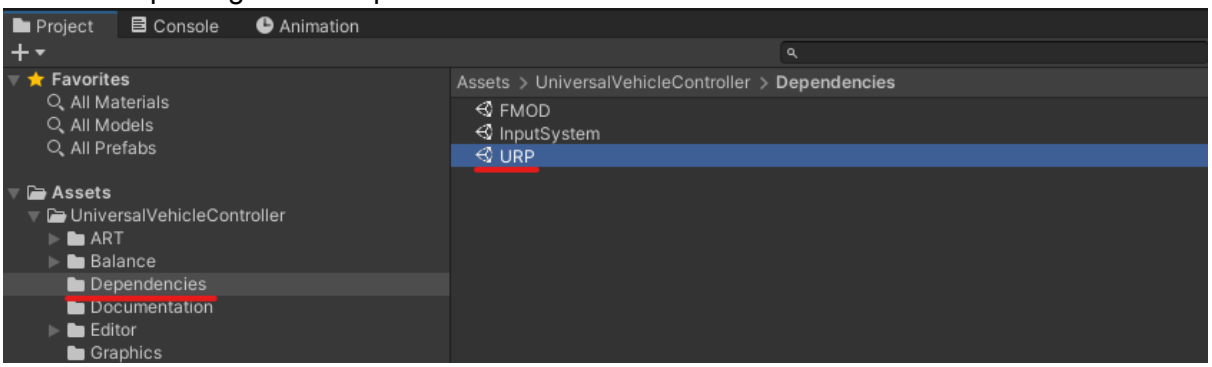
URP

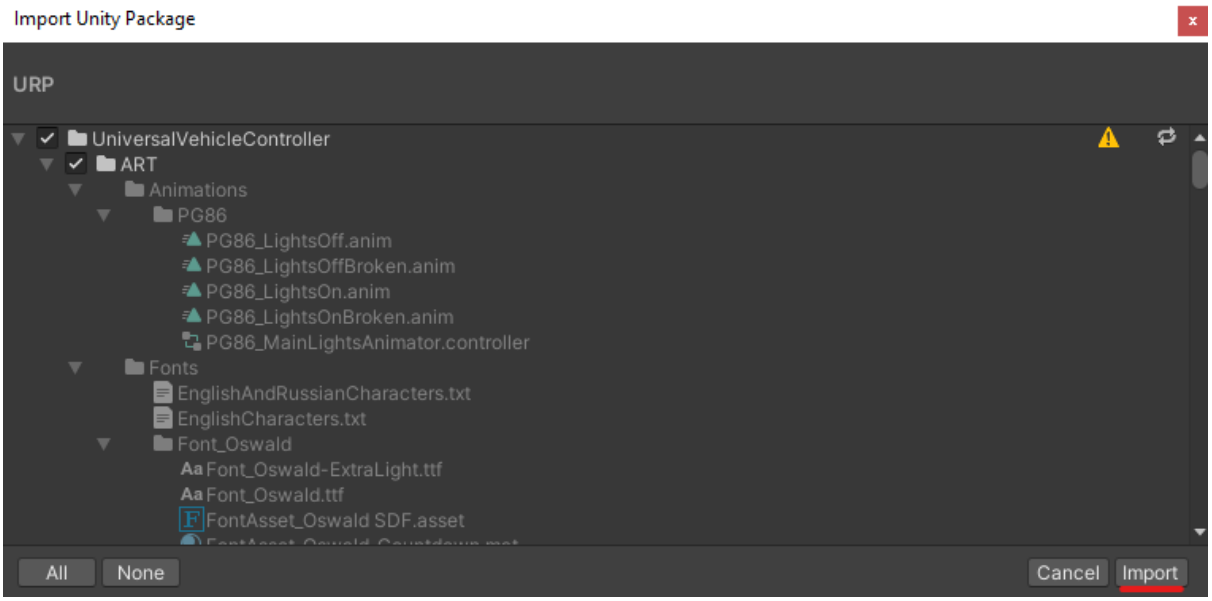
Installing URP:

- Install "Universal RP" via PackageManager.



- Install URP.package from Dependencies folder



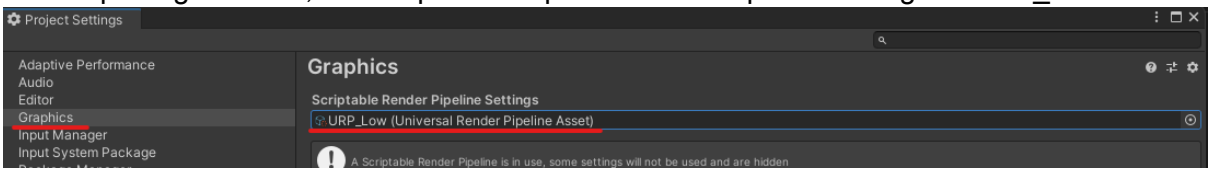


When importing this file, the project files will be changed:

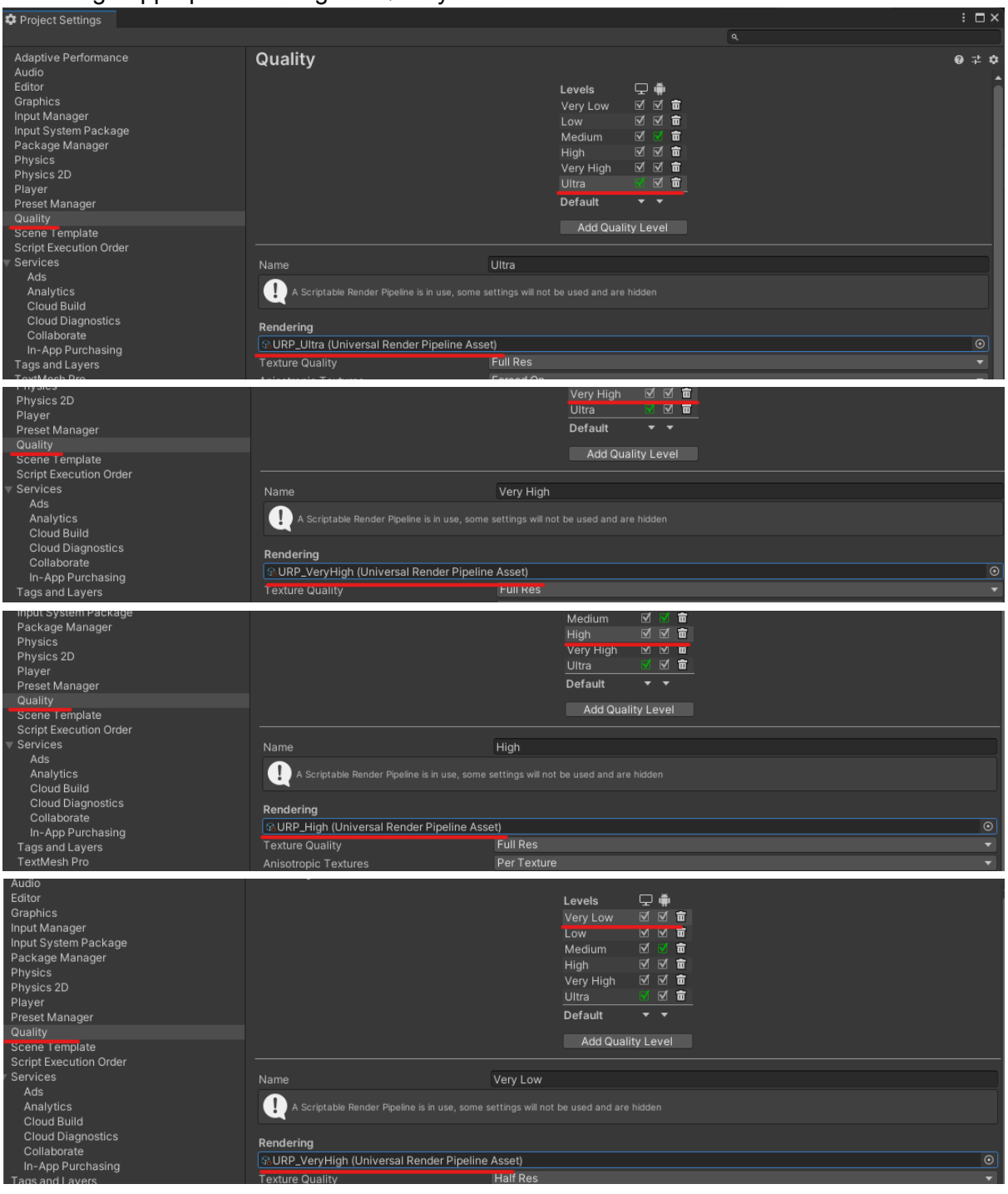
All materials

Prefabs from the Graphics folder.

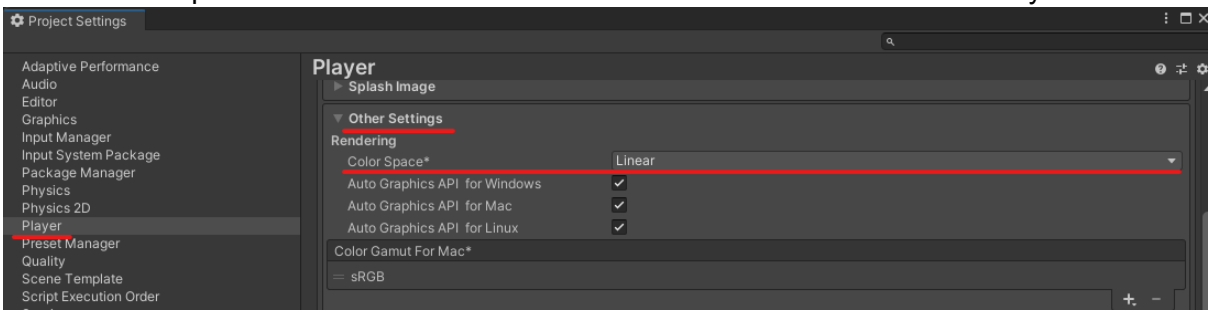
- After importing the URP, set Graphics.ScriptableRenderPipelineSettings to URP_Low.



- And assign appropriate settings in Quality



- Switch Color Space to Linear. You can use Gamma and customize the URP as you like.

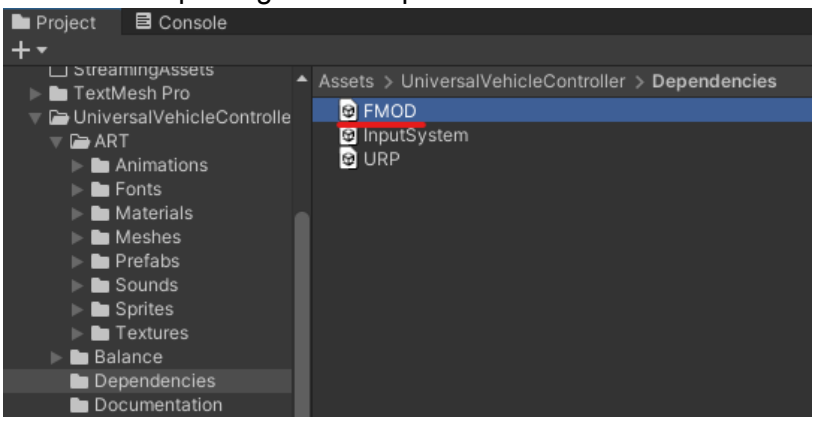


FMOD Sound

The asset can be switched to FMOD, with this plugin the sound will be better, with the knowledge of FMOD Studio, you can simply and easily customize any sounds in the game.

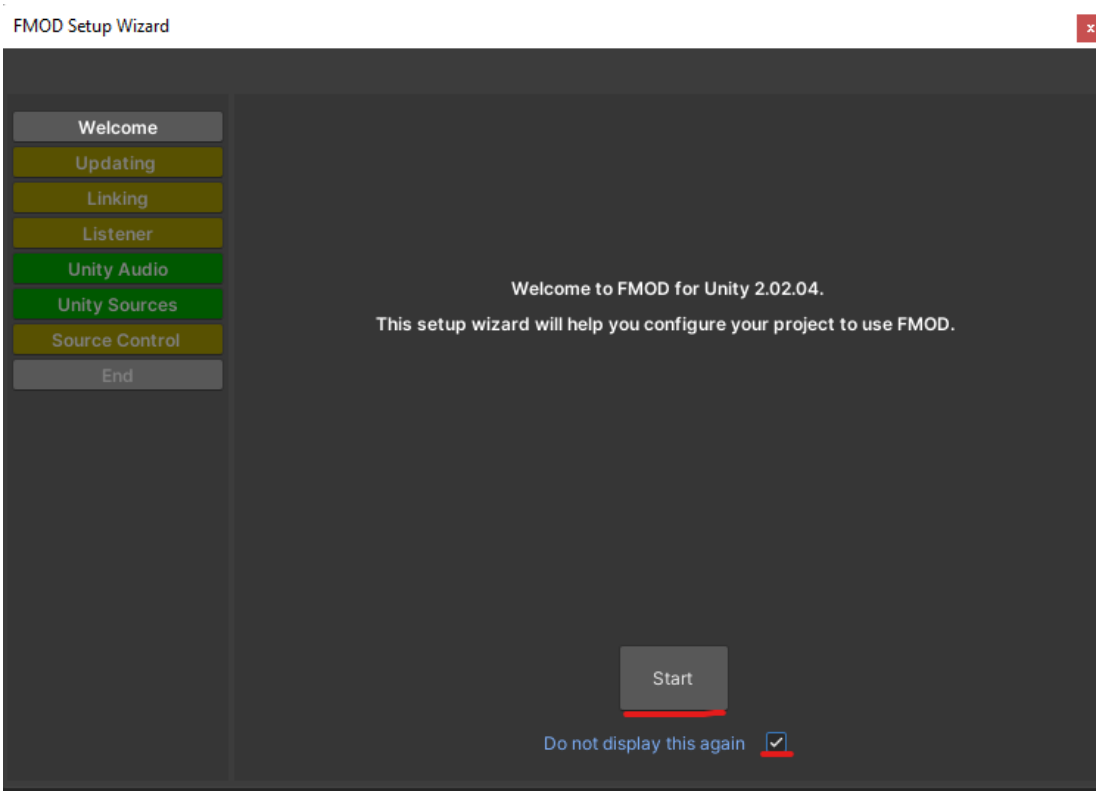
Installing FMOD:

1. Install FMOD.package from Dependencies folder

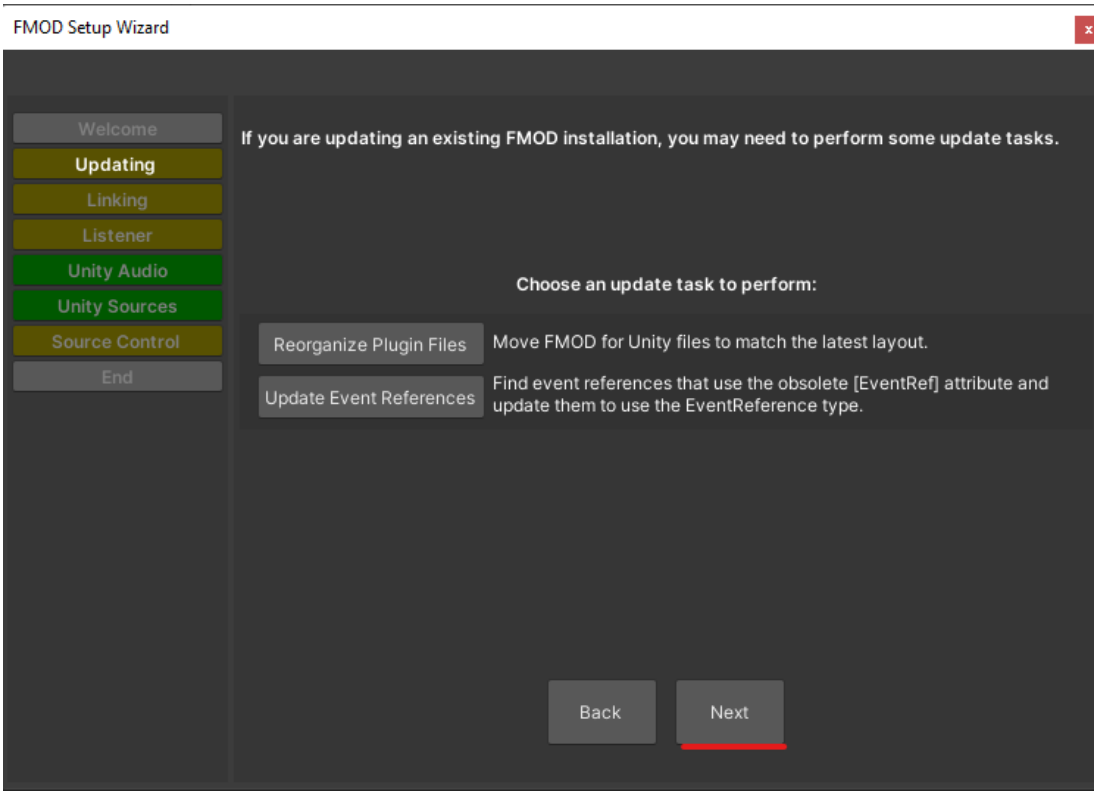


2. Download and install the FMOD asset.
3. After installing FMOD, the "FMOD Setup Wizard" window will open.

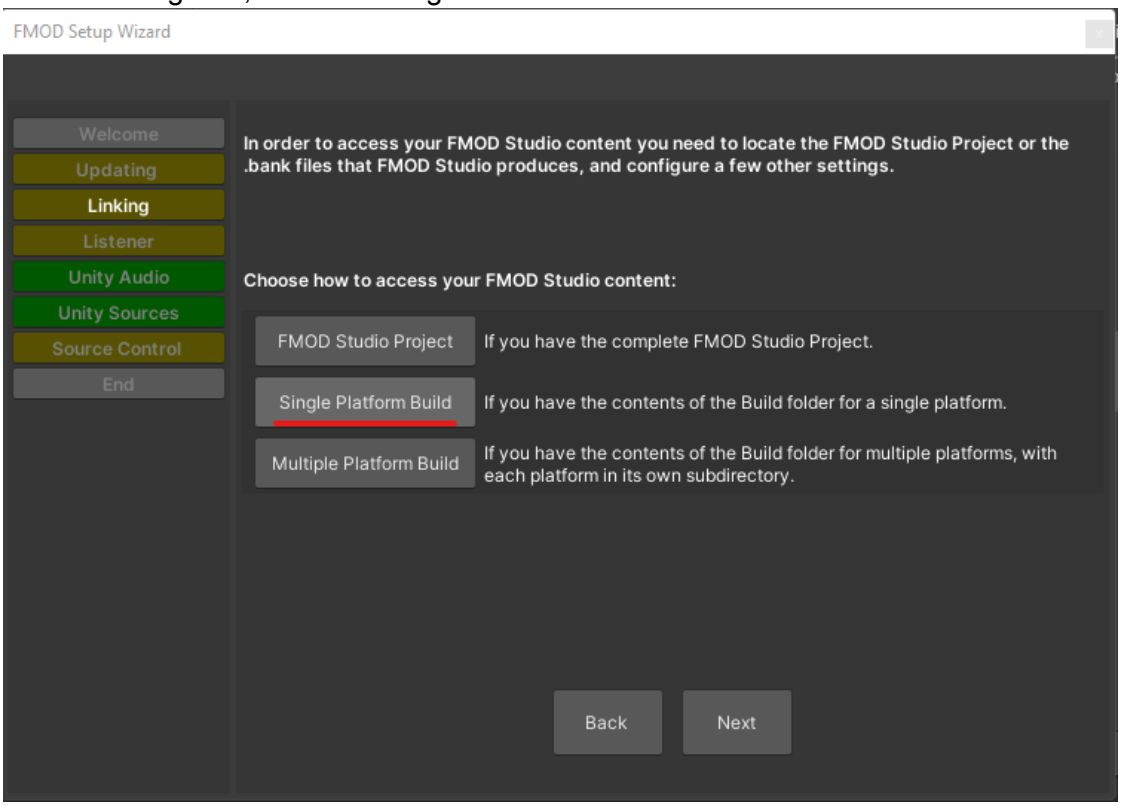
On the "FMOD Setup Wizard" tab, check the box "Do not display this again" and click the "Start" button



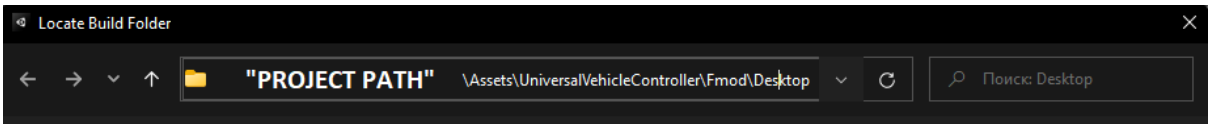
4. In the "Updating" tab, click "Next button".

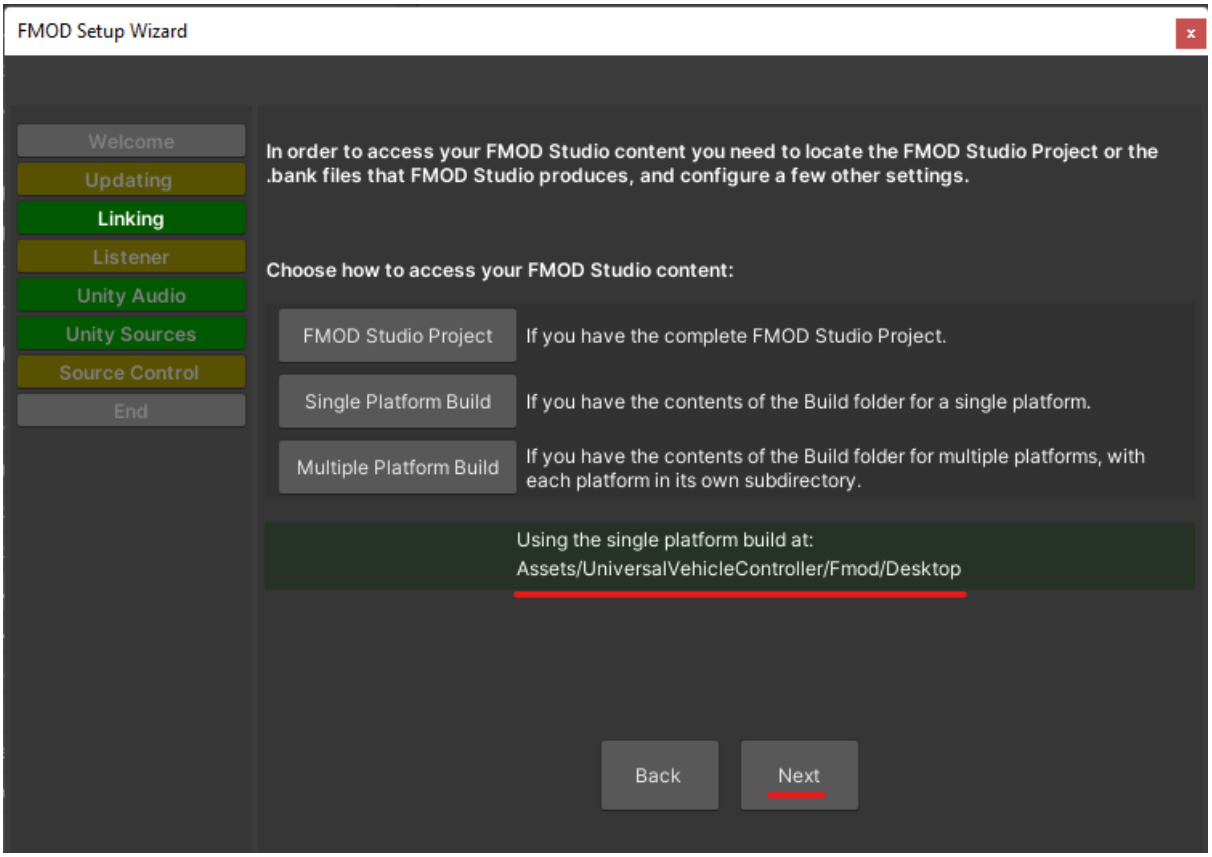


5. In the "Linking" tab, click the "Single Platform Build" button



Specify the path "Project path/Assets/UniversalVehicleController/Fmod/Desktop"

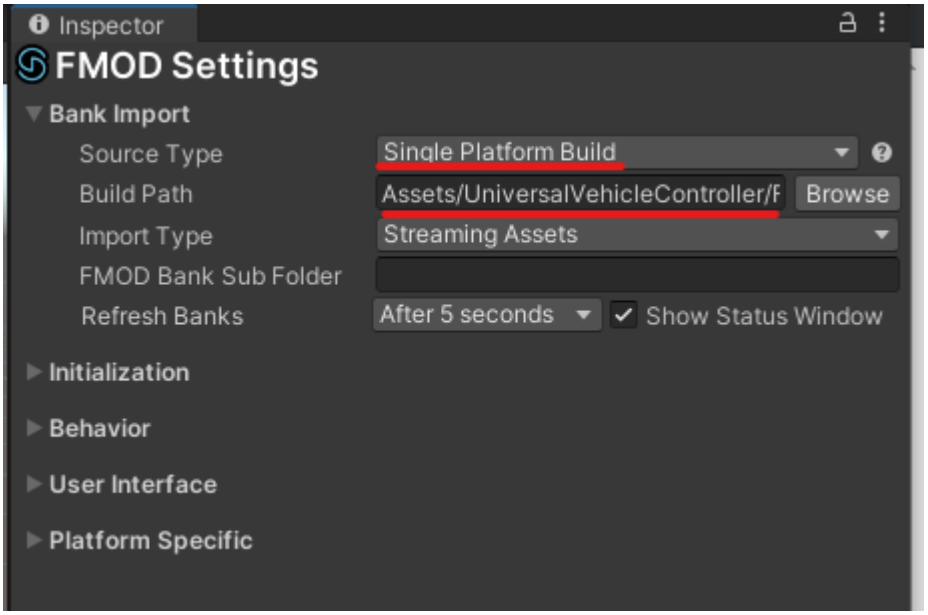




After that, click the "Next" button in all other tabs.

You can also set up FMOD manually:

6. In FMOD Studio Settings select "Single Platform Build"
7. Specify build path "Assets/UniversalVehicleController/Fmod/Desktop"



After that the sound should work, the FMOD project for editing is on the path "Assets/UniversalCarController/Fmod/FmodProject"

To edit a project, you need FMOD Studio 2.00.08 or higher.

When importing this file, the project files will be changed and new ones will be added:

CarSFX.prefab

DieselTruckSFX.prefab

MotorbikeSFX.prefab

MuscleCarSFX.prefab

TrailerSFX.prefab

VehicleSFX.cs

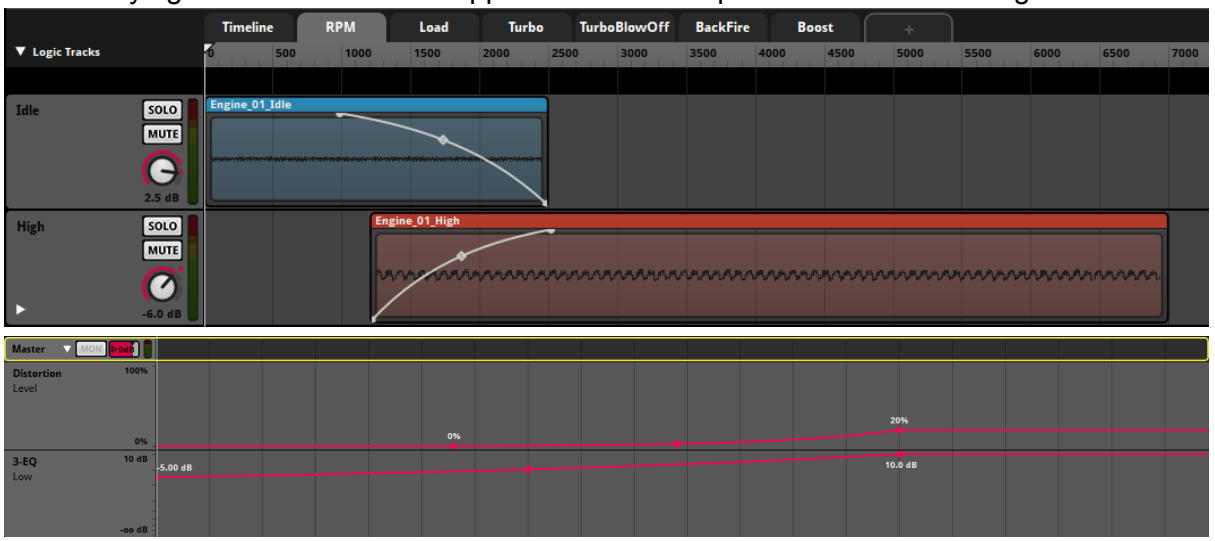
CarSFX.cs

FMOD event example Engine_01 в FMOD.

Engine_01 - This is the most complex Event in the asset:

Engine_01 contains input parameters that are calculated in Unity:

- RPM - the float type accepts a value from 0 to 10000 (For this event the maximum is 7000), affects the tracks Idle (Idle RPM), High (High RPM). Also affects the Master (All Event Tracks) by cutting and modifying the bass. RPM also appears on the Autopitch tracks Idle and High.

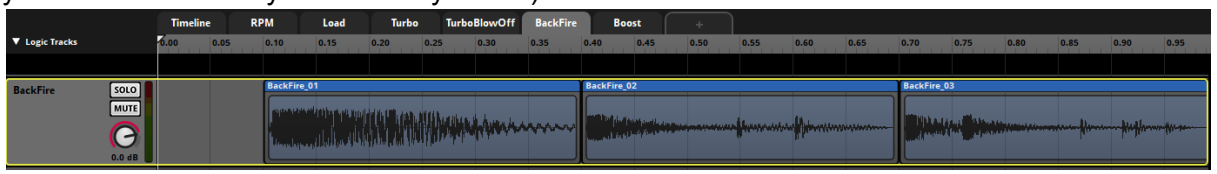


- Load - the float type accepts a value from -1 to 1, affects only the change in low frequencies in the Master.

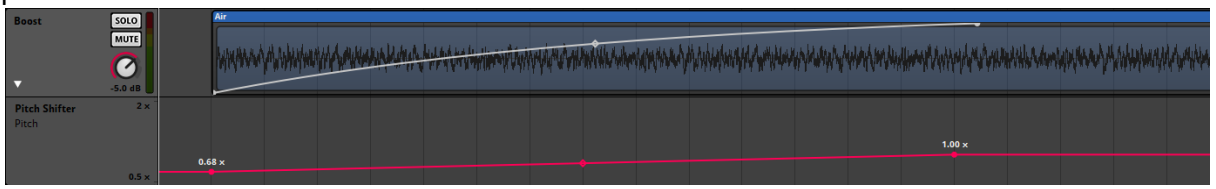
- Turbo - the float type takes a value from 0 to 1, affects the Turbo track (Turbine whistle).

- TurboBlowOff - the float type takes a value from 0 to 1, plays the TurboBlowOff track if the value is higher than 0.2. The closer the value is to 1, the louder the track.

- BackFire float type takes a value from 0 to 1, when a flash of fire occurs, a random number from 0.2 to 1 is sent to this parameter to play different sounds (3 is currently configured, if necessary, you can add as many sounds as you like).



- Boost - the float type takes a value from 0 to 1, affects the Boost track, changes the volume and pitch of the track.



Multiplayer

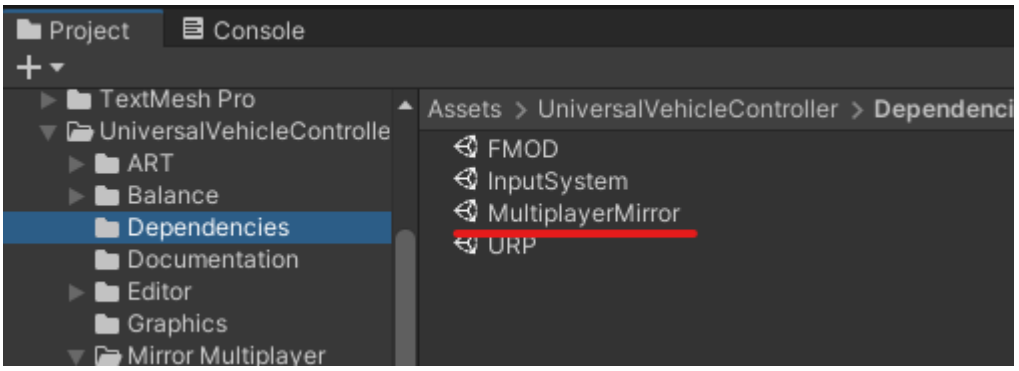
Mirror

By default, KCP transport is used, you can learn more about transport types on the Mirror official website.

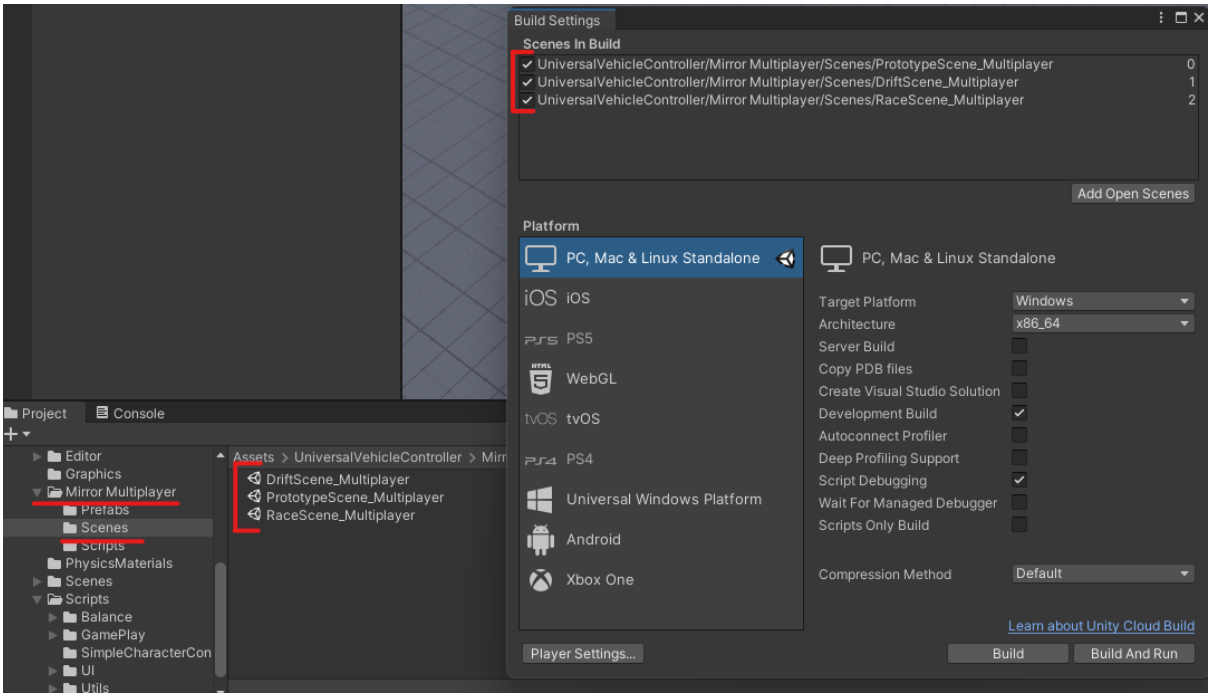
!!!ATTENTION!!! Do not use UVC_NetworkManager and (GameController or SimpleCharacterController) in the same scene. These are 3 different controllers for different types of games.

Install Mirror:

1. Install the plugin MultiplayerMirror.unitypackage



2. Download and install Mirror asset.
3. In BuildSettings remove all scenes and add multiplayer scenes.

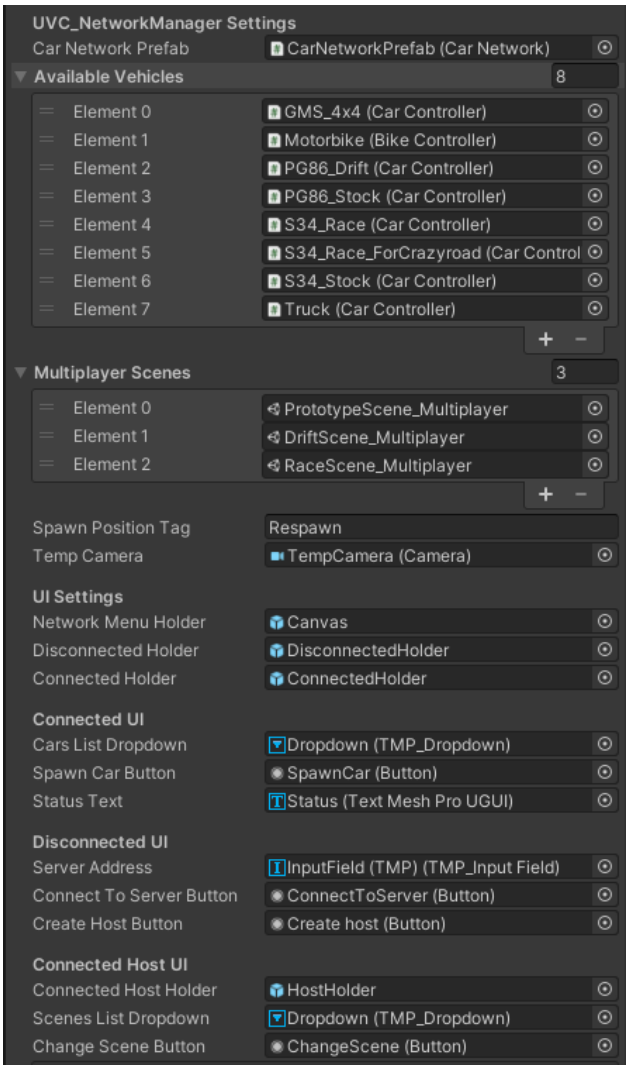


4. Open any of the multiplayer scenes the game is ready to build and run.

MultiplayerMirror components:

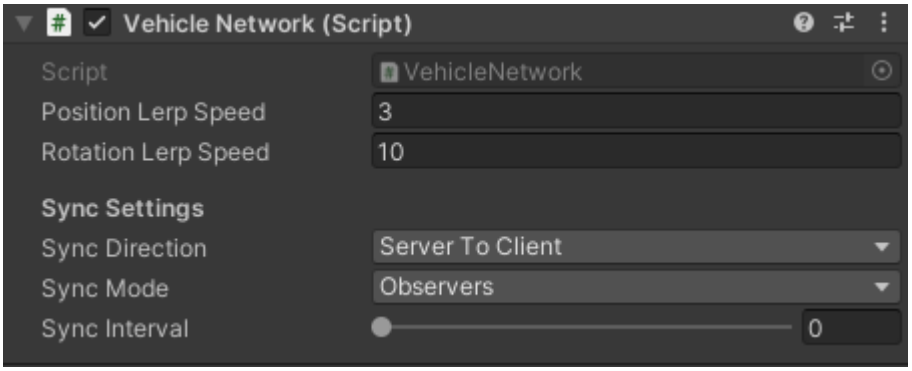
UVC_NetworkManager - heir to Network Manager, synchronizes the creation of cars, also contains the logic for changing scenes.

contains fields:



- CarNetworkPrefab - The prefab is analogous to the PlayerPrefab in the mirror, it synchronizes the state of cars between clients. CarNetworkPrefab is also in the SpawnablePrefabs list.
- AvailableVehicles - list of cars available for multiplayer, all UVC cars are already added to this list, you can delete existing cars or add your own.
- MultiplayerScenes - multiplayer scenes.
- SpawnPositionTag - Tag for finding spawn positions.
- TempCamera - Temporary camera, works while there is no player camera.
- NetworkMenuHolder - GameObject of the main menu that is enabled/disabled when the ESC key is pressed.
- DisconnectedHolder - The GameObject is enabled when there is no connection to the server.
- ConnectedHolder - The GameObject is enabled when there is a connection to the server.
- CarsListDropdown - List of cars available for multiplayer.
- SpawnCarButton - A button that sends a request to the server to spawn a car.
- StatusText - Text for displaying the connection status, it displays: Status (Connected / Disconnected), number of connected clients and client ping.
- ServerAddress - To be able to enter the server address to connect to.
- ConnectToServerButton - Button to connect to the server.
- CreateHostButton - The button for creating a host, when creating a host, it does not matter what is in the ServerAddress.
- ConnectedHostHolder - GameObject that is enabled at the host when the connection is active. Needed to be able to change scenes.
- ScenesListDropdown - List of available scenes.
- ChangeSceneButton - Scene change button, when pressed, loads a scene from ScenesListDropdown.

VehicleNetwork - heir to[NetworkBehaviour](), synchronizes the physical parameters of the car: position, rotation, velocity, angularVelocity. Also synchronizes damage.
Contains fields:



- PositionLerpSpeed - Position synchronization speed. When synchronizing, the car moves smoothly to the synchronized position.
- RotationLerpSpeed - Rotation sync speed. When synchronized, the vehicle turns smoothly towards the synchronized rotation.

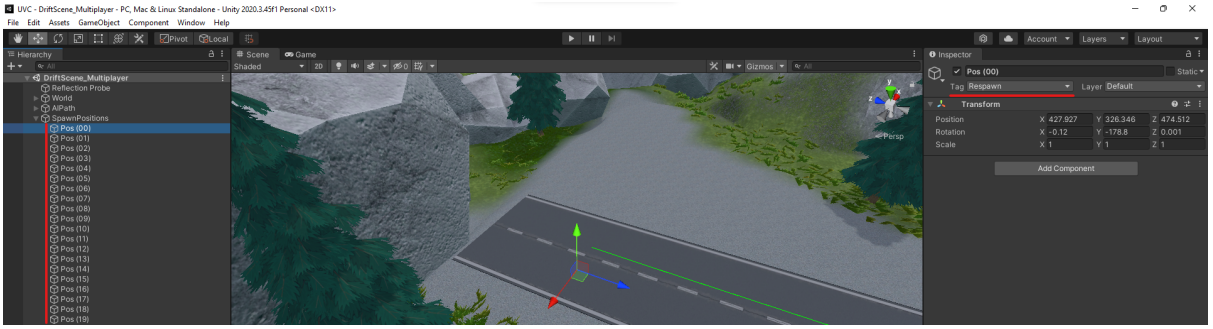
CarNetwork - heir to VehicleNetwork, synchronizes control, state of lighting devices. Has no public fields.

Adding cars:

Simply add the created vehicle prefab to the UVC_NetworkManager.AvailableVehicles of the UVC_NetworkManager prefab.

Adding scenes:

- Add positions for cars to spawn on the scene and specify the desired tag for them (Default “Respawn”), players' cars will spawn at these points, 1st player - 1st point, 2nd player - 2nd point, etc.



- Add the UVC_NetworkManager prefab to the scene, make sure there are no GameController and SimpleCharacterController on the scene.
- Add this scene to the UVC_NetworkManager.MultiplayerScenes of the UVC_NetworkManager prefab.

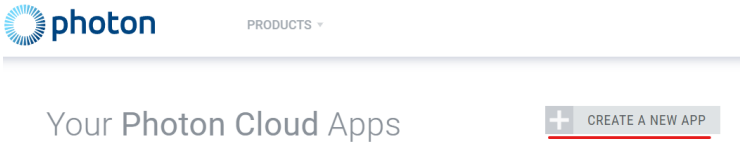
Photon

Photon multiplayer uses PUN 2.
Documentation PUN.

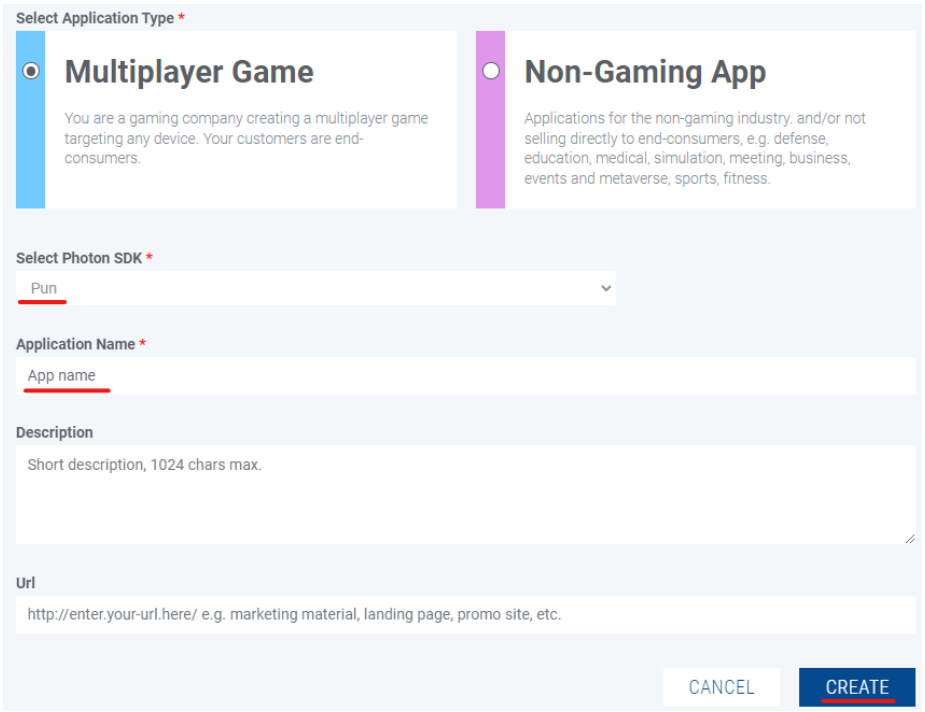
!!!ATTENTION!!! Do not use UVC_NetworkManager and (GameController or SimpleCharacterController) in the same scene. These are 3 different controllers for different types of games.

Preparing the application in photonengine

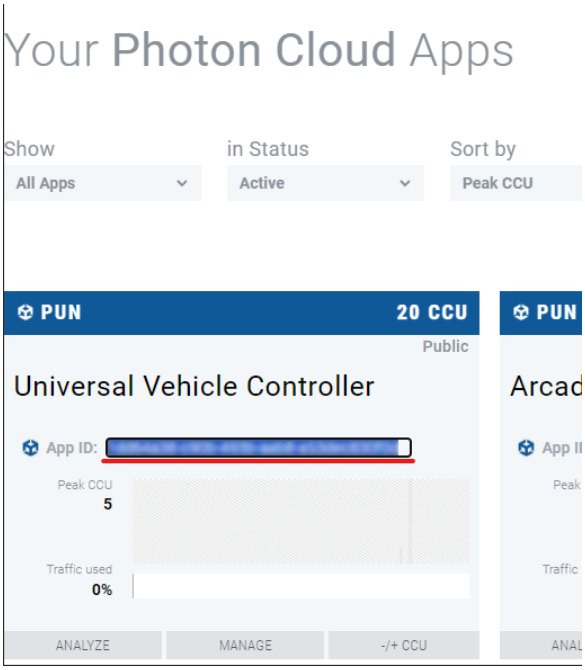
1. Login or register in photonengine.
2. Go to the page dashboard.
3. Click the button “Create a new app”



4. Select “Pun” in “Select Photon SDK”, enter the application name and click the button “Create”

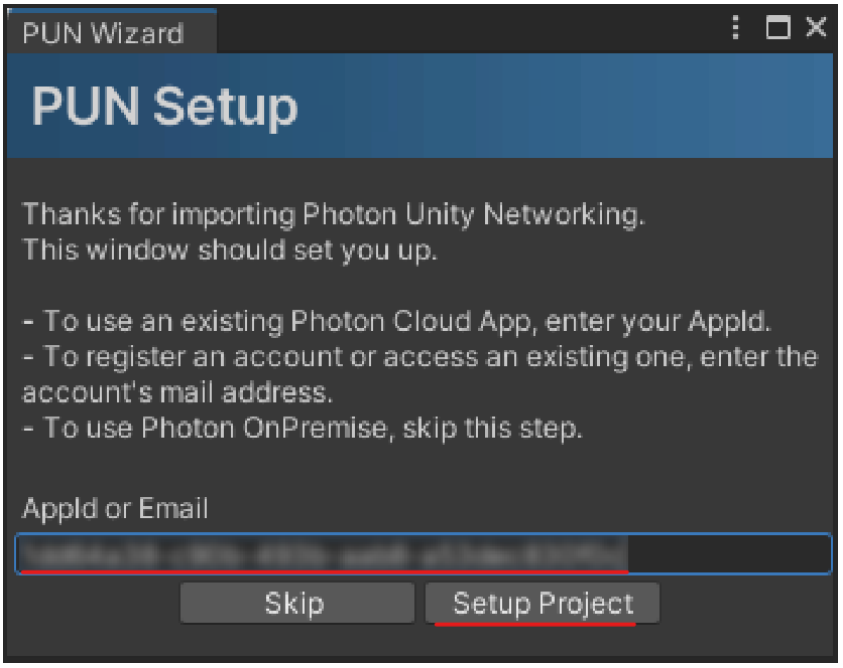


5. After creating the application, copy the App ID, we will need it in the editor

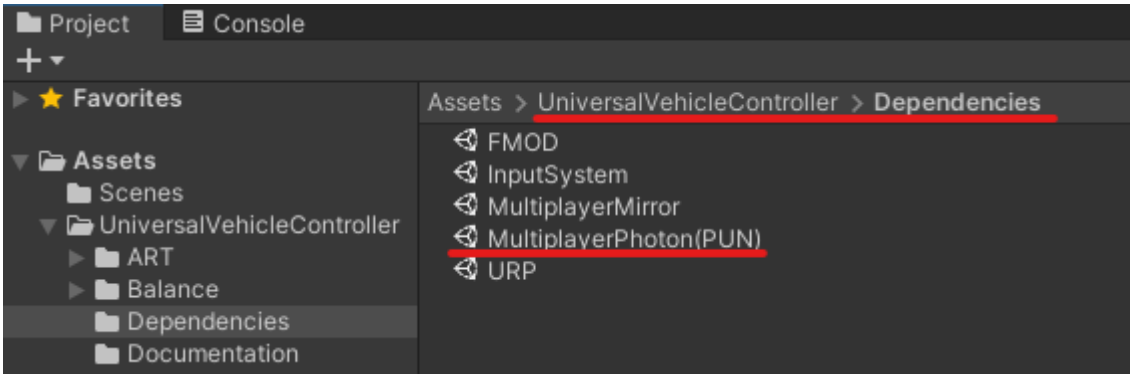


Installing Photon:

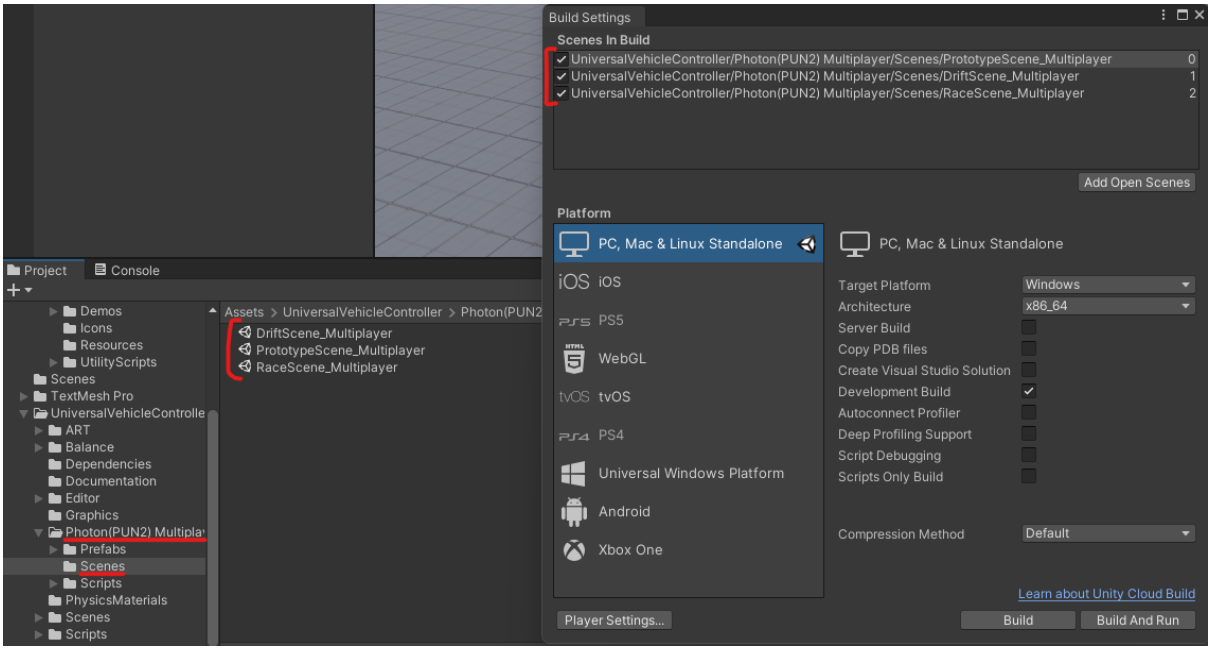
1. Download and install the PUN 2 asset.
2. After installation, paste the copied App ID and click the “Setup Project” button



3. Install the MultiplayerPhoton(PUN).unitypackage plugin



4. In BuildSettings remove all scenes and add multiplayer scenes.

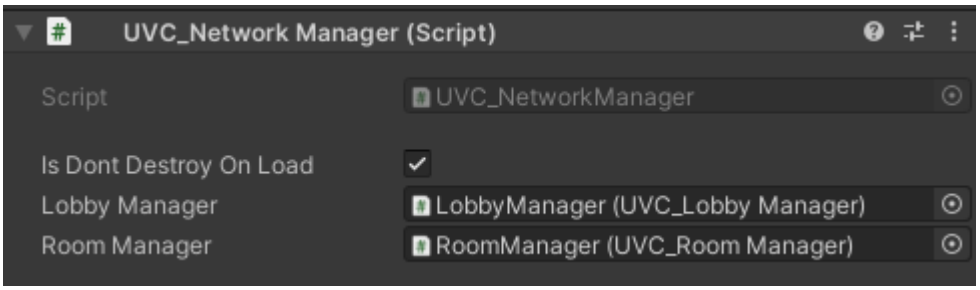


1. Open any of the multiplayer scenes the game is ready to build and run.

Photon components:

UVC_NetworkManager - implements the IConnectionCallbacks and IMatchmakingCallbacks interfaces for connection control. For correct operation it must always be active.

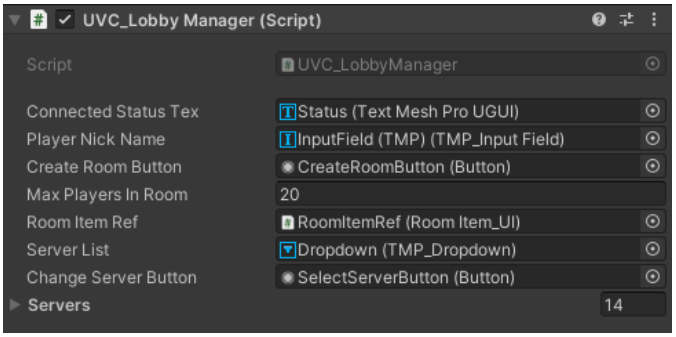
contains fields:



- IsDontDestroyOnLoad - must be true for the transition between scenes to work correctly, you can change the logic at your discretion.
- LobbyManager - link to UVC_LobbyManager, enabled when there is no connection to the room.
- RoomManager - link to UVC_RoomManager, enabled when there is a connection to the room.

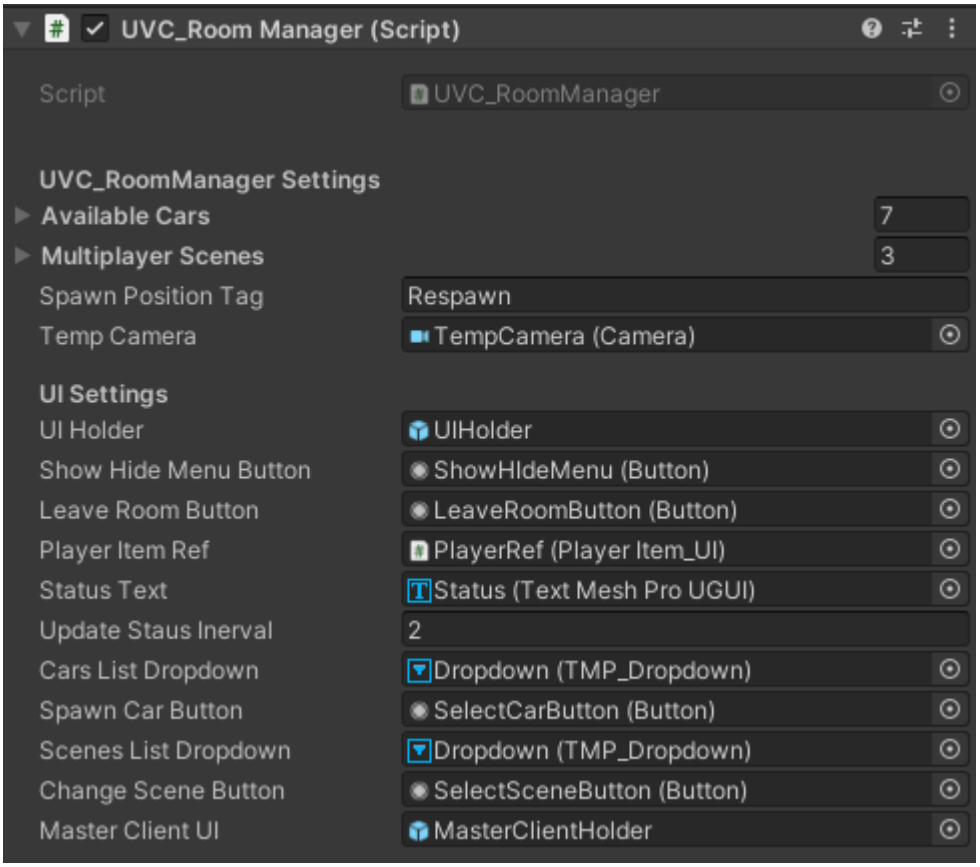
UVC_LobbyManager - implements the ILobbyCallbacks interface, manages the logic in the lobby (Searching for rooms, connecting to a room, creating rooms, selecting a region).

contains fields:



- ConnectedStatusTex - TextMeshProUGUI which displays the connection status; when disconnected, the reason for disconnection is also displayed.
- PlayerNickName - InputField in which you can enter the player's NickName.
- CreateRoomButton - a button that creates a room when clicked.
- MaxPlayersInRoom - maximum number of players, default is 20, you can add your own logic to change the number of players in the room.
- RoomItemRef - UI element displaying information about the room.
- ServerList - A list to which all available regions will be added.
- ChangeServerButton - a button that, when clicked, changes the region to the one selected in the ServerList.
- Servers - available regions, when changing the available regions, also change them inPhoton.

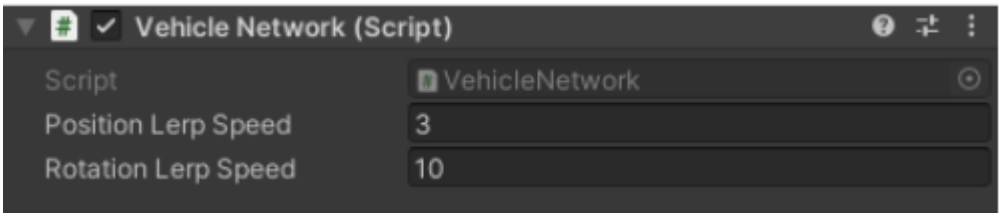
UVC_RoomManager - implements the IInRoomCallbacks interface, controls the logic in the room (Selecting a car, changing the scene only for the owner of the room).
contains fields:



- AvailableCars - list of cars available for multiplayer, all UVC cars have already been added to this list, you can remove existing cars or add your own.
- MultiplayerScenes - multiplayer scenes, below we will describe how to add your own scenes.
- SpawnPositionTag - Tag for searching spawn positions.
- TempCamera - Temporary camera, works while there is no player camera.
- UIHolder - GameObject of the main menu that turns on when you press the ESC key.
- ShowHideMenuButton- the button that turns on turns off UIHolder.
- LeaveRoomButton- a button that, when pressed, exits the room.
- PlayerItemRef - UI element for displaying players in the room.
- StatusText - Text to display the connection status, it displays: Status (Connected/Disconnected), the number of connected clients and the client ping.
- UpdateStausInerval - status update interval.
- CarsListDropdown - List of cars available for multiplayer.
- SpawnCarButton - A button sending a request to the server to create a car.

- ScenesListDropdown - List of available multiplayer scenes.
- ChangeSceneButton - Scene change button, when pressed, loads the scene from ScenesListDropdown.
- MasterClientUI- The scene selection GameObject will only be enabled on the master client.

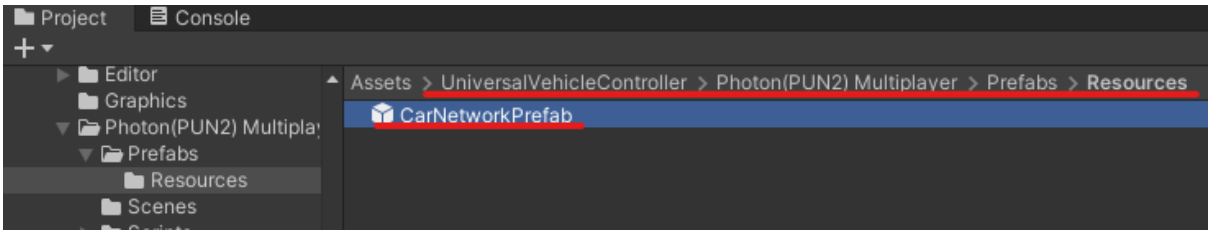
VehicleNetwork - heir to MonoBehaviourPunCallbacks, IPunObservable, synchronizes the physical parameters of the car: position, rotation, velocity, angularVelocity. Also synchronizes damage. Contains fields:

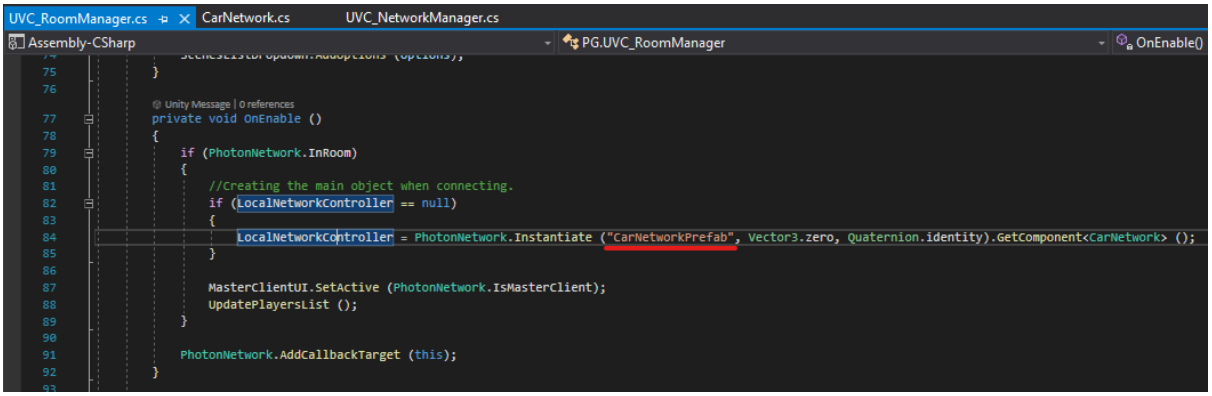


- PositionLerpSpeed - Position synchronization speed. When synchronizing, the car moves smoothly to the synchronized position.
- RotationLerpSpeed - Rotation sync speed. When synchronized, the vehicle turns smoothly towards the synchronized rotation.

CarNetwork - heir to VehicleNetwork, synchronizes control, state of lighting devices. Has no public fields.

The prefab with the CarNetwork component is located in Resources named CarNetworkPrefab, Photon finds it by name, renaming or moving this prefab is critical.





Synchronization logic:

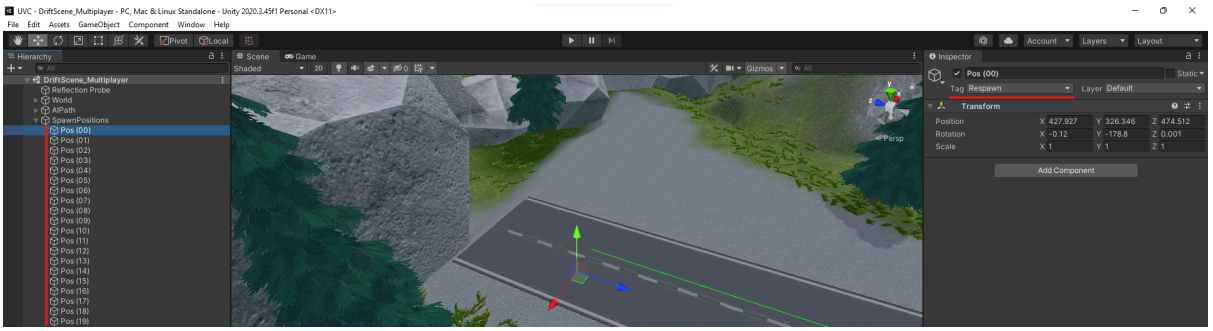
When a player connects to a room, he creates a CarNetworkPrefab for all players; through this prefab, the creation and parameters of the selected car are synchronized. CarNetworkPrefab is placed in UVC_NetworkManager when created and is not deleted when loading scenes.

Adding cars:

Simply add the created vehicle prefab to the UVC_NetworkManager.UVC_RoomManager.AvailableVehicles of the UVC_NetworkManager prefab.

Adding scenes:

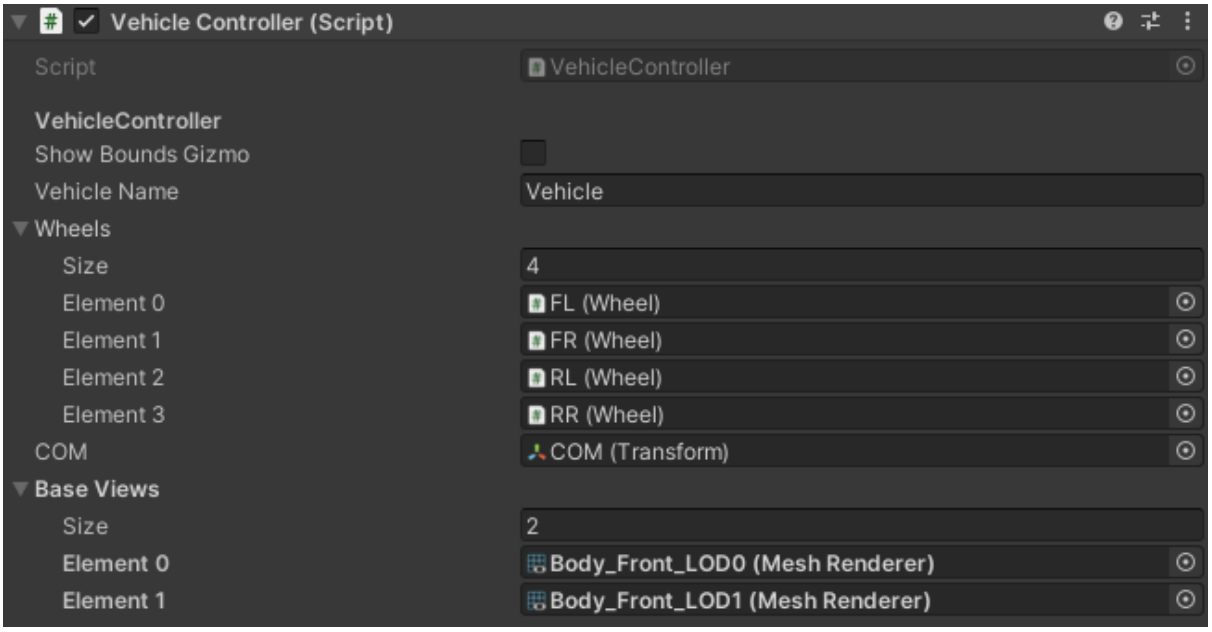
- Add positions for cars to spawn on the scene and specify the desired tag for them (Default “Respawn”), players' cars will spawn at these points, 1st player - 1st point, 2nd player - 2nd point, etc.



- Add the UVC_NetworkManager prefab to the scene, make sure there are no GameController and SimpleCharacterController on the scene.
- Add this scene to the UVC_NetworkManager.UVC_RoomManager.MultiplayerScenes of the UVC_NetworkManager prefab.

VehicleController.cs

Base class for all types of vehicles, includes:

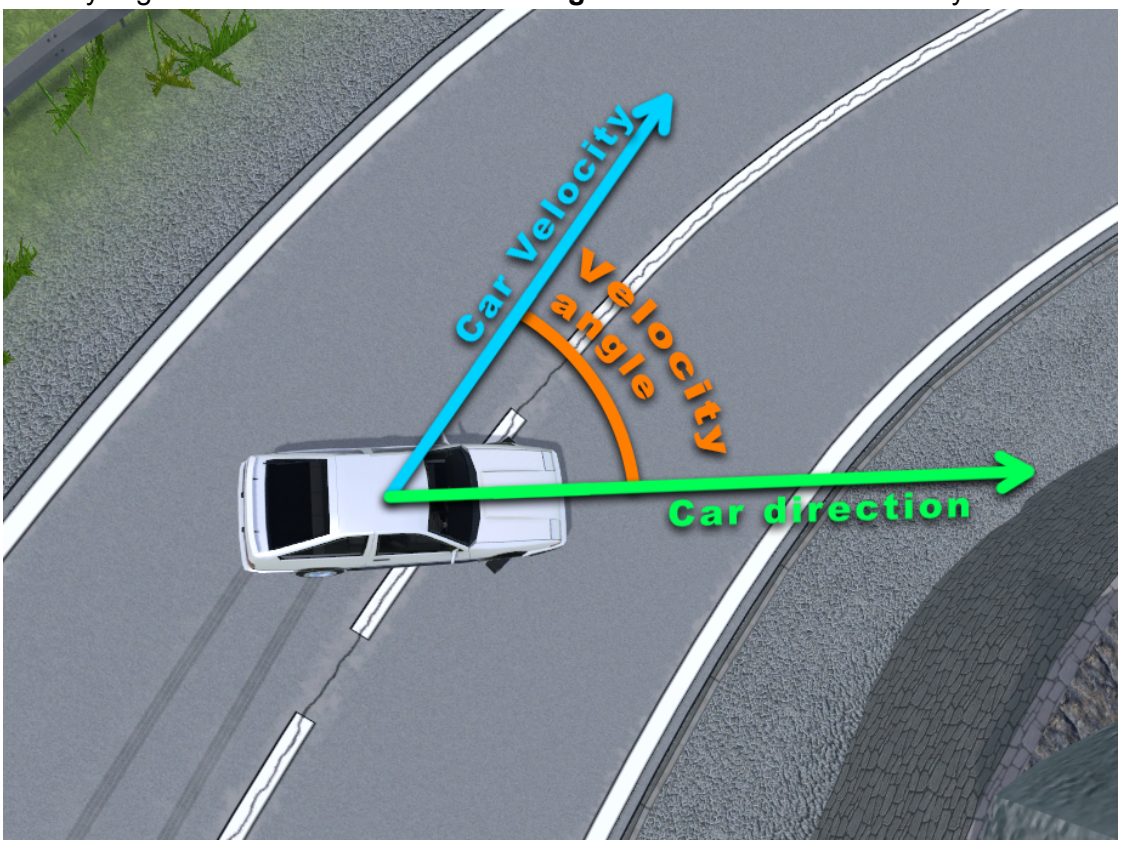


- VehicleName - Vehicle name, the field is needed to find the original prefab of the car when restart the vehicle.
- Wheels – Reference to ***Wheel*** types of wheels (Wheel class with be explained later)
- COM – Center Of Mass of the vehicle. It heavily affects vehicle behavior (the lower the COM, the more the vehicle is stable. It's better to place it like in real-world vehicle
- Base Views – Array of type Renderer. Used to determine the visibility of the vehicle VehicleController.VehicleVisible (this property is used to check for playing sound and visual effects). The array is used as some parts may be part of the LOD group (like in this case). You should add vehicle body parts in the array.

  This class has logic to reset the vehicle (right now the vehicle is being reset at the point where it is. You can create custom reset behavior replacing "ResetVehicle" method) and logic for vehicle collisions. If you want to track collision in any class you can simply subscribe to Action<VehicleController, Collision> CollisionAction.

These properties are being calculated:

- VehicleIsVisible - returns **true** if any of the cameras renders any Renderer from Base View array
- VehicleIsGrounded - returns **true** if at least one wheel is touching the ground
- CurrentSpeed is measured in units (meters) per second
- SpeedInHour is measured in **kph** or **mph** depending on GameSettings.asset
- VehicleDirection - direction of the vehicle movement
- VelocityAngle - vehicle movement vector **angle** relative to the vehicle body



CarController.cs

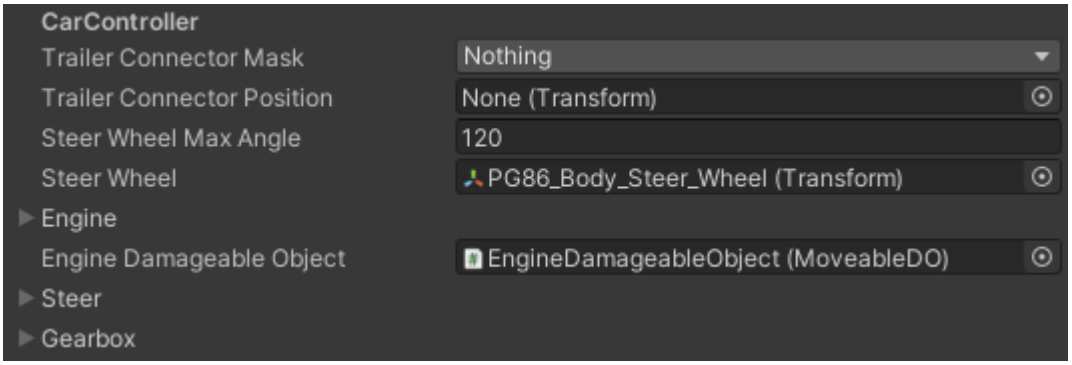
CarController inherits VehicleController, it contains all logics and physics for the vehicle. To simplify things this class is divided into 4 different classes:

- CarController.cs – general fields, properties and methods like information about steering wheel, trailer etc.
- Engine.cs – engine's behavior, calculating engine's RPM, load, rev limiter, turbo, boost etc.
- Steering.cs – vehicle handling, wheel turning and steering helpers which makes the handling feel arcade-like
- Transmission.cs – calculates the torque transferred from the engine to the wheels. All the logic for shifting the gears is presented here (automatic and manual)

Next will be more detailed information about each of the components.

CarController.cs

Contains following fields:

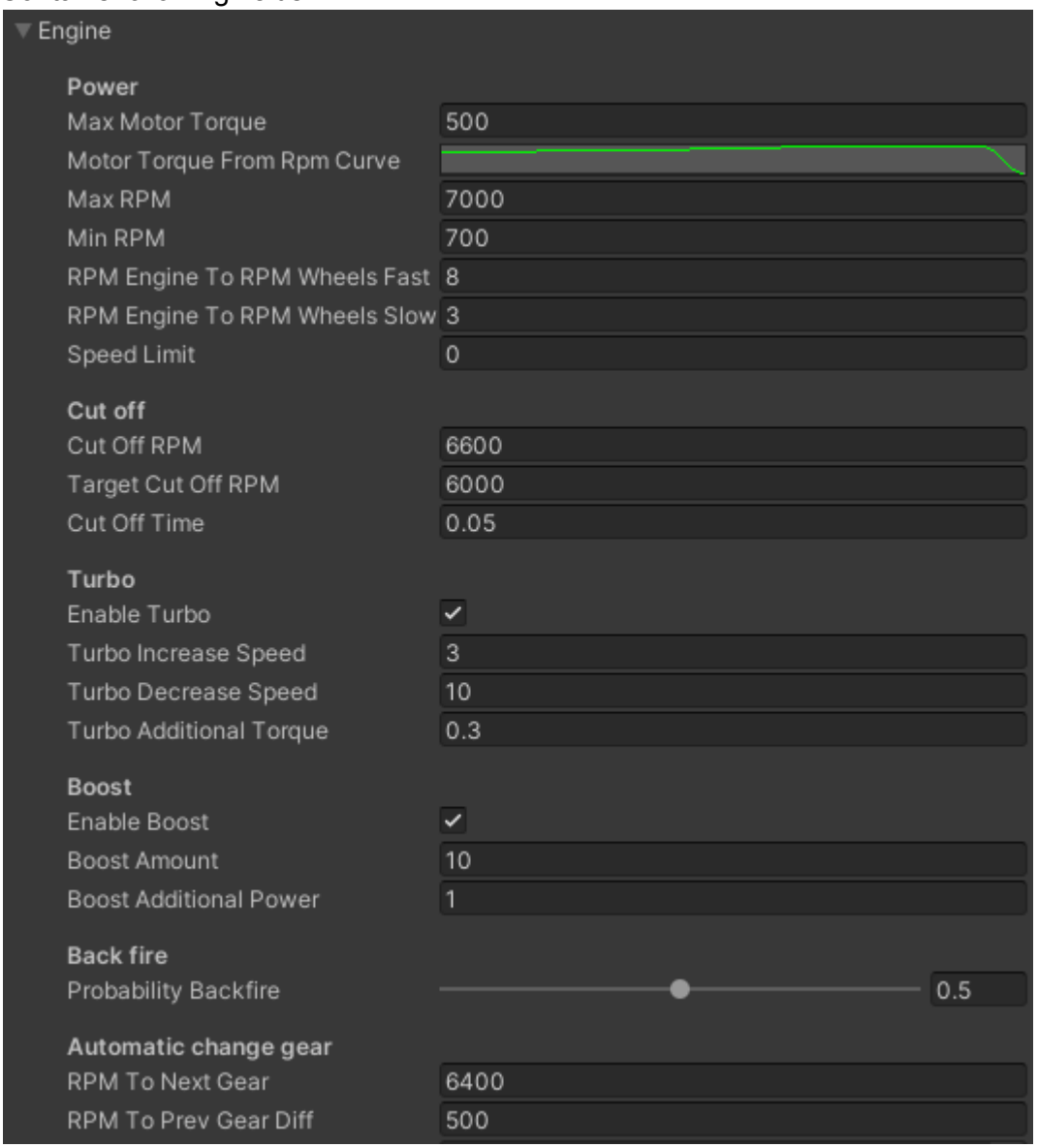


- TrailerConnectorMask – Mask for connecting the trailer
- TrailerConntectorPosition – Point where the trailer connects to the vehicle
- SteerWheelMaxAngle – Max angle of steering wheel rotation (only visually)
- SteerWheel – reference to the steering wheel
- EngineDamageablePbject - The damage to the engine depends on the location of this object. If this object is Null, the engine will not be damaged.

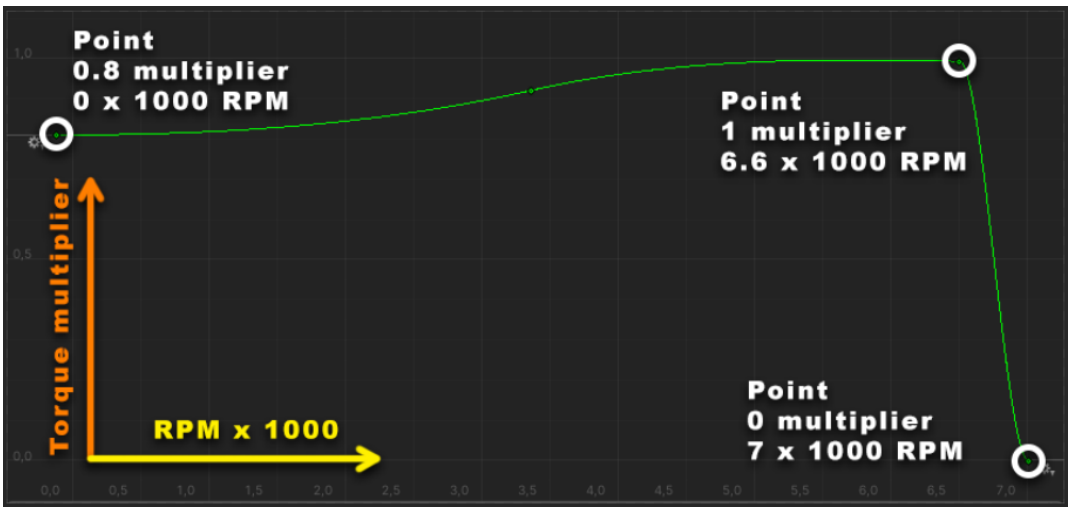
Engine.cs

Contains only Engine field of type EngineConfig. This was done to have the possibility to swap this config (for ex. Vehicle tuning or engine swap).

Contains following fields:



- MaxMotorTorque – Motor torque transferred to the wheels. Changing this parameter, you can easily change engine power of the vehicle
- MotorTorqueFromRpmCurve – torque curve of the engine. Recommended size of the curve X is MaxRPM / 1000 and Y is 1

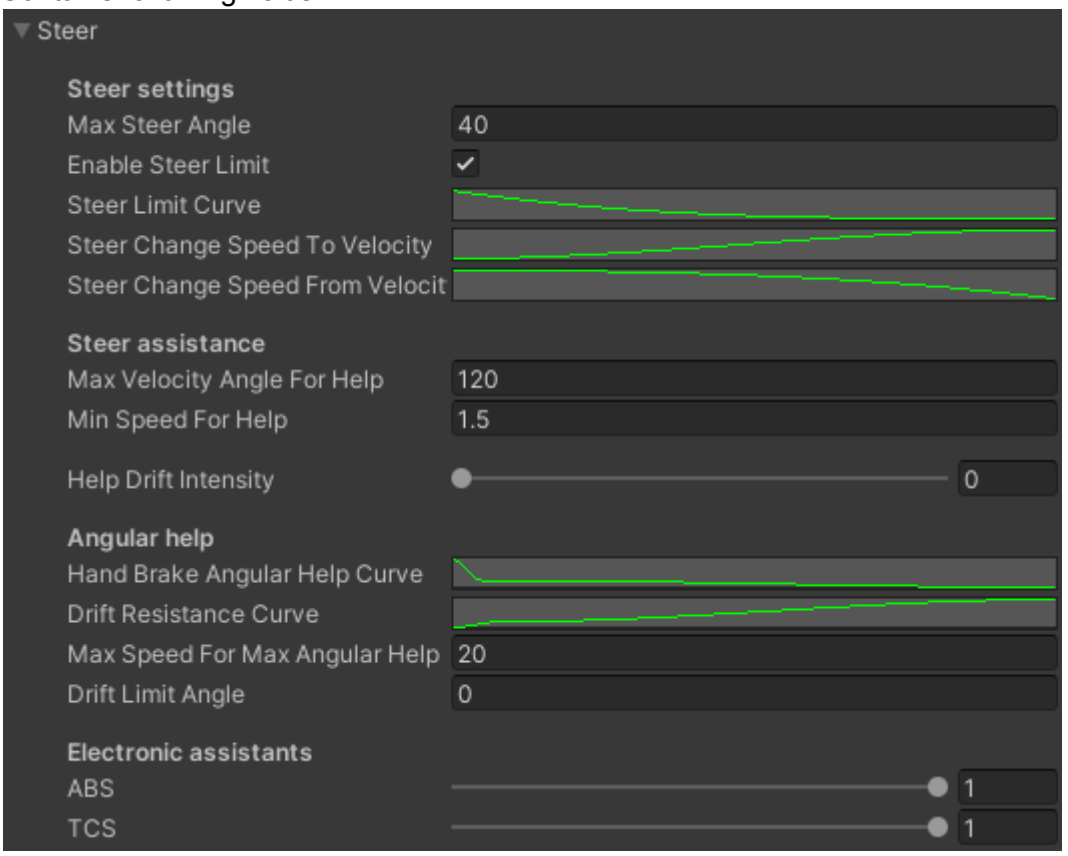


- MaxRPM – Max RPM of the engine. Used for calculating max angle of RPM Needle
- MinRPM – Idle RPM
- RPMEngineToRPMWheelsFast – Speed of increasing RPM (Accelerating)
- RPMEngineToRPMWheelsSlow - Speed of decreasing RPM (Decelerating)
- SpeedLimit - Speed limit, measured in units per second (1ups == 3.6 kph == 2.236 mph), limits engine power. Limit is disabled if value == 0.
- CutOffRPM – Value when rev limiter kicks in. Engine can not rev higher than this
- TargetCutOffRPM – Value to which RPM drops
- CutOffTime – Time of dropping the RPM to TargetCutOffRPM when rev limiter is kicked in
- EnableTurbo - Enables / Disables the turbo of the car.
- TurboIncreaseSpeed - The speed at which the turbo value increases.
- TurboDecreaseSpeed - The speed at which the turbo value decreases.
- TurboAdditionalTorque - Additional engine power multiplier at maximum turbo.
- EnableBoost - Enables / Disables boost (Nitro) for the car.
- BoostAmount - The amount of boost, measured in seconds.
- BoostAdditionalPower - Additional engine power multiplier at maximum when using boost.
- ProbabilityBackfire – the probability of backfire to happen from the exhaust pipe, 0 – never, 1 – always.
- MaxBrakeTorque – Brake force. Transferred to all wheels.
- RPMToNextGear – RPM when shifting to next gear will happen if automatic gearbox is selected and all the conditions are satisfied in Transmission.cs
- RPMToPrevGear - Downshift if current rpm + RPMToPrevGearDiff is less than previous gear shift. Will happen if automatic gearbox is selected and all the conditions are satisfied in Transmission.cs

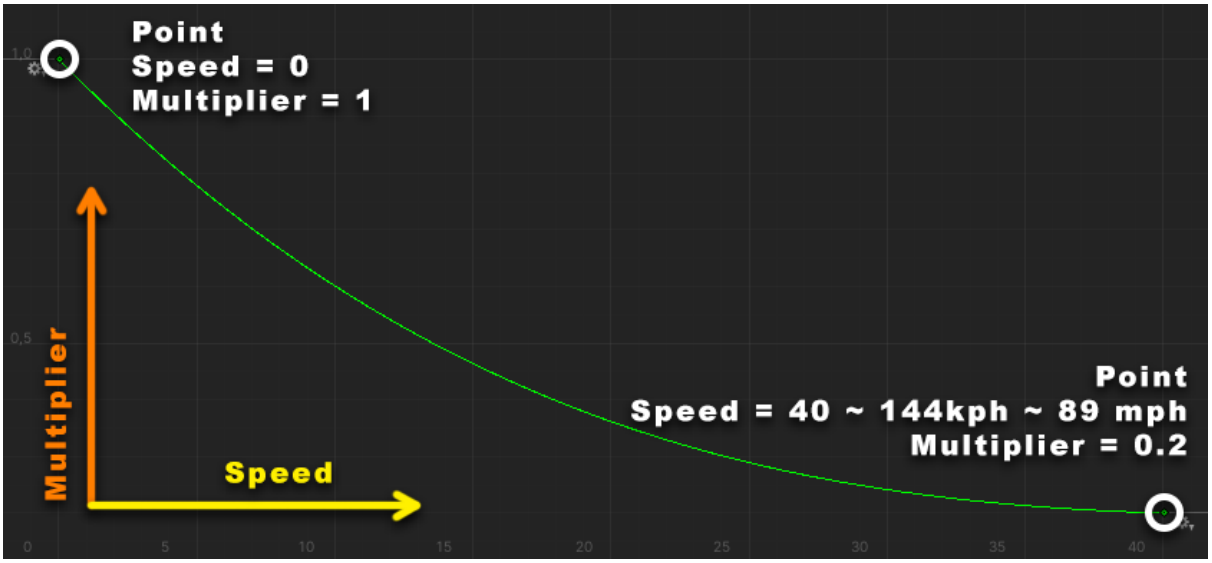
Steering.cs

Contains only Steer field of type SteerConfig. This was done to have the possibility to swap this config (for ex. Vehicle tuning or suspension swap).

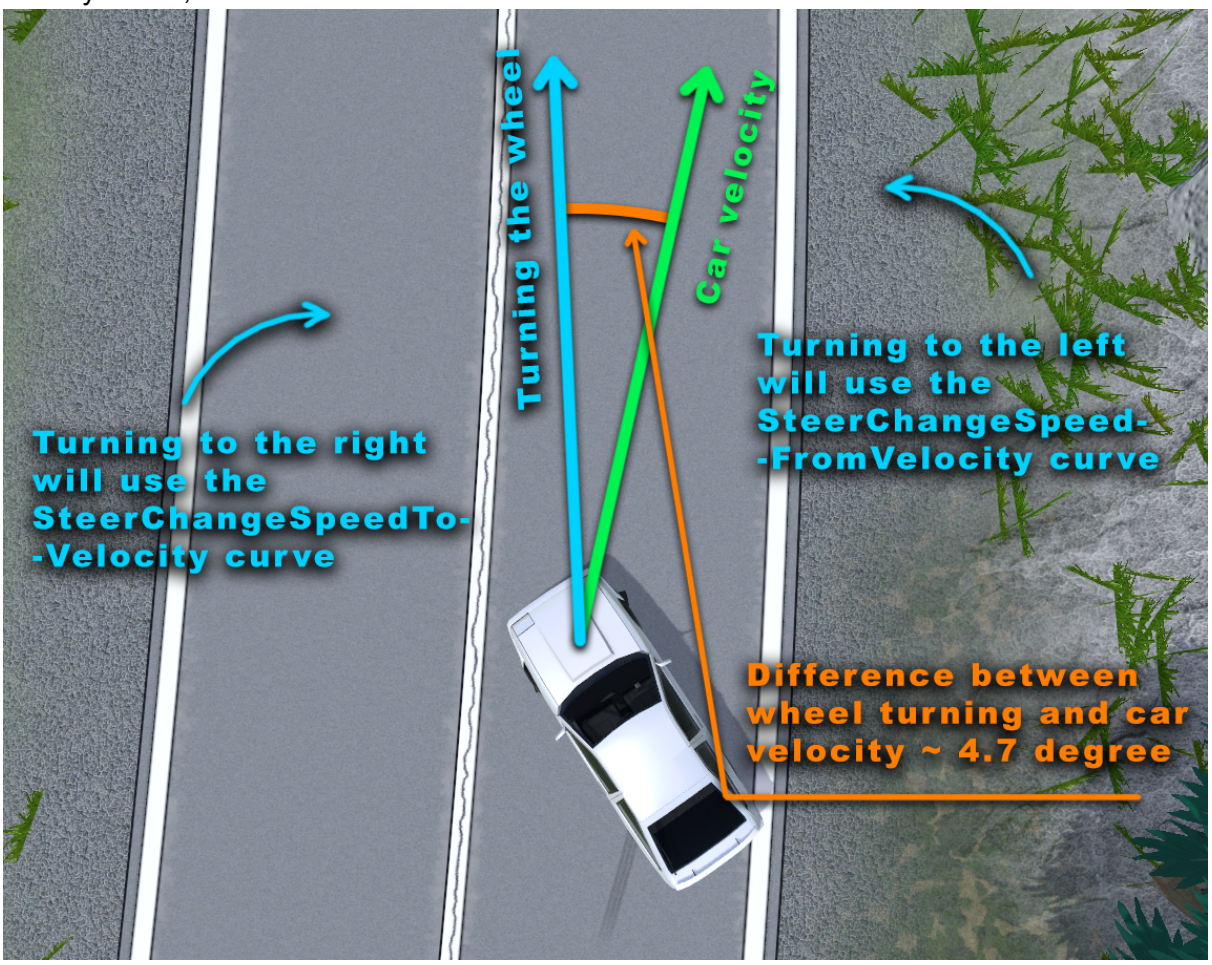
Contains following fields:



- MaxSteerAngle – Max steering angle of the wheels
- EnableSteerLimit – If enabled limits the angle of steering depending on the speed
- SteerLimitCurve – Limit curve if Steer limit is enabled

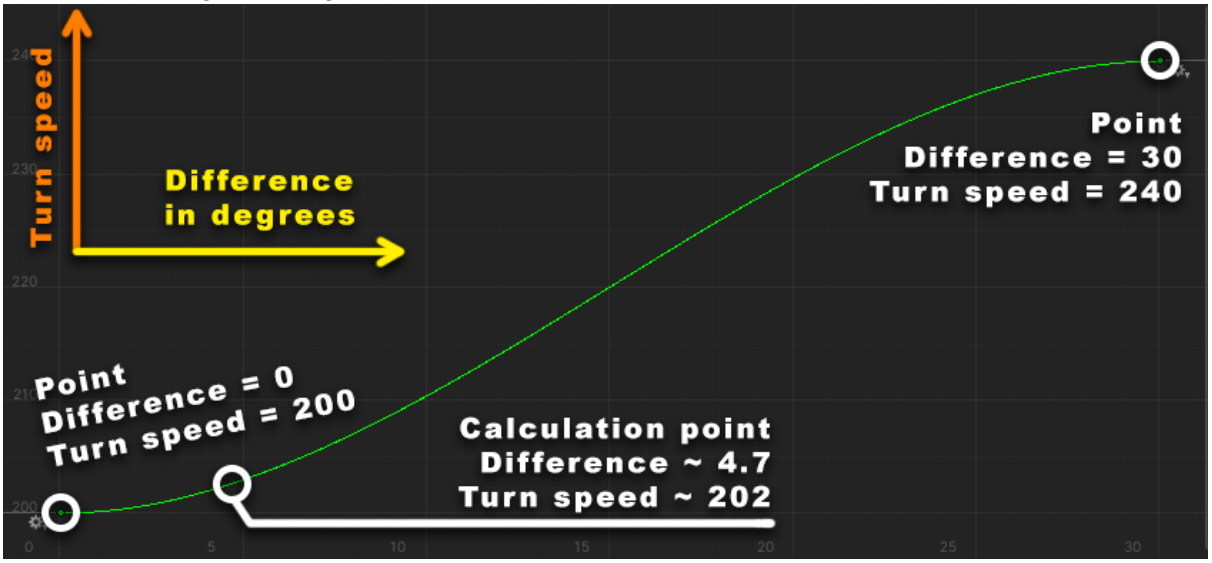


- SteerChangeSpeedToVelocity – turning speed of the wheels towards car velocity (towards car velocity steering wheel is turning faster and easier), the further the wheel is turned from car velocity vector, the faster it turns towards it.



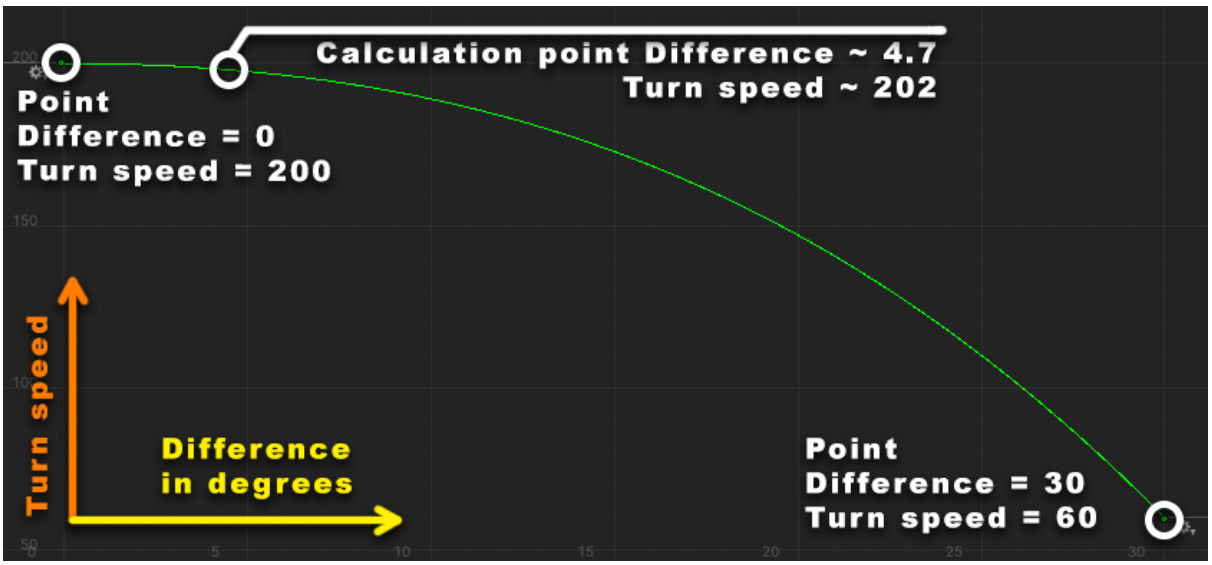
Picture shows SteerChangeSpeedToVelocity and SteerChangeSpeedFromVelocity.

Curve when turning to the right:



- SteerChangeSpeedFromVelocity – turning speed of the wheels to the opposite side of car velocity. (from car velocity steering wheel is turning slower and tougher), the further the wheel is turned from car velocity vector, the slower it turns further from it

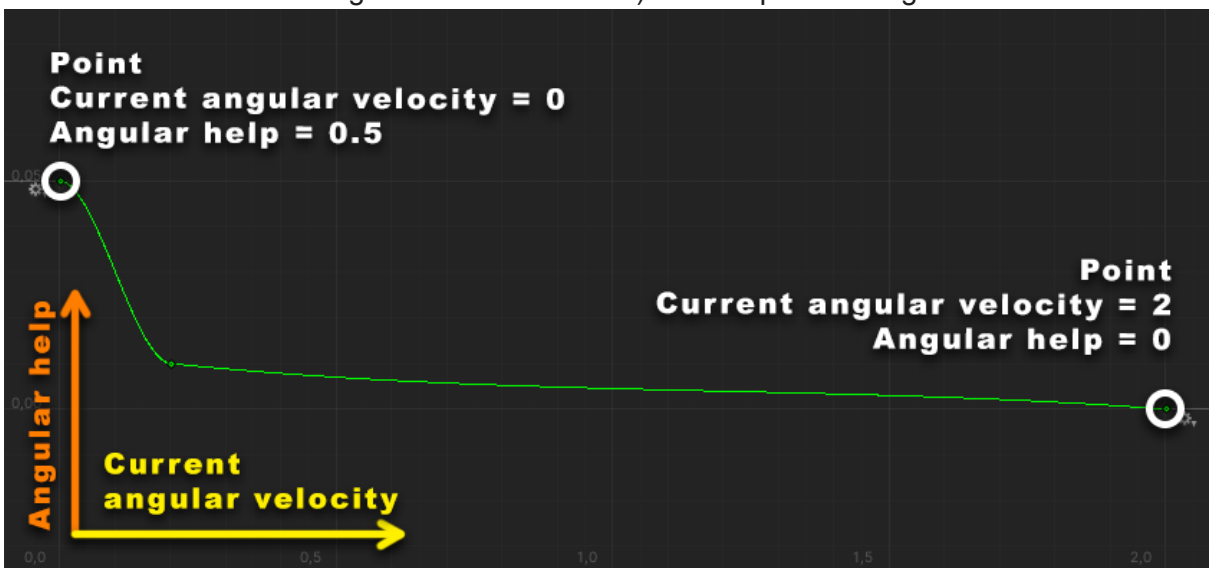
Curve when turning to the right:



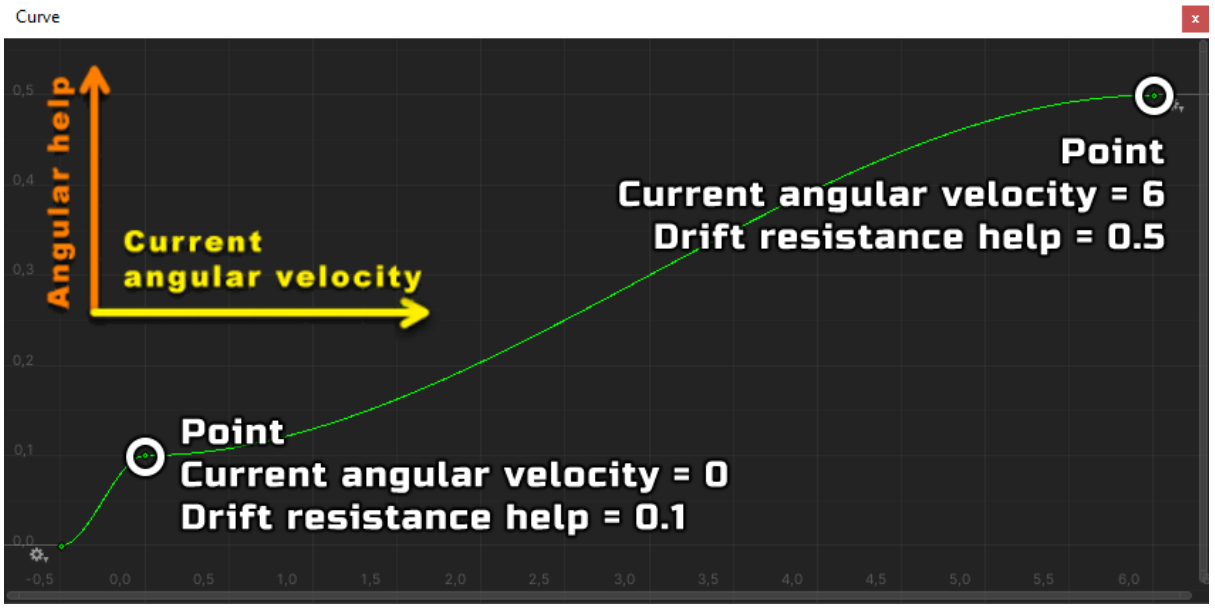
- HelpDriftIntensity – intensity of steering towards drifting, 0 – no help, 1 – max help. Variable only affects turning of the wheels
- MaxVelocityAngleForHelp – max angle when steering help is active (if angle is greater, most probably the car will start to slide)
- MinSpeedForHelp – Min speed when steering help is active

Next go variables affecting angular velocity (force which affect the vehicle regardless of its wheels direction) they give feeling of arcade-like controls

- HandBrakeAngularHelpCurve – Curve that changes the angular velocity of the vehicle when the handbrake is applied. It is necessary to set the initial impulse of rotation (to simulate skidding on cars with wheels with strong friction with the road) or to help with a negative rotation force.



- DriftResistanceCurve – curve to adjust drift resistance. Change multiplier of angular velocity if there is input from angular velocity and VelocityAngle. Depends on current vehicle's angular velocity



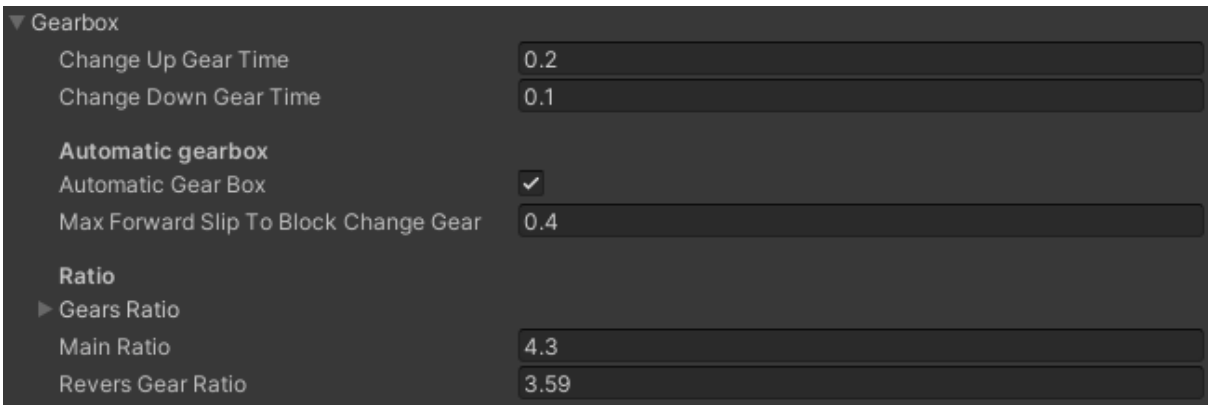
Currently the curve is such that the greater the current angular velocity is from the preferable direction (Horizontal input), the more the angular velocity changes.

- MaxSpeedForMaxAngularHelp - The speed at which the maximum values are used, for example, at a speed of 0, the assistants in changing the angular velocity will not work.
- DriftLimit - Drift limit, measured in degrees, when the limit is on, the control becomes more arcade. Limit is disabled if value == 0.
- ABS - The anti-lock braking system prevents the wheels from blocking when braking, which results in the most efficient braking. 0 - ABS is disabled, 1 - ABS is working at maximum.
- TCS - Traction control system prevents wheel slip during acceleration (Engine power is limited) due to which the most efficient acceleration is achieved. 0 - TCS is disabled, 1 - TCS is working at maximum.

Transmission.cs

Contains only the Gearbox field of type GearboxConfig. This was done to have the possibility to swap this config (for ex. Vehicle tuning or gearbox swap).

Contains fields:

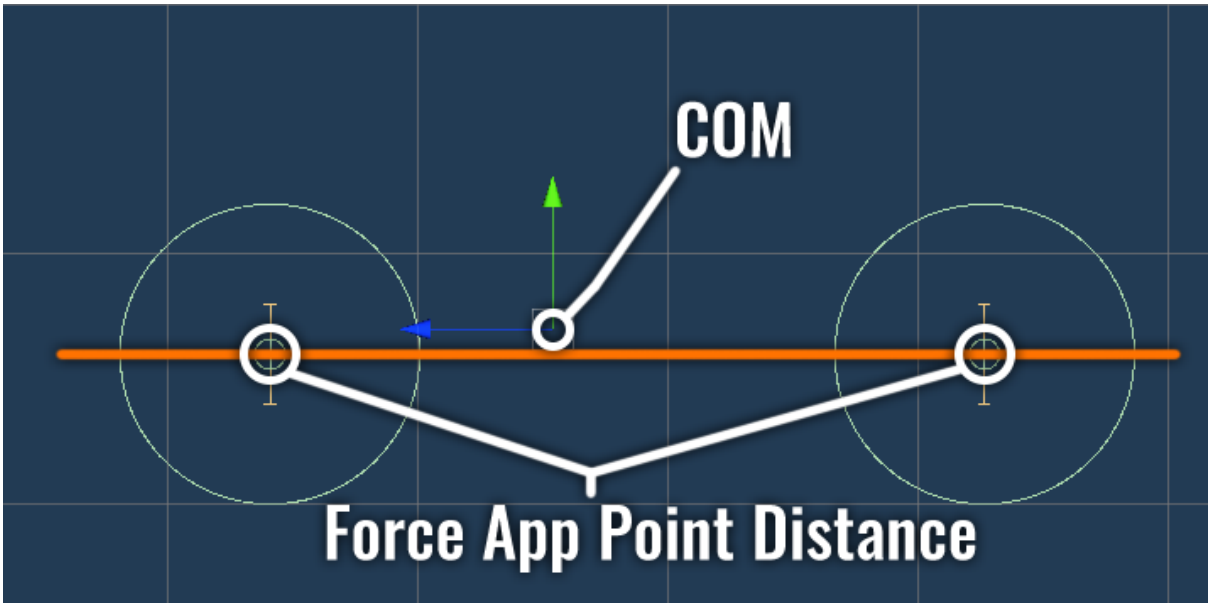


- ChangeUpGearTime – Shifting up delay
- ChangeDownGearTime – Shifting down delay
- AutomaticGearBox – Automatic gearbox
- MaxForwardSlipToBlockChangeGear – Max tire slip to not let gearbox shift up (for ex. If tires slip too much, gearbox will not shift up)
- GearsRatio – gear ratio for each gear (the number of array elements is the number of gears for the vehicle transmission). Fill this array with real data (ex. from Wikipedia)
- MainRatio – Main gear ratio
- ReversGearRatio – Reverse gear ratio

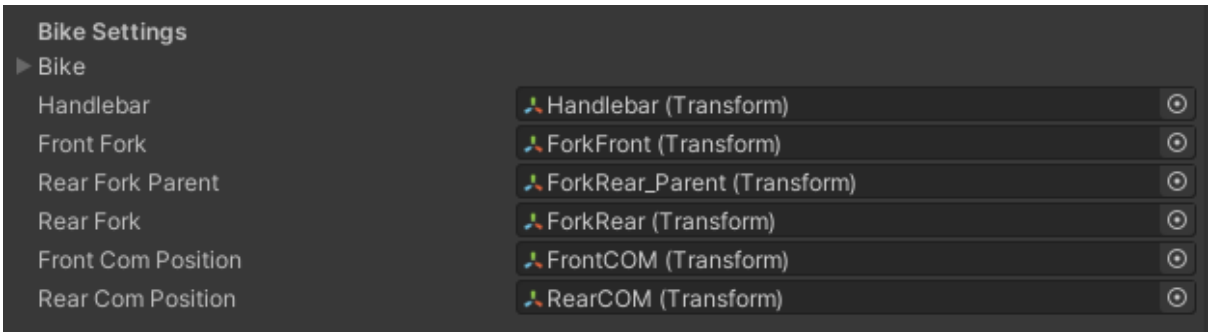
BikeController.cs.

BikeController is the inheritor of the CarController class, the class implements additional logic for maintaining balance on 2 wheels, crash logic, visual display of moving parts (steering wheel, front fork, rear fork).

!!!Attention!!! The COM position greatly affects the behavior of the bike. The closer the COM position is to the line between the ForceAppPointDistance of the wheels, the more stable the bike behaves when changing RB.Velocity (Acceleration, braking, turns).



Contains fields:

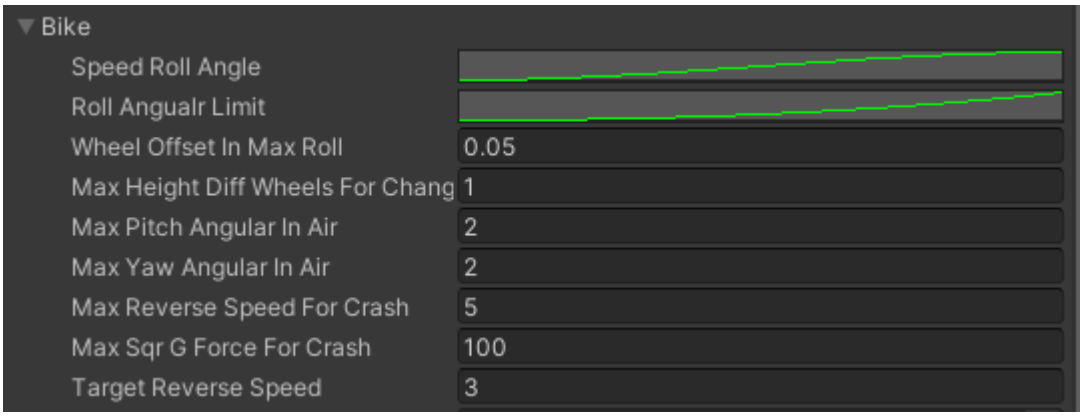


- Bike - Field of type BikeConfig (Detailed description further, in this section).
- Handlebar - Motorcycle handlebar, rotates with all child objects. The tilt of this object determines in which plane the steering wheel will turn (Only a visual change, it does not affect the physics of the bike), you can look at the setting of the handlebar in the CreateBikeWindow section.
- FrontFork - Front fork, it is important that the fork pivot is rotated in the same way as the handlebar pivot (Parallel) for correct display and animation of suspension.
- RearForkParent - The parent for the rear fork should be at the same point as the rear fork, but with Vector.zero local rotation (Needed to determine the point from which to track the rear wheel). Created automatically if it was not configured before PlayMode,.
- RearFork - Rear fork that follows the rear wheel to simulate the shock absorption of the bike's rear suspension.

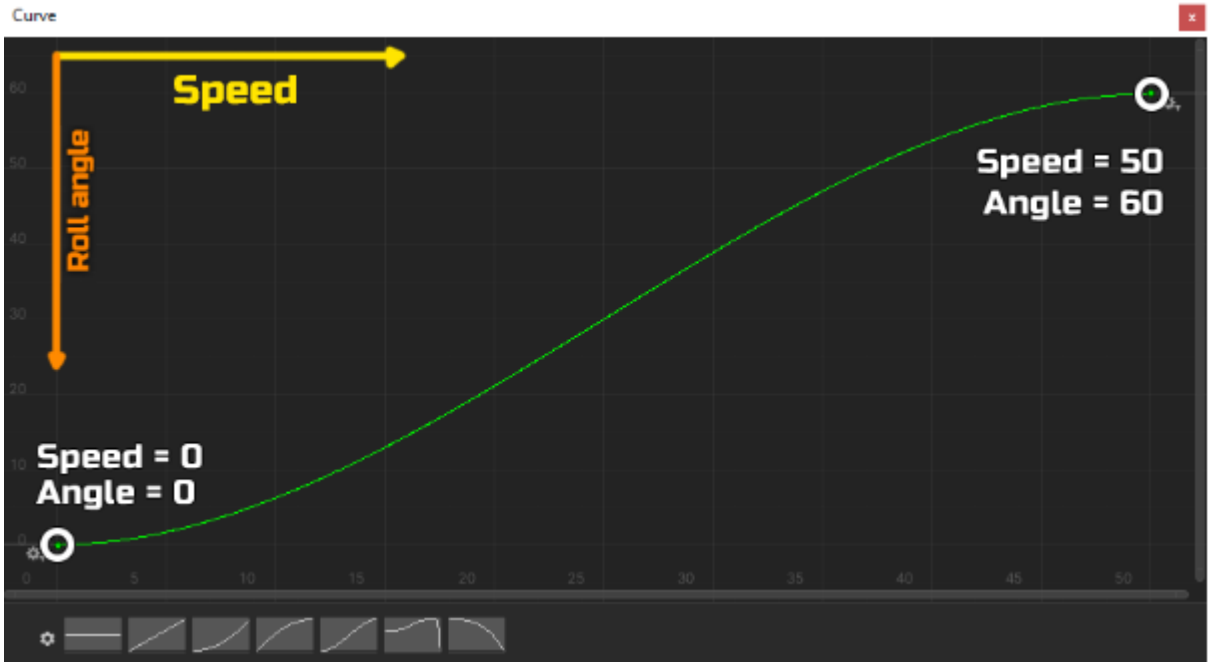
- FrontComPosition - The bike's COM shifts to this position (front wheel) with positive pitch (When the bike is on the ground).
- RearComPosition - The bike's COM shifts to this position (Rear Wheel) with negative pitch (When the bike is on the ground).

BikeConfig

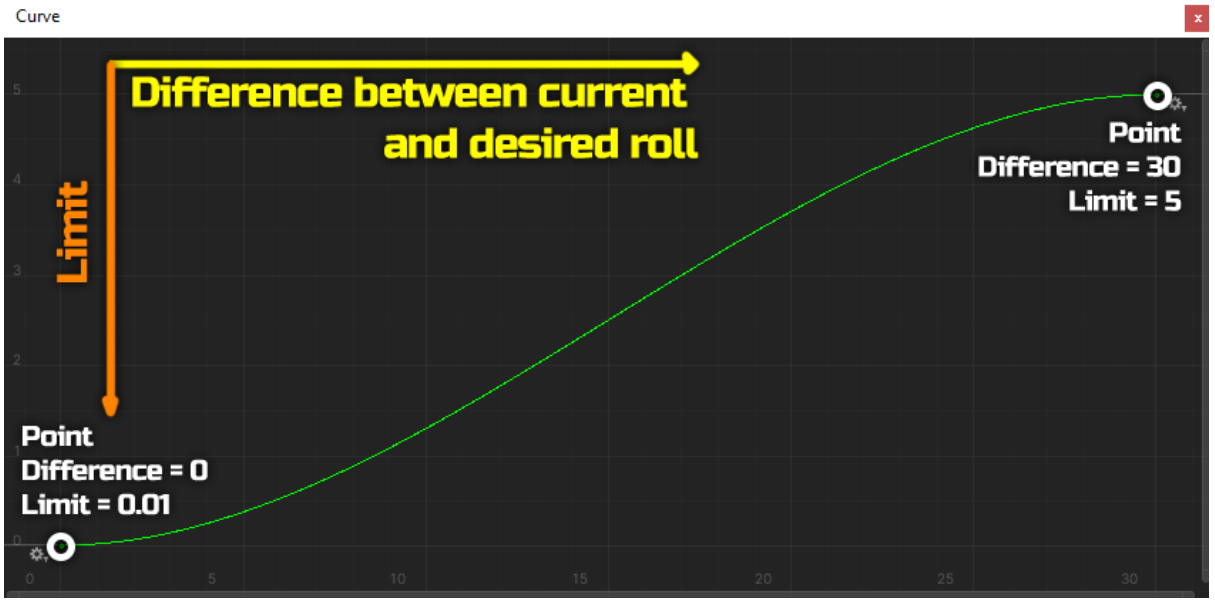
Contains fields:



- SpeedRollAngle - The roll curve of the bike depends on the current speed * ControlHorizontal. In the example, it is set: the higher the speed, the more the bike tilts.



- RollAngularLimit - Limiting the roll force. In the example, it is configured: The closer the current roll is to the desired roll, the stronger the limitation (In order to avoid rocking the bike).

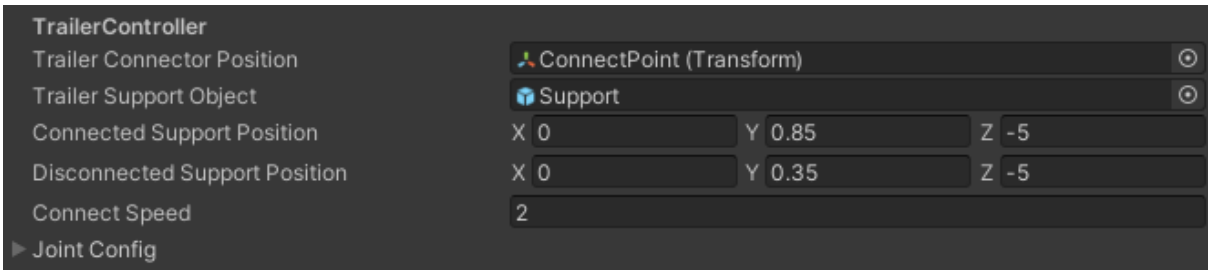


- WheelOffsetInMaxRoll - Offsets the WheelCollider in the direction the bike is leaning.
- MaxHeightDiffWheelsForChangeCom - Maximum difference in height between wheels for changing the Center of Mass. The closer the difference is to this value, the COM (Center of Mass) is closer to the standard position (For the balance effect of the bike, so that the bike does not roll back or forward). For example: The bike is going straight, the height difference == 0, when you press NigativePitch, COM is transferred to the RearComPosition as much as possible, the bike will start to rears up when accelerating, with an increase in the wheel height difference COM will gradually shift to the standard position.
- MaxPitchAngularInAir - The rate of change in the Pitch force in the air.
- MaxYawAngularInAir - The rate of change in the Yaw force in the air.
- MaxReverseSpeedForCrash - Reverse speed at which the bike will crash.
- MaxSqrGForceForCrash - Maximum gforce at which an accident occurs (For example, when hitting hard or landing).
- TargetReverseSpeed - Reverse speed.

TrailerController.cs

TrailerController inherits VehicleController, it contains all logics and physics for using the trailer, creating the connection, automatically removing the connection if the trailer gets destroyed, lights and brakes logic.

Contains fields:



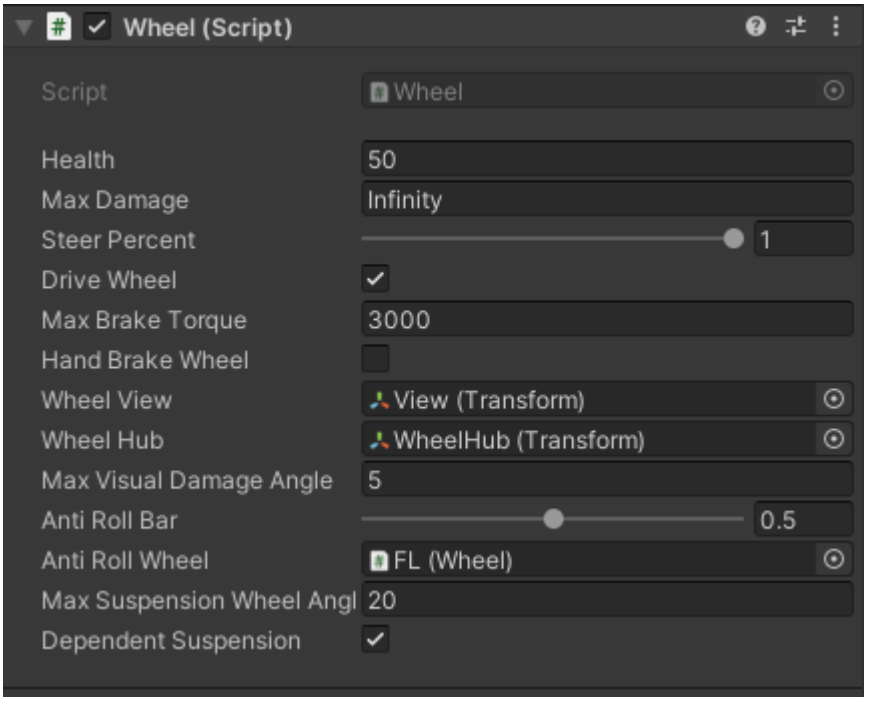
- TrailerConntectorPosition – point where trailer connects to the vehicle
- TrailerSupportObject – trailer's support props
- ConnectedSupportPosition – position of connected support props
- DisconnectedSupportPosition - position of disconnected support props
- ConnectSpeed – speed of changing the position of support props
- JointConfig – a set of parameters for creating ConfigurableJoint when attaching the trailer

Wheel.cs

This class inherits MoveableDO (to displace the wheel towards hit's direction).

Enclosure for standard WheelCollider, calculates current slip, temperature (in order to visualize tire smoke), information about contacting terrain.

Contains fields:



- Health – durability of the part
- MaxDamge – max damage dealt to part in one hit
- SteerPrecent – MaxSteerAngle percent of turning the wheel. Range from -1 to 1. 1 – max, 0 – wheel is not turning, -1 – negative max angle
- DriveWheel - if true engine torque will be transferred to this wheel. All the torque is distributed evenly between drive wheels: if there is one wheel it gets 100% of torque, if two one gets 50% as the another and so on
- MaxBrakeTorque – max brake force of a wheel
- HandBrakeWheel – if true brake from will be applied to this wheel
- WheelView – wheel object that takes position and rotation of the wheel, if you need to adjust the wheel's offset you need to change the position of nested in View objects along X axis.
- WheelHub– The object to which the wheel is attached, takes position and rotation of the wheel (Rotates only on the Y axis), this object is only needed for visual effect, such as a brake pad.
- MaxVisualDamageAngle – all nested in View elements randomly rotate to this value. Depends on how hard the damage was dealt. This is only a visual effect as it heavily affects the vehicle. After a small damage the vehicle starts moving not straight so there is no technical damage right now.
- AntiRollBar - Imitation of the anti-roll bar, works only with AntiRollWheel (Opposite wheel on the axis), if the AntiRollWheel is lower, then an upward force is applied at the point of the wheel, due to which the car resists a rollover. takes values from 0 to 1, depending on the mass of the car (The greater the mass of the car, the more force).
- AntiRollWheel - Opposite wheel on the axis.

- MaxSuspensionWheelAngle - Changing the wheel angle when the suspension is working, for visual effect only.
- DependentSuspension - The angle of the wheel will depend on the opposite wheel, only works with AntiRollWheel, for visual effect only.

BikeWheel.cs

BikeWheel is a child of the Wheel class, it is only needed for the correct visual display of the wheels. BikeWheel does not change the position of the WheelView, the animation of the suspension of the bike is implemented in the BikeController.

WheelCollider

In order the Wheel to work there must be a WheelCollider attached to the object. WheelCollider has a heavy impact on the vehicle's behavior. Many people just do not know how to properly configure WheelCollider, if you have issues with it try to carefully read the documentation WheelCollider I think the majority have problems with **Forward/Sideways Friction**

Forward Friction – friction along the wheel's direction. Occurs during acceleration (transferring of torque to the wheels)/braking

Sideways Friction – friction perpendicular to the wheel's direction. Occurs during vehicle sliding/drifting

Next I provide you some examples of Forward/Sideways Friction. Changing these values, you can achieve the behavior you want. Important: mass of the vehicle and wheel affects on these parameters (for ex. if the vehicle is too light, these parameters will not act as they are called)

Small friction (suitable for Sideways for drift):

Extremum Slip - 0.3

Extremum Value - 0.65

Asymptote Slip - 0.6

Asymptote Value - 0.2

Medium friction:

Extremum Slip - 0.4

Extremum Value - 1

Asymptote Slip - 0.8

Asymptote Value - 0.5

High friction:

Extremum Slip - 0.4

Extremum Value - 2

Asymptote Slip - 0.8

Asymptote Value - 1.2

Very high friction:

Extremum Slip - 0.4

Extremum Value - 4

Asymptote Slip - 0.8

Asymptote Value - 2

Enormous friction:

Extremum Slip - 0.4

Extremum Value - 10

Asymptote Slip - 0.4

Asymptote Value - 10

Stiffness – multiplier for Extremum and Asymptote value (default is 1). Changes the strength of the friction. Value changes depending on the ground material (see GroundDetection)

Artificial Intelligence (AI).

There are three types of AI in the asset: Race, Drift, and Pursuit. For each type of AI, there is an asset with settings. It is important that different configurations of cars (different speed, controllability, etc.) may require different AI settings.

AIPath.

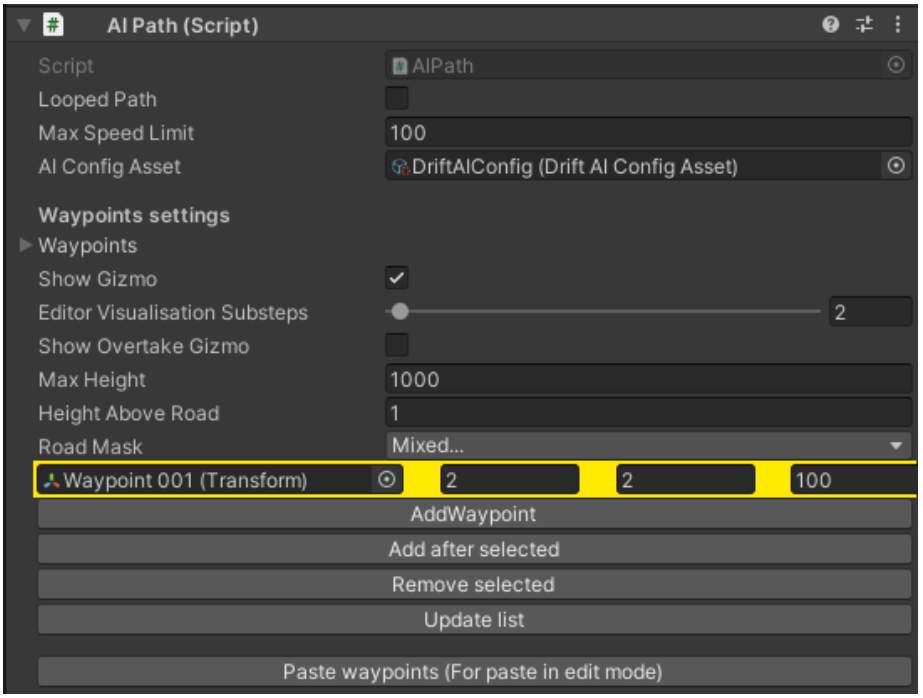
This class is needed to create paths that the AI will follow in modes such as racing and drift.

The class consists of 2 parts:

AIPath.cs - it contains all the data and logic for determining route points.

AIPathEditor.cs - it contains the logic for editing data and displaying the Gizmo, works only in the editor, does not affect the game logic.

The class contains fields:



- LoopedPath - Loop path, on - circle, off - segment from point A to B.
- MaxSpeedLimit - Maximum speed limit.
- AIConfigAsset - Asset with AI configuration, all AIs without a reference to the asset will refer to the asset from this field.
- Waypoints - List of WaypointData points.
- ShowGizmo - Displaying Gizmo.
- EditorVisualisationSubsteps - The size of the segments that make up the Gizmo of the entire path, the parameter only for visualizing the path in the editor window, does not affect the logic.
- ShowOvertakeGizmo - Shows the Gizmo overtaking line.
- MaxHeight - Maximum height of waypoints (From this height, the ray will be cast down to find the desired height).
- HeightAboveRoad - The height above the surface that the ray hits when updating waypoints.

- RoadMask - The mask with which the ray interacts.
- SelectedWaypoint - Additional field for editing the selected point.

Buttons:

- AddWaypoint - Adds a point to the end of the list, with the parameters of the last point.
- AddAfterSelected - Adds a point after the selected point, with the parameters of the selected point.
- RemoveSelected - Deletes the selected point, highlights the next one after the deleted one.
- UpdateList - Updates the height of all points, renames all points in order in the list.
- CopyWaypoints - Copies all point settings n Playmode (To be able to edit the path in PlayMode).
- PasteWaypoints - Pastes copied points from Playmode to EditMode.

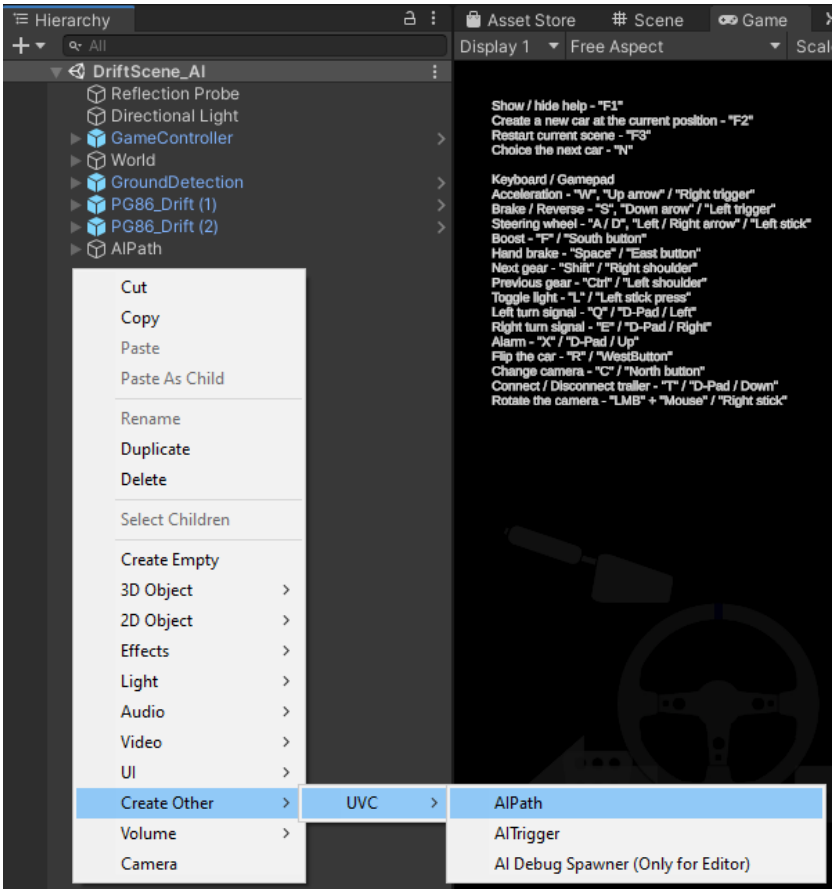
WaypointData - Waypoint data, all fields are in one line for ease of editing and normal data perception, especially in lists. The data has tooltips on hover.

Contains fields:

- Point - Transform the point to determine the position.
- OvertakeZoneLeft - The size of the overtaking area on the left along the road.
- OvertakeZoneRight -The size of the overtaking area on the right along the road.
- SpeedLimit - Speed limit on this section of the road.

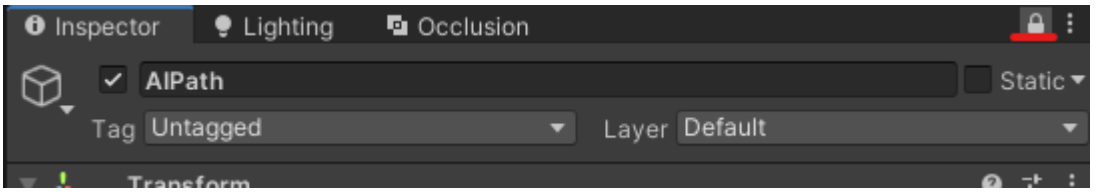
Creating AIPath:

To create an object with the AIPath component, you need to: In the Hierarchy window, right-click and select the menu item "Create Other/UVC/AIPath".



Editing AIPath:

For the convenience of editing, you need to lock the Inspector window with the selected AIPath object, so that the focus does not shift from this object.



After that you can create a new point (if it is a new AIPath, without created points), the first point is created at the zero point, all subsequent points are created at the position of the previous point and with the parameters of the previous point.

You can also use the keyboard shortcut:

- Shift + LMB, in this case a point will be created at the cursor position (If there is a collider).
- Ctrl + LMB, in this case the selected point will move to the cursor position (If there is a collider).

Orthographic top view can be set for editing (Middle mouse click Y)

After creating the first point, you need to place it where the beginning of the path is on your track. Then, by clicking on the AddWaypoint button, add the required number of points with the required parameters.

To edit an already created point, you can hover over it and click any mouse button, after that the point is selected and you can move it or change any parameters.

AITrigger.

A trigger affecting AI behavior. Now with the help of the trigger, you can only configure the zones of use of the boost for the AI, with the possibility of choosing the probability of using the boost.

To create an object with the AITrigger component, you need to: In the Hierarchy window, right-click and select the menu item "Create Other/UVC/AITrigger".

It has only the BoostProbability field: 0 - the boost will not be enabled, 1 - the boost will always be enabled.

When the trigger is exited, if the boost was enabled, it stops.

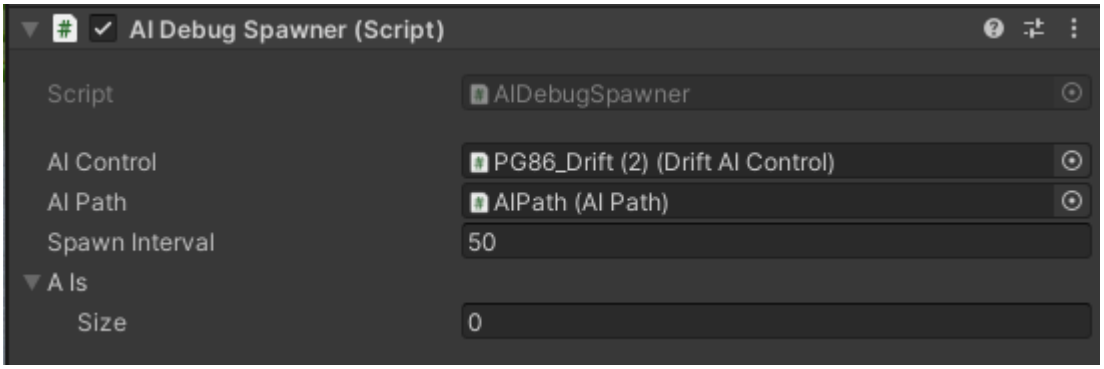
If you wish, you can add any other logic to AITrigger.cs.

AISpawner.

Needed for quick testing and path editing.

To create an object with the AISpawner component, you need to: In the Hierarchy window, right-click and select the menu item "Create Other/UVC/AISpawner".

Contains fields:



AIControl - A car that will be duplicated along the path.

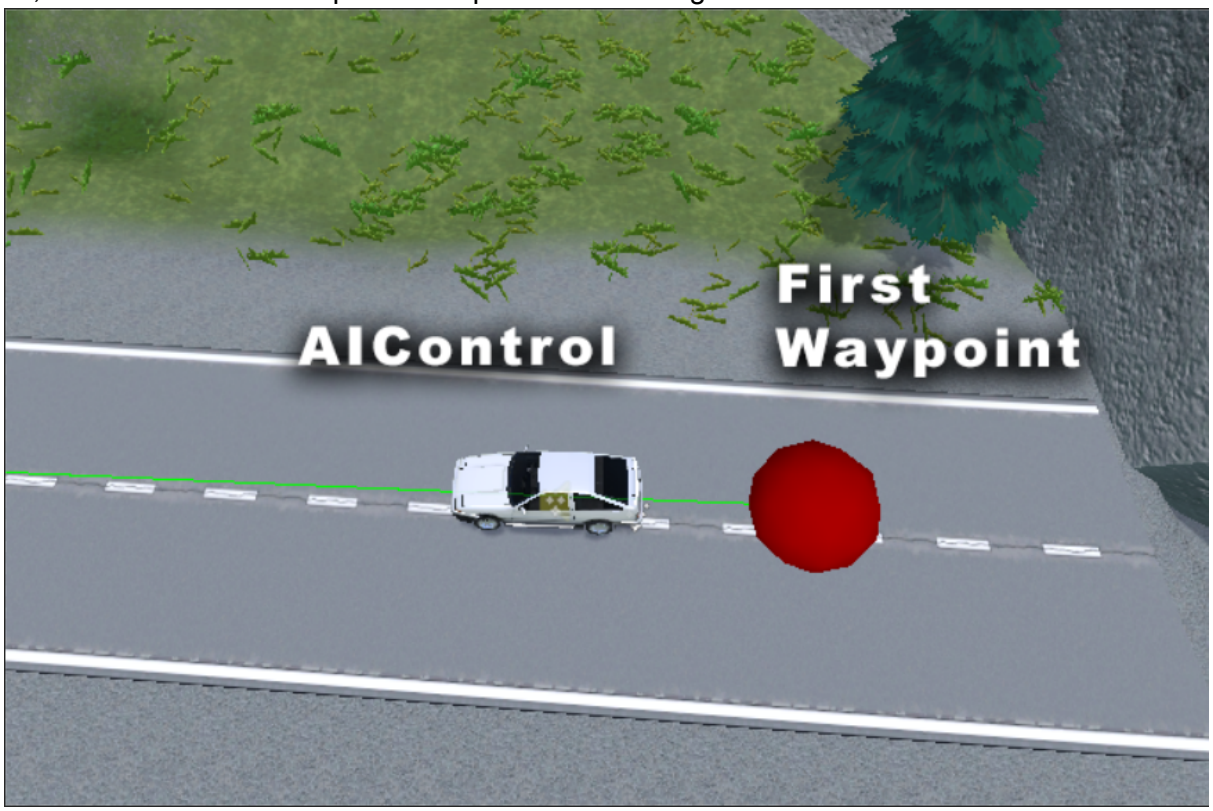
AIPath - The path along which the car will be duplicated.

SpawnInterval - Distance between duplicate cars.

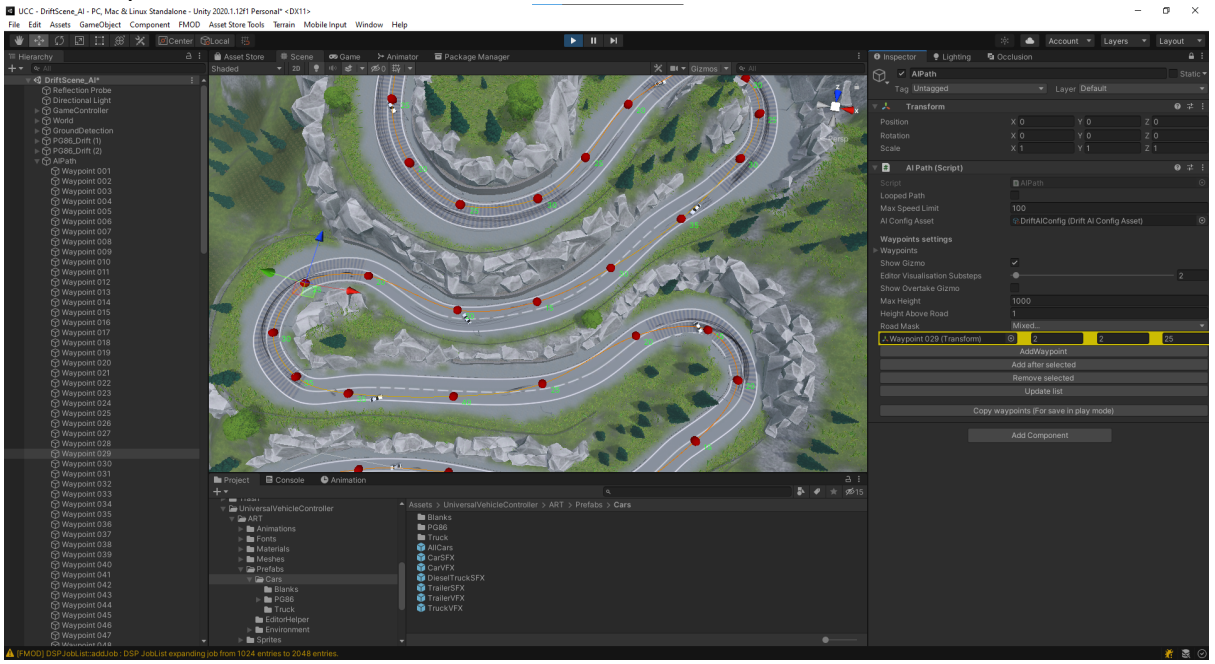
AIs - List with duplicate cars (Cars are added there when creating duplicates).

Usage example:

1. Create an AISpawner, specify a link to the desired car and the desired path (The car must be moved to a point near the beginning of the path, so that it is slightly after the first point in the direction of the path). Also specify SpawnInterval, the larger the interval, the fewer duplicates will be, also the number of duplicates depends on the length of the track.



2. Launch Playmode, switch to AIPath, lock the Inspector and start editing the path (You can change the position of points, speed limits and the size of the overtaking zones), for all changes, all AIs will change their behavior, this is much more convenient and faster than launching Playmode after each edit paths.



3. After completing editing, it is important DO NOT EXIT PLAYMODE you need to copy all changes to the Clipboard by clicking on the "CopyWaypoints" button, only after that you can exit Playmode.

4. In Editmode you can click on the "PasteWaypoints" button, after which the changes copied from Playmode will be pasted.

5. You can remove or hide AISpawner.

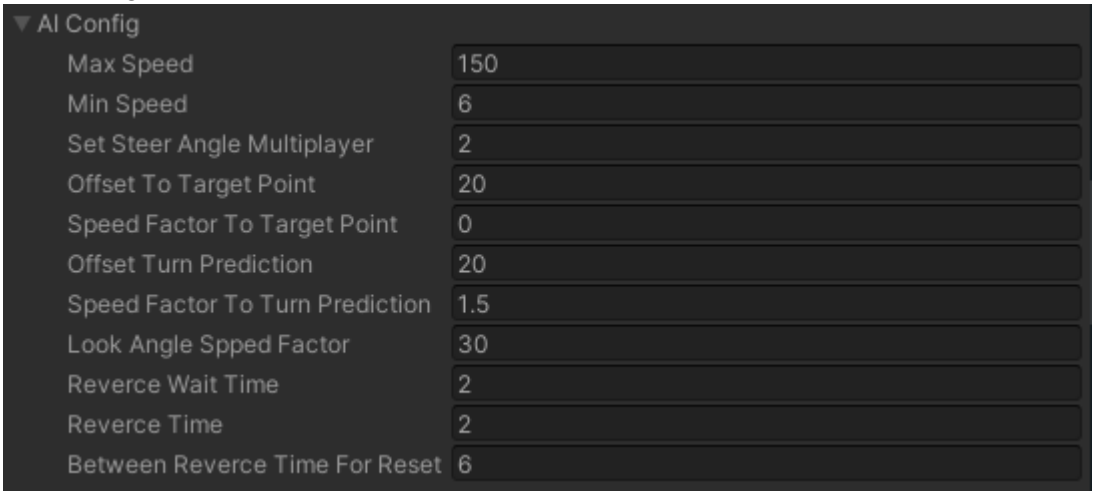
BaseAIControl.

The base class for AI, all AI are inheritors of this class, this class implements the ICarControl interface, it contains a link to AIConfigAsset and the logic for entering and exiting AITRigger.

BaseAIConfigAsset

Asset with AI settings of type BaseAIConfig, these settings apply to all AI types in the UVC asset.

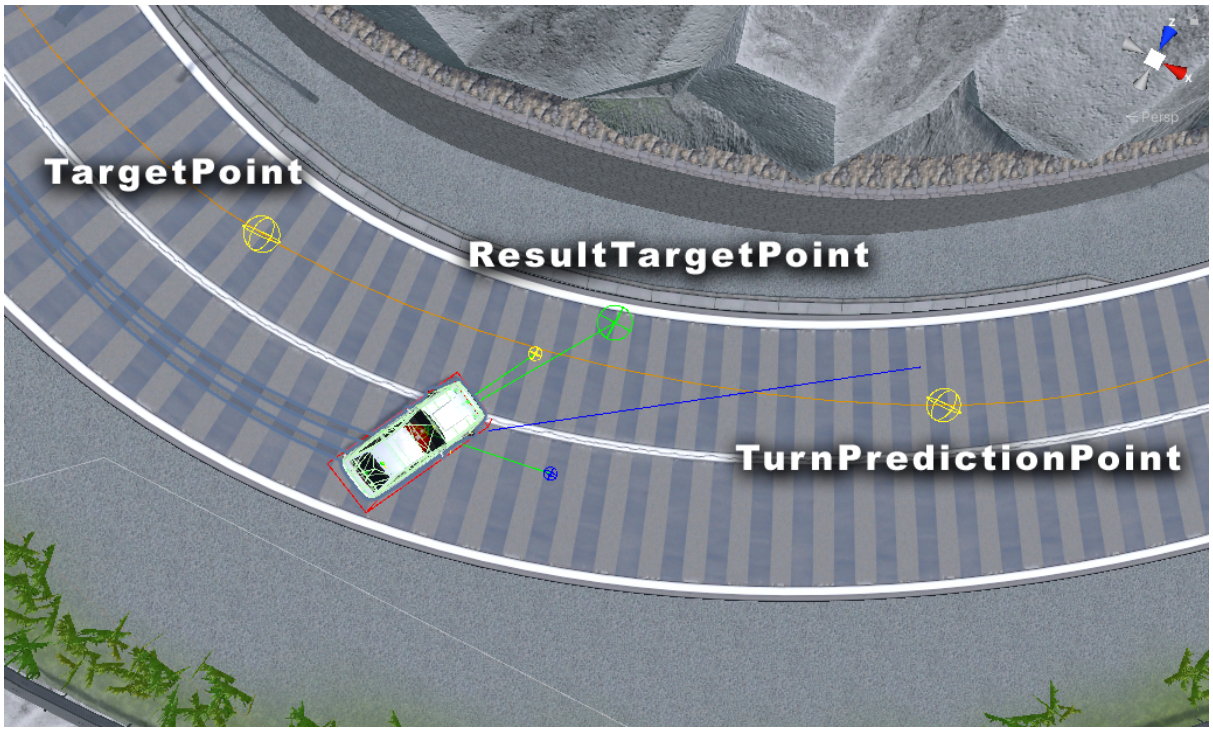
BaseAIConfig contains fields:



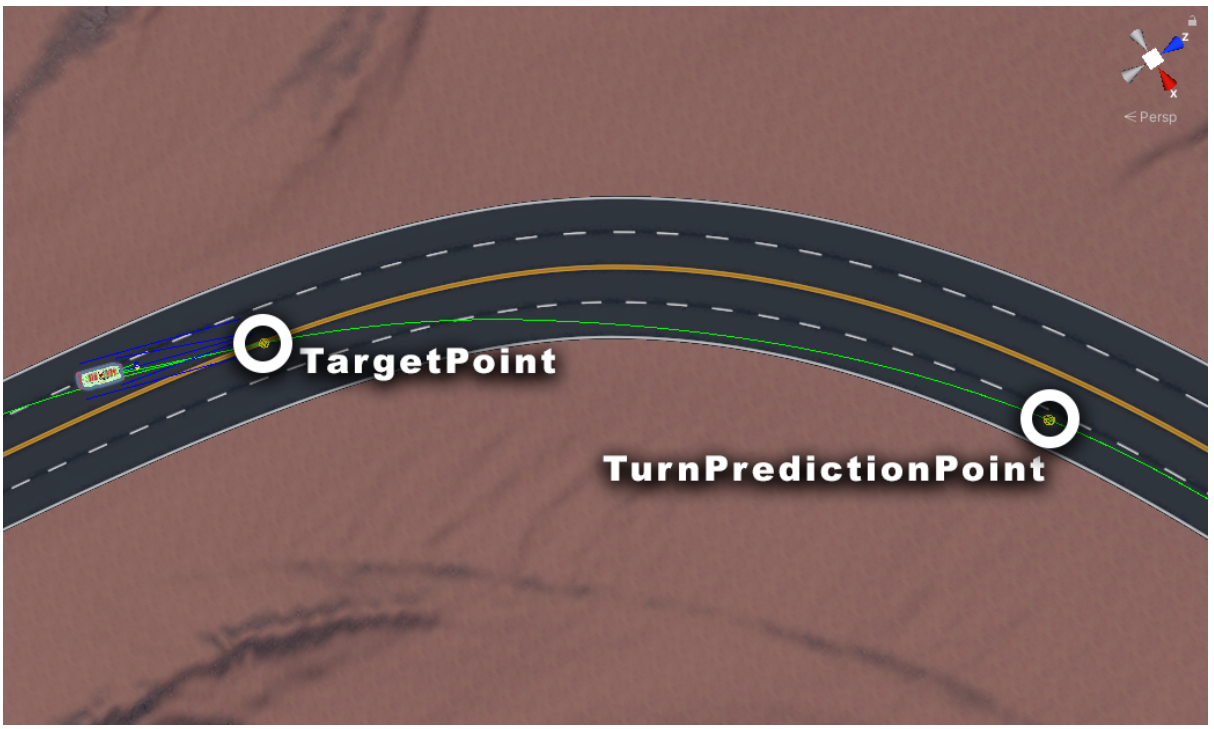
- MaxSpeed - The maximum speed (unit per second) that the AI will attempt to achieve.
- MinSpeed - The minimum speed (unit per second) that the AI will attempt to achieve.
- SetSteerAngleMultiplayer - multiplier of the steering wheel to TargetPoint, the higher this parameter, the faster the AI will turn to the desired position (The optimal value is from 1 to 4, depending on the speed and type of AI).
- OffsetToTargetPoint - TargetPoint constant offset. below are examples for all types of AI.
    - For RaceAIControl and DriftAIControl, offset from the current position on the path.
    - For PursuitAIControl, displacement from the position of the pursued vehicle towards the movement (Rigidbody.velocity) of the pursued vehicle.
- SpeedFactorToTargetPoint - Dynamic offset TargetPoint. below are examples for all types of AI.
    - For RaceAIControl and DriftAIControl, the additional offset depends on the speed of the car.
    - For PursuitAIControl, the additional offset depends on the speed of the vehicle being pursued.
- OffsetTurnPrediction - Point offset for cornering speed control, for different AIs is applied in the same way as OffsetToTargetPoint. below are examples for all types of AI.
- SpeedFactorToTurnPrediction - Dynamic Offset TurnPredictionPoint, for different AIs, is applied in the same way as SpeedFactorToTargetPoint. below are examples for all types of AI.
- LookAngleSppedFactor - The maximum angle between the direction of the AI car and a point (Depends on the type of AI), the larger the angle of this parameter, the more the AI car will slow down.
    - For DriftAIControl, the angle between the direction of the AI car and the calculated AI target point.
    - For RaceAIControl and PursuitAIControl, the angle between the direction of the AI car and TurnPredictionPoint.
- ReverceWaitTime - Waiting time before activating reverse gear, if the AI car is stuck or ran into an obstacle, then after this time, reverse gear will engage and the wheels turn in the opposite direction.
- ReverceTime - The time at which the reverse gear is activated.
- BetweenReverceTimeForReset - If during this time the reverse gear is switched on again, the car will be reset (The reset logic must be done based on the requirements of your game, now the reset occurs in the same position).

OffsetToTargetPoint and OffsetTurnPrediction settings for different types of AI.

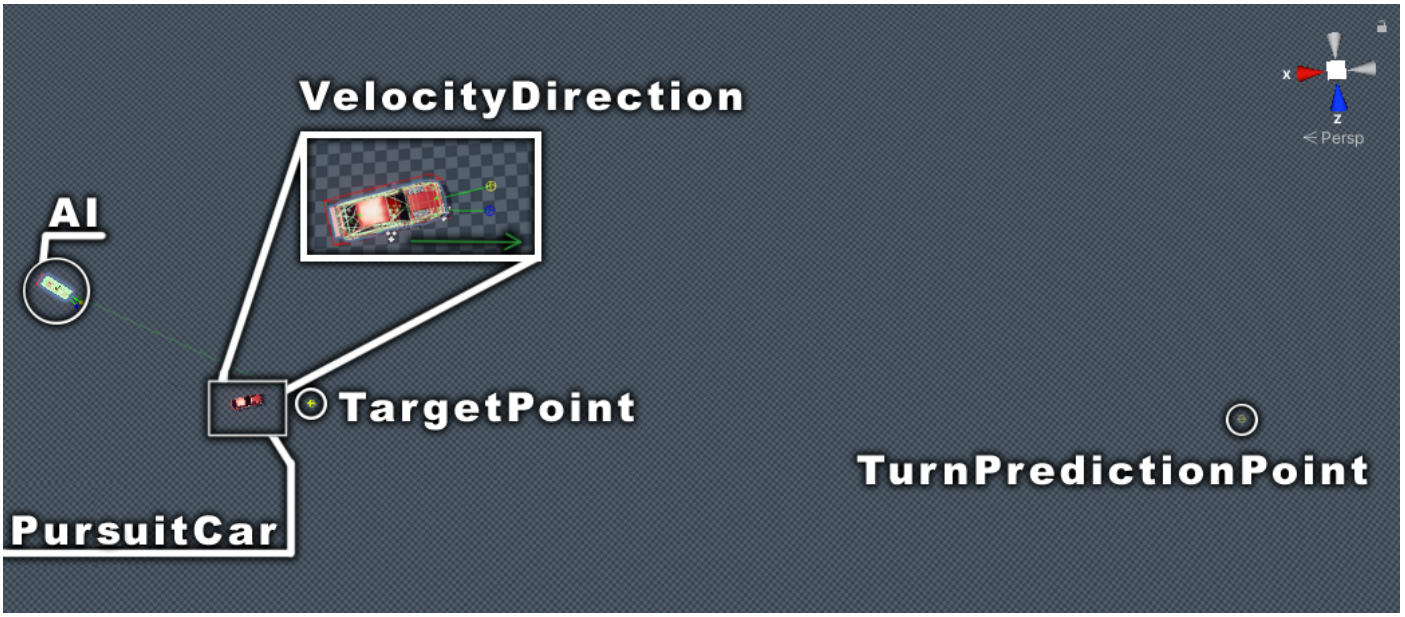
For DriftAIControl, the settings are in the DriftAIConfig asset. The point to which the car follows is calculated dynamically by the formula: (TargetPoint + TurnPredictionPoint) * 0.5f, ResultTargetPoint is located exactly between the TargetPoint and TurnPredictionPoint points, to be able to direct the AI car towards the inside of the turn.



For RaceAIControl, the settings are in the RaceAIConfig asset for the RaceScene scene and in the RaceAIConfig_ForRing asset for the MobileScene scene. The point to which the car follows is TargetPoint.



For PursuitAIControl, the settings are in the PursuitAIConfig asset. The point to which the car follows is TargetPoint.



RaceAIControl.

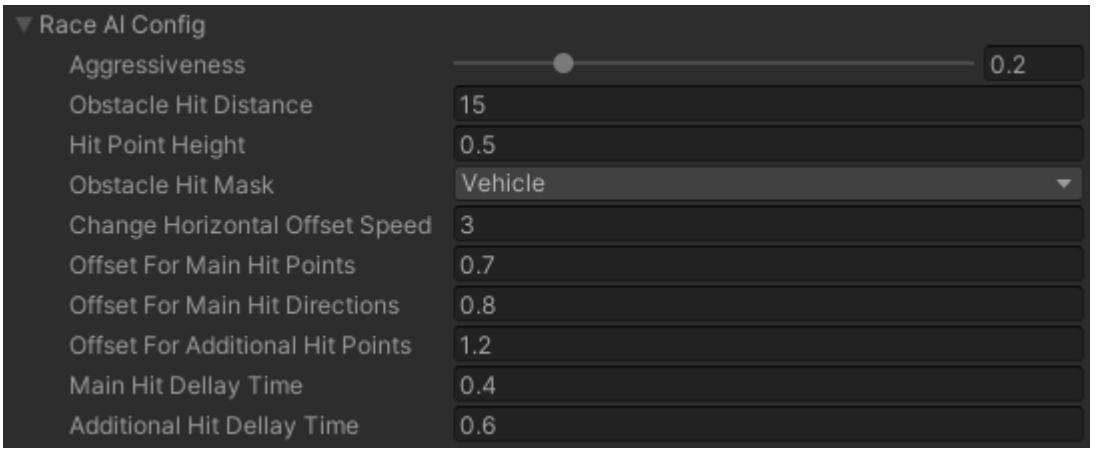
An AI class for high-speed races that follows the path has the logic of overtaking rivals. The asset has all the tracks with long and smooth turns, if your game has sharper turns and the AI will not cope well with them, then speed limiting with AIPAth can help you.

The algorithm for calculating tracking points, see "OffsetToTargetPoint and OffsetTurnPrediction settings for different types of AI".

RaceAIConfigAsset

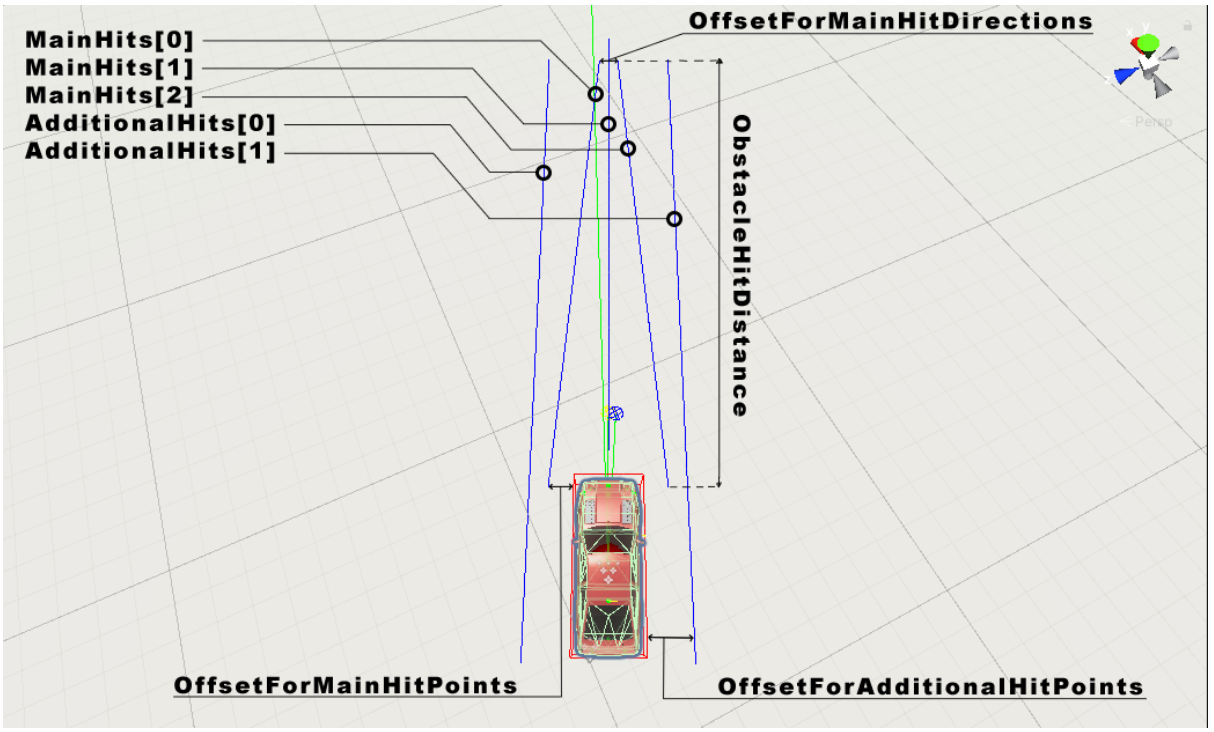
The inheritor of BaseAIConfigAsset, contains the RaceAIConfig class, needed for the RaceAiControl configuration.

RaceAIConfig contains fields:



- Aggressiveness - AI aggressiveness, 0 - AI will slow down in front of the car, 1 - AI will not slow down, will ram the car in front.
- ObstacleHitDistance - Distance for checking obstacles.
- HitPointHeight - Raycast height for checking obstacles.

- ObstacleHitMask - Obstacle mask, by default contains only layer 8, your layer numbers may differ.
- ChangeHorizontalOffsetSpeed - The speed of change of the horizontal offset, for overtaking rivals.
- OffsetForMainHitPoints - Offset of the position of the main rays relative to the side borders of the car, 0 - the position of the rays will be on the border of the car.
- OffsetForMainHitDirections - Offset of the direction of the main rays, 0 - rays will be parallel to the direction of the car, 1 - rays will be directed towards the center of the car.
- OffsetForAdditionalHitPoints - Offset of the position of additional rays relative to the side borders of the car, 0 - the position of the rays will be on the border of the car.
- MainHitDellayTime - The interval for using the main rays for optimization.
- AdditionalHitDellayTime - The interval for using additional rays for optimization.



Overtaking work algorithm.

If all rays are free of obstacles, then CurrentHorizontalOffset tends to 0 at a speed (ChangeHorizontalOffsetSpeed * 0.2f).

If the main ray has an obstacle, then CurrentHorizontalOffset trends in the opposite direction to the overtaking border with a ChangeHorizontalOffsetSpeed * speed (1 - percentage to the obstacle from the ray length), the closer the obstacle, the faster the offset changes.

If MainHits [0] (left ray) has an obstacle, then the offset occurs to the right side.

If MainHits [1] (right ray) has an obstacle, then the offset occurs to the left side.

If both rays have obstacles, then the logic is turned on for the ray with a smaller distance.

If only MainHits [2] (center ray) has an obstacle, then the offset continues to tend towards the side to which there is already offset.

If the additional ray is obstructed and the main rays are not obstructed, then CurrentHorizontalOffset remains in place.

DriftAIControl

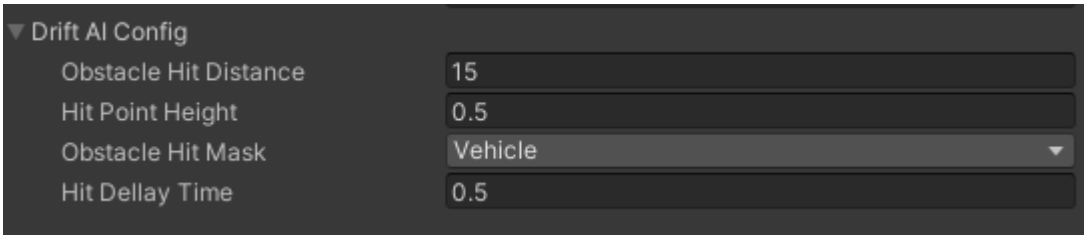
The AI class for drifting, follows the path, is very sensitive to settings and to speed, in the asset all the paths for DriftaAIControl have fine tuning for speed limiting, due to the inability to determine minor turns at high speeds, the limitation also gives more control over the AI and its predictability behavior. DriftAIControl has one raycast to the side between the direction of the car and the force of its movement, to detect the car in front and to brake if necessary.

The algorithm for calculating tracking points, see "OffsetToTargetPoint and OffsetTurnPrediction settings for different types of AI".

DriftAIConfigAsset

The inheritor of BaseAIConfigAsset, contains the DriftAIConfig class, needed for the DriftAiControl configuration.

DriftAIConfig contains fields:



- ObstacleHitDistance - Distance for checking obstacles.
- HitPointHeight - Raycast height for checking obstacles.
- ObstacleHitMask - Obstacle mask, by default contains only layer 8, your layer numbers may differ.
- HitDellayTime - Raycast usage interval, for optimization.

PursuitAIControl

AI class for pursuing a car. The class has an additional field "TargetRB" - this is a link to the pursued car, if you wish, you can add logic for selecting the pursued car. This is a primitive persecution, without any trickery.

Now the AI can be at rest until the pursued car comes into view, after the car comes into view, the AI starts to pursue the car (The pursuit does not turn off, but you can add shutdown logic).

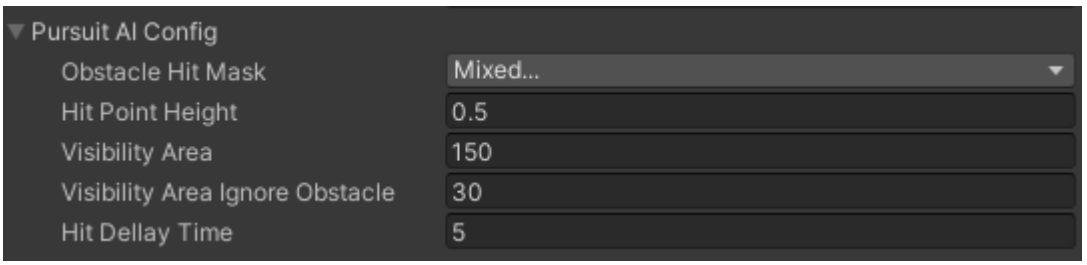
Now AI is pursuing the car without taking into account the environment, with sharp turns (for example, city streets), it can run into the collider or fall into the abyss. You can improve the pursuit logic to your liking or combine it with other types of AI. Unfortunately for all types of games, I cannot create a universal pursuit AI, it is highly dependent on the level design and the type of game.

For the algorithm for calculating tracking points, see "OffsetToTargetPoint and OffsetTurnPrediction settings for different types of AI".

PursuitAIConfigAsset

Inherited from BaseAIConfigAsset, contains the PursuitAIConfig class, needed for the PursuitAiControl configuration.

PursuitAIConfig contains fields:



- ObstacleHitMask - Obstacle mask, by default contains only layer 8, your layer numbers may differ.
- HitPointHeight - Raycast height for checking obstacles.
- VisibilityArea - Visibility zone, 180 degrees in front of the car, obstacles are not ignored. If the pursued car approaches from the front and the pursued vehicle is not obscured by an obstacle, the AI will begin pursuit.
- VisibilityAreaIgnoreObstacle - Visibility area around the car, obstacles are ignored. If the pursued car approaches this distance from either side, the AI will begin pursuit.
- HitDellayTime - Raycast usage interval, for optimization.

Lights, glass

Main component handling logic for all the lights CarLighting. Requires CarController to be attached.

Contains only one field:

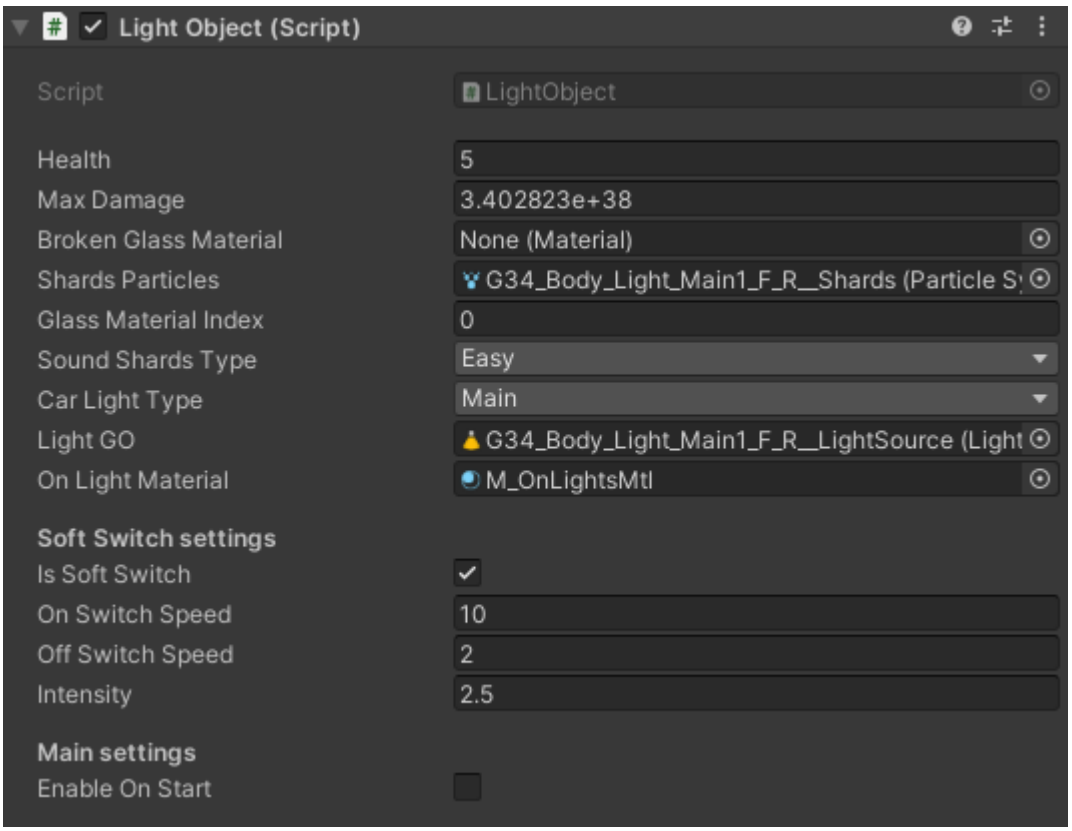
TurnsSwitchHalfRepeatTime – half-time cycle of the blinkers (used for all blinkers and warning lights)

All subscriptions to events (reverse gear, braking, lights on/off), control of nested lights handles inside this (CarLighting) class.

LightObject

This is a config object – here you can define the type of the lights and other parameters. LightObject inherits GlassDO (GlassDO is used for glass and has the same logic for destruction). The number of lights with the same type is unlimited.

Contains fields:



Fields from GlassDO:

- BrokenGlassMaterial – broken material if you need to show shards after the destruction. If this field is null, then after destruction the object will disappear
- ShardsParticles – particle system instantiated during destruction
- GlassMaterialIndex – material index in case Mesh has more than one SubMesh
- SoundShardsType - sound played when an object is destroyed.

Fields from LightObject:

- CarLightType – on light type depends the moment when the light turns on
- LightGO – light object. Just turns on/off when switched
- OnLightMaterial – material applied to Renderer (if there is such attached to the object) Emission flag should be checked. The same material can be applied to all the lights (like in this asset) as MaterialPropertyBlock is used.
- IsSoftSwitch – flag for smooth switching of the lights to imitate incandescent lamps (Gradual turning on and off). If flag is not checked switching of the lights will be during one frame just replacing OnLightMaterial with default one (default material is assigned in Renderer.materials)
- OnSwitchSpeed – Switch on speed when IsSoftSwitch is checked.
- OffSwitchSpeed – Switch off speed when IsSoftSwitch is checked.
- Intensity – Intensity of the lights when IsSoftSwitch is checked
- EnableOnStart – check if you need the lights turned on at the beginning (for ex. race during the night)

For ease of use there are methods to create glass and light shards called CreateGlassShards and CreateLightShards which you can call from the component's contextual menu. After creating the shards proper direction and size need to be selected if needed.

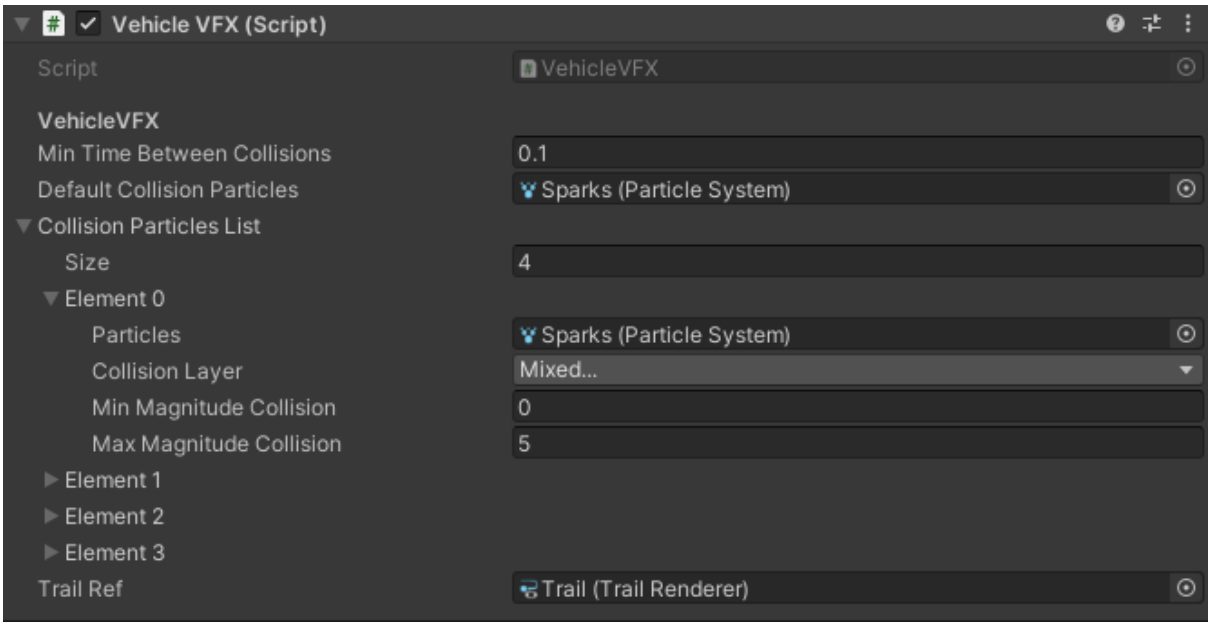
Visual effects

VehicleVFX

Base component for handling the VehicleVFX logic. Must be a child of VehicleController.

VehicleVFX creates tire smoke (depending on the ground the wheel is touching), tire skid marks, sparks and smoke for collisions.

Contains fields:

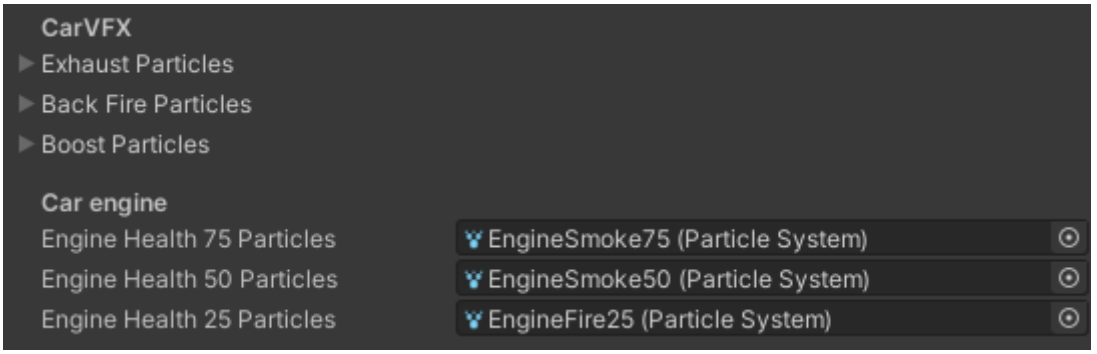


- MinTimeBetweenCollisions – if during this time a collision happens particles will not be emitted
- DefaultCollisionParticles – default collision particles system if there is no ParticleSystem from CollisionParticles
- CollisionParticlesList – list of type CollisionParticles, when collision happens proper ParticleSystem is being applied from this list. CollisionParticlesList contains fields:
    - Particles – reference to the Particle System
    - CollisionLayer – collision system is selected if collided object is from this LayerMask
    - MinMagnitudeCollision – collision system is selected if collision magnitude is above this value
    - MaxMagnitudeCollision – collision system is selected if collision magnitude is below this value
- TrailRef – when tire is sliding, copies of Trail object are created on the contact area
- Dust/Smoke under the wheels emits GroundConfig.SlipParticles or GroundConfig.IdleParticles, depending on the current wheel's slide. Current GroundConfig is stored in Wheel class.

CarVFX

CarVFX inherits VehicleVFX, must be child of CarController. Used to emit car effects like smoke/fire from exhaust.

Contains fields:

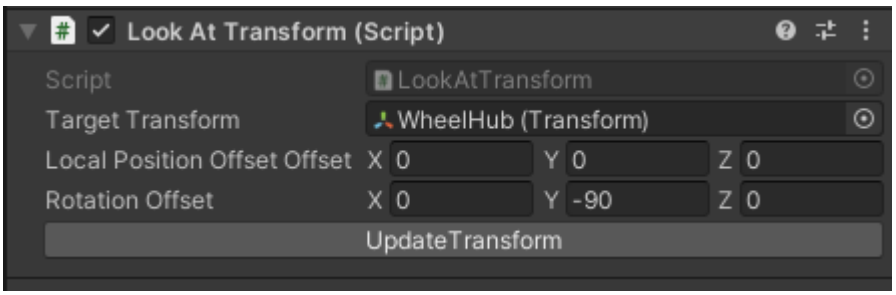


- ExhaustParticles - Particle systems emitted during engine load (Suitable for diesel trucks).
- BackFireParticles - Flashes of fire from the exhaust pipe, during cutoff or gear changes.
- BoostParticles - Fire from the exhaust pipe while using the boost.
- EngineHealth75Particles - The particle system is activated when the engine is at 75% health.
- EngineHealth50Particles - The particle system is activated when the engine is at 50% health.
- EngineHealth25Particles - CThe particle system is activated when the engine is at 25% health. At engine health <25%, the EngineHealth50Particles particle system is also active.

LookAtTransform

LookAtTransform is a universal component, used in the asset to visually simulate the suspension, for example, motorcycle shock absorbers or GMS_4x4 suspensions.

Contains fields:



- TargetTransform - Transform which the object will follow.
- LocalPositionOffset - Offset looking position (If you need to follow not the center of the transform).
- RotationOffset - Additional rotation, can be used to adjust the direction.
- Button UpdateTransform - Applies rotation (To be able to configure the component without starting PlayMode).

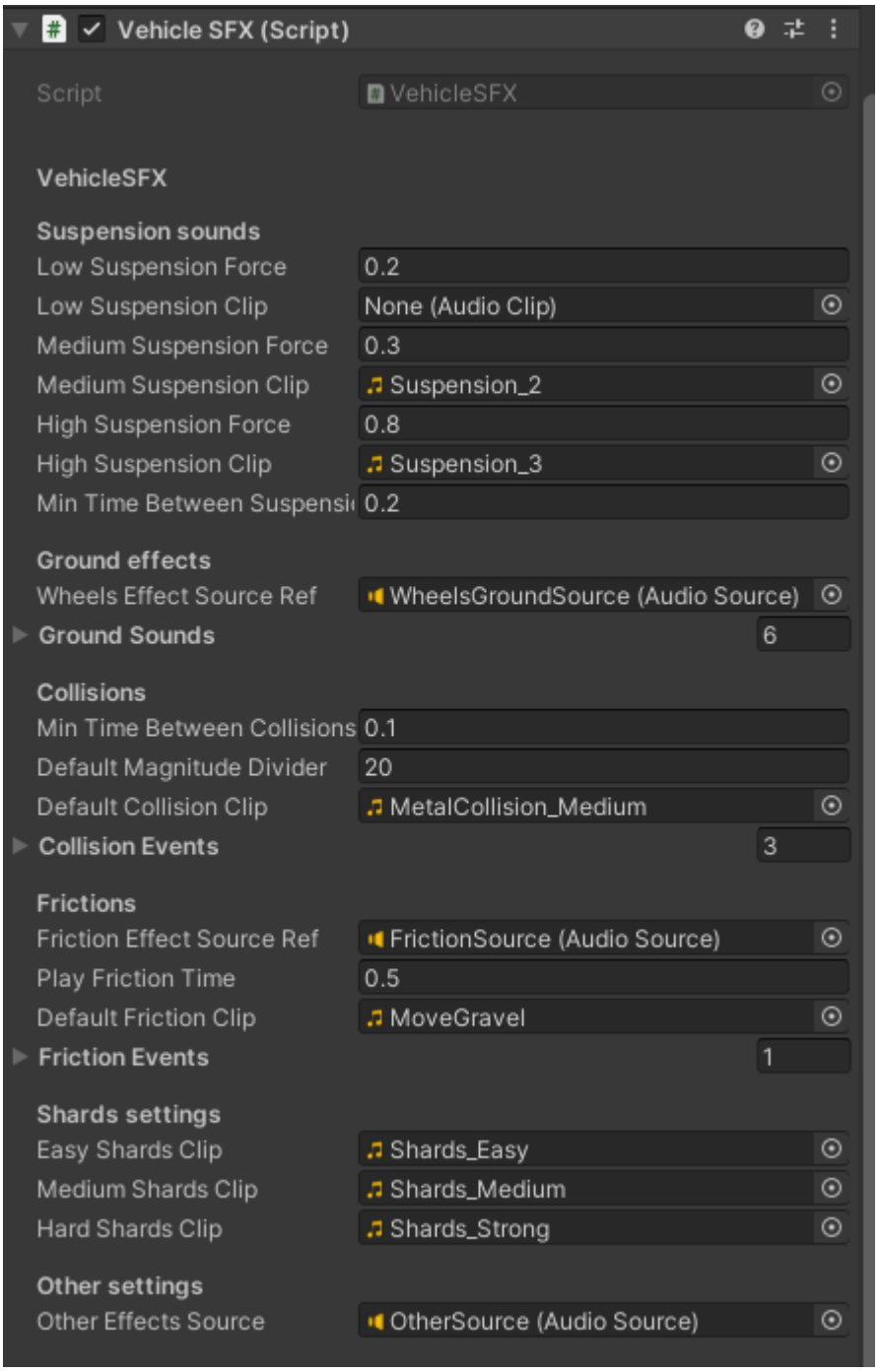
Sound effects

The player's car has to have an AudioListener attached. when using PG.GameController AudioListiner is turned on/off automatically.

VehicleSFX

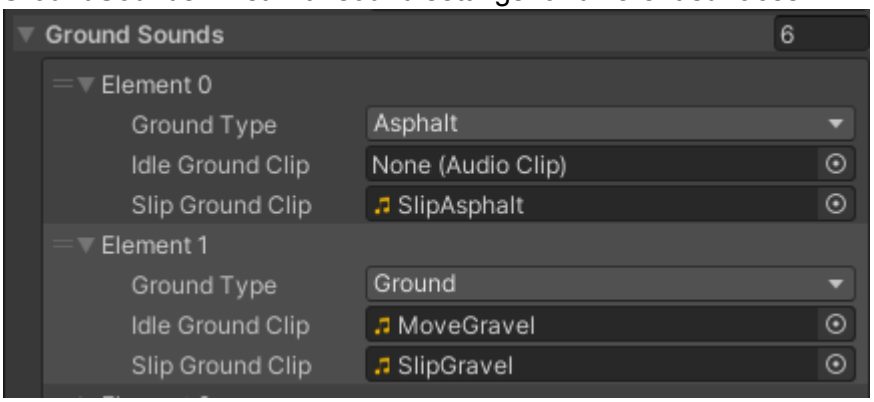
Base component for handling VehicleSFX, must be a child of VehicleController. Some sounds are not handled by this component (for ex. glass scatter sound)

Contains fields:



- LowSuspensionForce - The force at which a weak suspension sound will be played (Where force is 0 - the suspension does not change, 1 - the maximum suspension travel per update). MediumSuspensionForce and HighSuspensionForce work in a similar way.
- LowSuspensionClip - Weak suspension sound, plays when present. MediumSuspensionClip and HighSuspensionClip work in a similar way.

- GroundSounds - List with sound settings for different surfaces



  Contains fields:

  - GroundType - Surface type.
  - IdleGroundClip - The sound played when driving on the surface.
  - SlipGroundClip - The sound played when sliding on the surface.

- MinTimeBetweenCollisions – minimum time between sound fires to avoid artifacts during intense collisions.
- DefaultMagnitudeDivider – max magnitude divider to calculate sound volume if CollisionEvent is not found
- DefaultCollisionClip – sound played if CollisionEvent is not found
- CollisionEvents – array of type CollisionEvent determines the sound to play. For one layer you can set multiple sounds depending on collision magnitude
- FrictionEffectSourceRef - link to the Source, a copy is created for the sounds of sliding on various surfaces (Sounds are configured in the FrictionEvents list),
- PlayFrictionTime - Friction play time, if the contact occurs more often than PlayFrictionTime, then the sound is played in a circle.
- DefaultFrictionClip - The friction sound played if no ColissionEvent is found when friction.
- FrictionEvents - An array of type CollisionEvent. This array determines the sounds played when sliding, for one layer you can set several sounds depending on the strength of the slide.
- EasyShardsClip - The sound of weak shards on impact, such as a broken light.
- MediumShardsClip - The sound of medium fragments on impact, for example, the entire headlight is broken.
- HardShardsClip - The sound of hard shards on impact, such as broken glass.
- OtherEffectsSource - Source of additional effects that do not need a constant source, such as collision sounds, exhaust pipe shots, etc..

CollisionEvent contains fields:

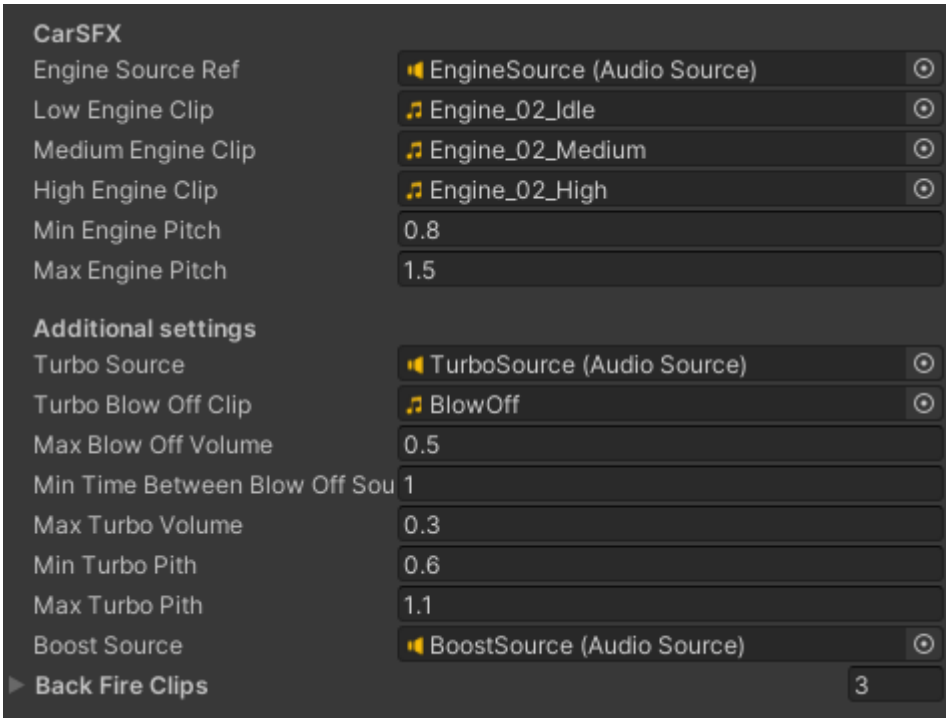
- CollisionEvent.Event – name of the sound event
- CollisionEvent.CollisionLayer – layer, colliding with which event is fired

- CollisionEvent.MinMagnitudeCollision – event is called if collision magnitude is above this value
- CollisionEvent.MaxMagnitudeCollision – event is called if collision magnitude is above this value, the same value is used as divider to determine the sound volume

CarSFX

CarSFX inherits VehicleSFX, must be child of CarController. Used to play engine sounds.

Contains fields:



- EngineSource - that plays engine sound constantly, serves as an reference for other engine sounds.
- LowEngineClip - (Optional), Engine sound at low RPM, played from 0 to 30 percent RPM.
- MediumEngineClip - (Optional), Engine sound at medium RPM, played from 30 to 60 percent RPM.
- HighEngineClip - (Optional), Engine sound at high RPM, played from 60 to 100 percent RPM.

The LowEngineClip, MediumEngineClip, HighEngineClip fields can be Null, in which case the sound will be played from the EngineSource.

- MinEnginePtich - Pitch at minimum engine speed.
- MaxEnginePtich - Pitch at maximum engine speed.
- TurboSource - Source for playing turbine sound.
- TurboBlowOffClip - Turbine BlowOff sound.
- MaxBlowOffVolume - The maximum volume of the BlowOff sound.
- MinTimeBetweenBlowOffSounds - The minimum time between BlowOff sounds (so that the release sound does not bother you).
- MaxTurboVolume - The maximum volume of the turbine sound.
- MinTurboPith - Pitch at minimum turbine speed.

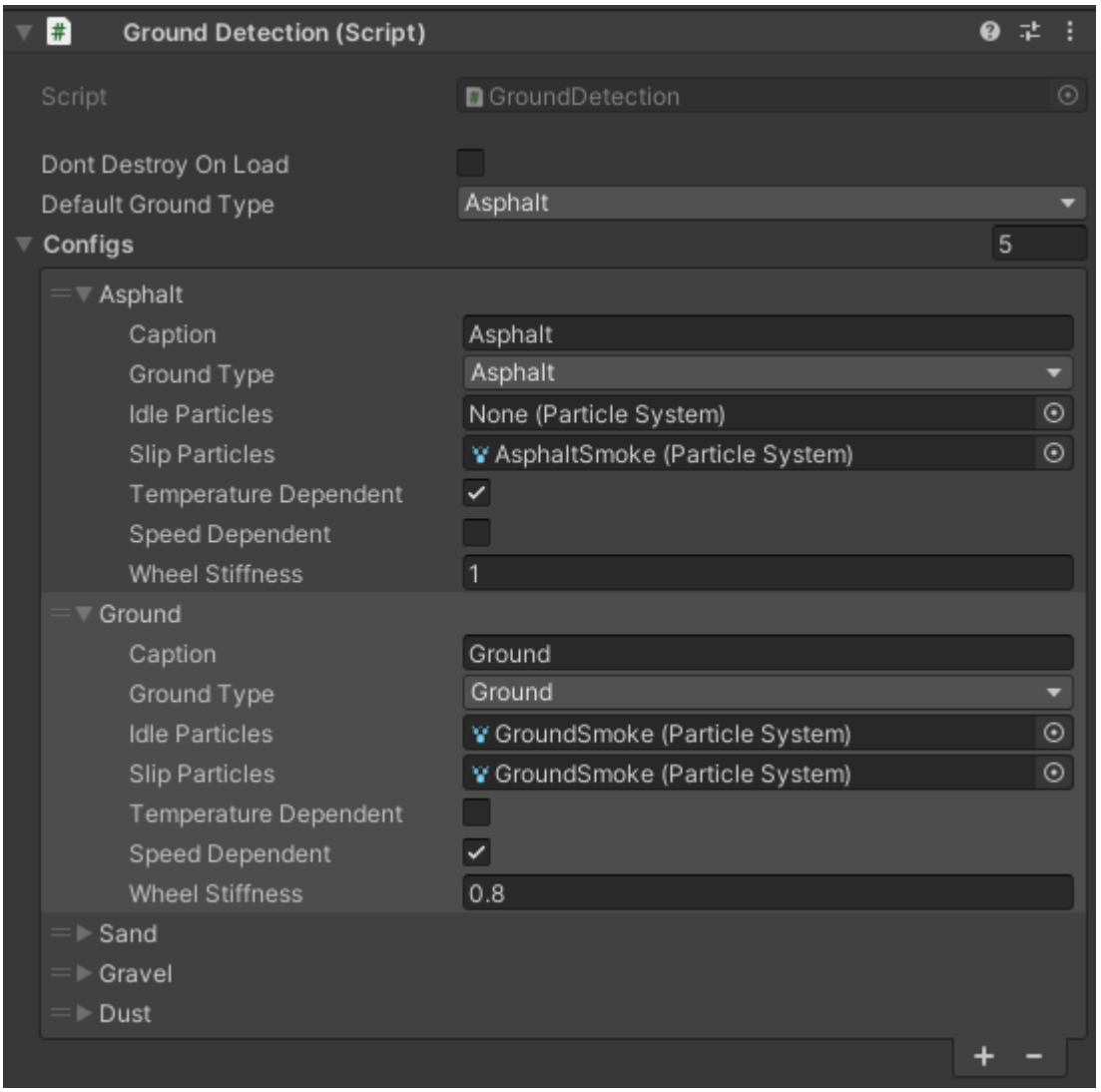
- MaxTurboPith - Pitch at maximum turbine speed.
- BoostSource - Source of nitro acceleration, just toggle on / off when used.
- BackFireClips - List of exhaust fire sounds.
- OtherEffectsSource - Source in which additional sounds are played (Sound of fire from the exhaust pipe, BlowOff sound).

GroundDetection:

GroundDetection.

Class which stores all the IGroundEntity attached to GameObjects on which the wheel is driving. In this project, the GroundDetection component is located in the GroundDetection prefab, which contains all Particle Systems (Dust/Smoke from under the wheels).

Contains fields:

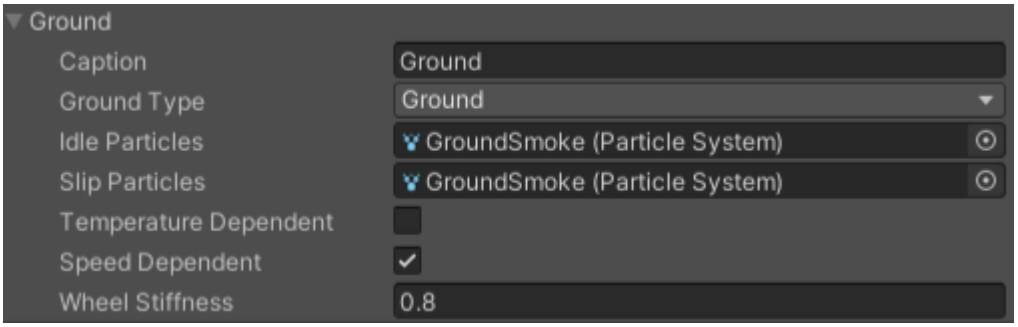


- DefaultGroundType - Default ground type (If no type is found for the current ground).
- Configs - List of GroundConfig surface settings.

GroundConfig

Contains ground settings on which depend wheel friction, particle systems and sounds.

Contains fields:



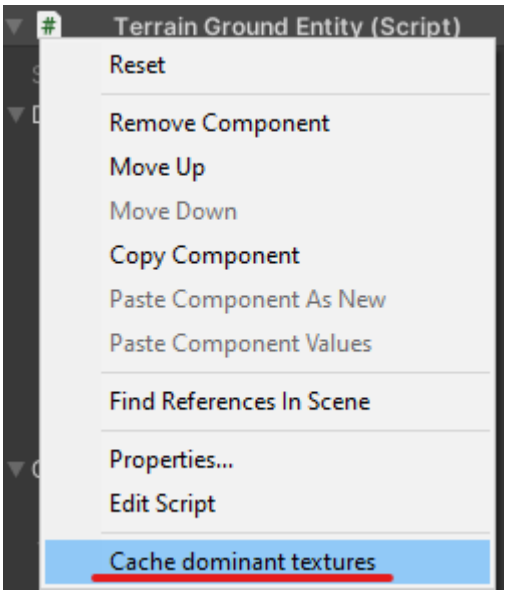
- Caption – just name of the preset for ease of use
- GroundType - Ground type.
- IdleParticles – particle system that emits particles if a wheel is just cruising on the ground (without slip), if equal to null particles will not be emitted
- SlipParticles – particle system that emits particles if a wheel is slipping on the ground (without slip), if equal to null particles will not be emitted
- TemperatureDependent – dependency on temperature, the higher the temperature – the further the particles are emitted (for ex. smoke from tires increases with the temperature on the road)
- SpeedDependent – dependency on speed, the higher the speed – the further the particles are emitted
- WheelStiffness – multiplier for wheel friction

ObjectGroundEntity

This component is used to determine the ground on which the wheel is going. Has just a field of type GroundType, when the wheel touches that ground, this GroundType will be passed to the wheel

TerrainGroundEntity

!!!WARNING!!! After editing the terrain or TerrainGroundEntity you need to execute a command from TerrainGroundEntity's contextual menu "Cache dominant textures" to calculate and store all textures into hidden array DominateTextures

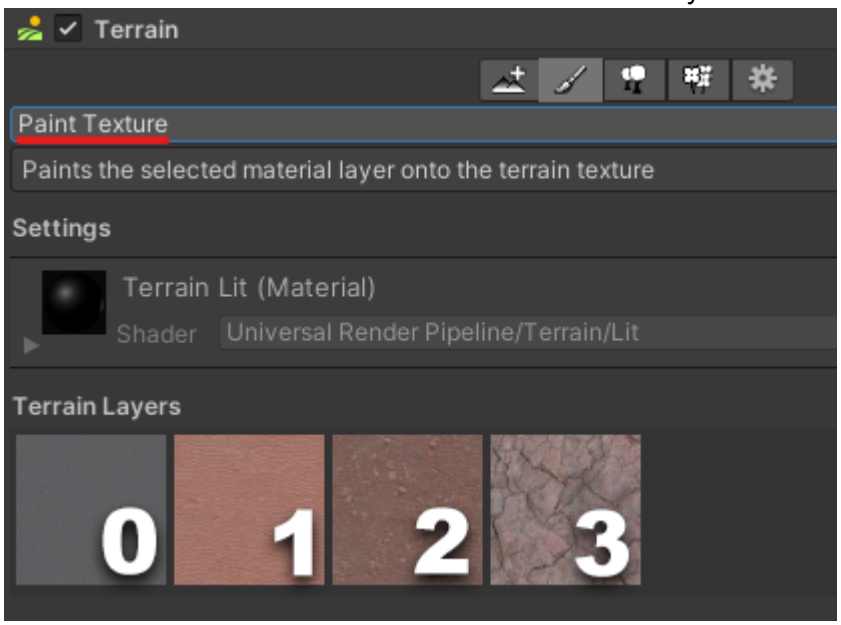


TerrainGroundEntity – this component is used to configure GroundConfigs, for every terrain texture you can set unique config

Contains fields:

- DefaultType – default type in case texture config is not found
- GroundConfigs – List of type TerrainGroundConfig, configs for different textures

TerrainGroundConfig is a shell for GroundConfig has an additional field TextureIndex where you need to set an index for the texture. An index is a terrain layer.

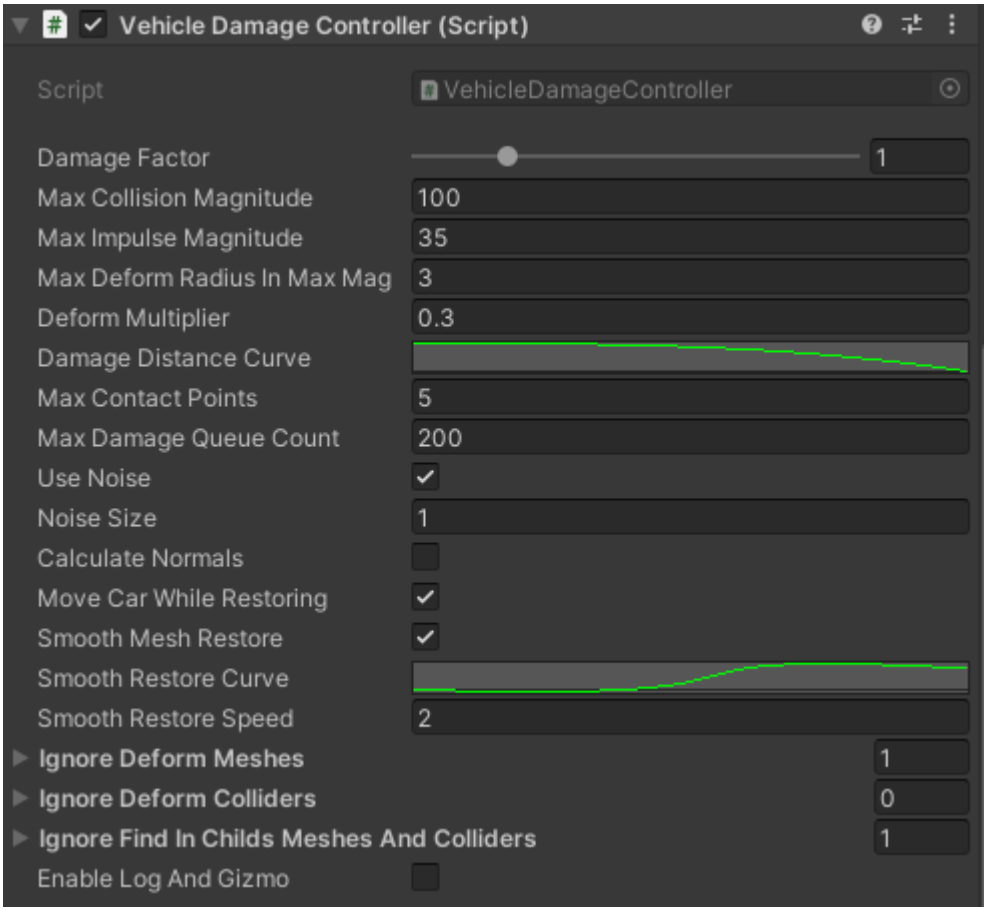


CarDamage

VehicleDamageController

Main component handling all the damages. At the start of the scene it searches for all Meshes, MeshColliders, DamageableObjects, DetachableObjects and MoveObjects (except object from IgnoreDeformMeshes and IgnoreDeformColliders) and adds them to the appropriate list for further damage application

Contains fields:



- DamageFactor – damage multiplier
- MaxCollisionMagnitude – max damage
- MaxDeformDistanceInMaxMag – max distance for damageable vertices, for instance: if max hit strength is 25 (25% of maximum) then the damage will spread to all vertices and object in radius 0.75 units from the hit origin (the further from hit origin the smaller is the damage)
- MaxContactPoints – max number of contact points in one hit
- MaxDamageQueueCount - The maximum number of collisions in the queue, for damage processing between which there was little time, this is necessary for optimization
- DamageDistanceCurve – the curve for calculating damage hit strength on a certain distance. Everywhere in the project square of the length is used (for optimization) which is why the force is calculated non-linearly. This curve is helping us avoid huge calculations
- UseNoise – using noise when damaging visible Meshes
- NoiseSize – noise size (the greater the value, the more the noise will appear)

- CalculateNormals – recalculate normal after the collision
- MoveCarWhileRestoring - Move the car while restoring, if the checkbox is active then the car turns the wheels down and moves one unit up
- SmoothMeshRestore - Enables smooth recovery, if the checkbox is disabled, then recovery will occur in 1 frame.
- SmoothRestoreCurve - Mesh restore animation curve, must start with key 0.0 and end with key 1.1
- SmoothRestoreSpeed - Recovery animation speed, the higher the speed, the faster the car will recover
- IgnoreDeformMeshes – list of MeshFilters to ignore during initialization
- IgnoreDeformColliders – list of MeshColliders to ignore during initialization
- IgnoreFindInChildsMeshesAndColliders - Objects which will not be searched for damaged meshes and colliders.

DamageableObject

Damageable objects have health and receive damage (the amount of damage is dependent on hit strength and distance of LocalCenterPoint from hit origin). LocalCenterPoint – is located in the center of mass of Mesh vertices if the object has MeshFilter, otherwise LocalCenterPoint == Vector3.zero.

You can inherit all damageable objects from this object (for ex. engine, transmission, steering) like it is done right now for some elements (wheels, glass, lights).

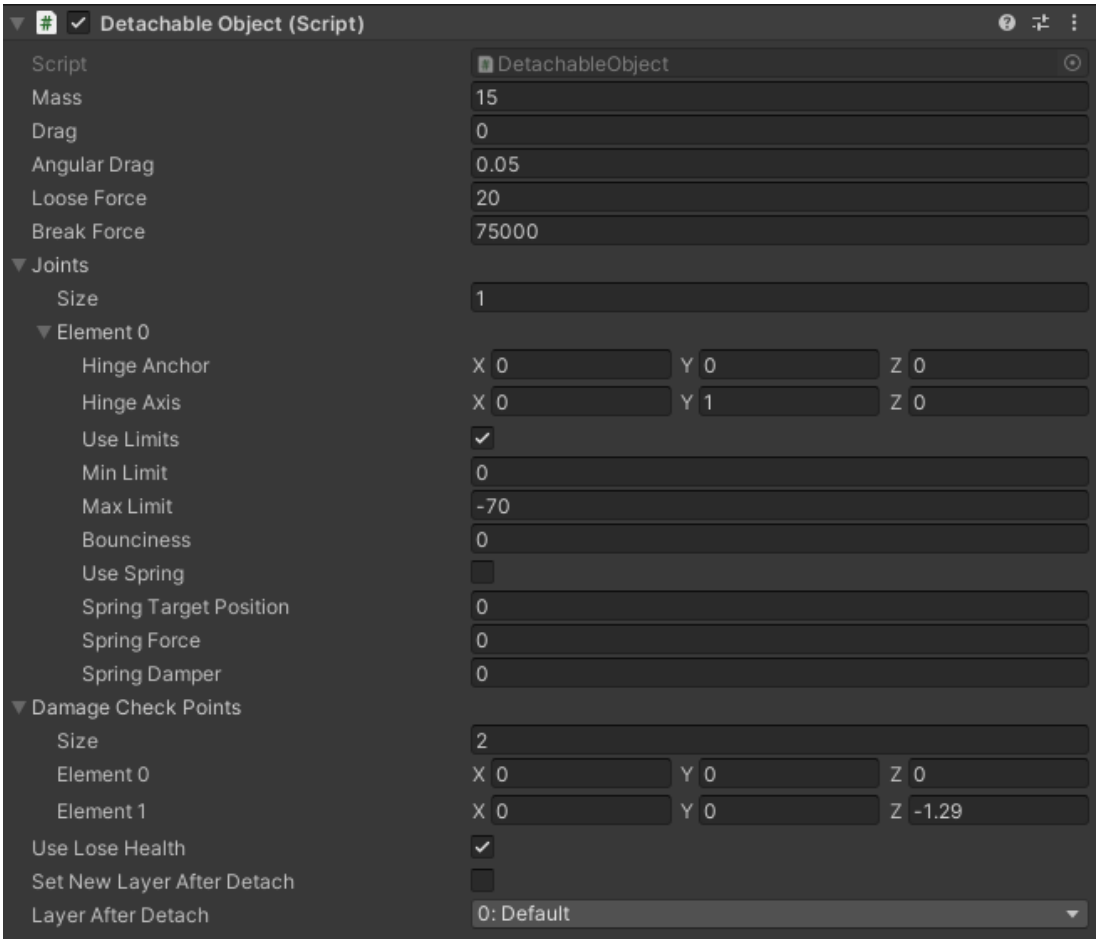
Contains two fields:

- Health – health in the beginning
- MaxDamage – max damage in one hit

DetachableObject

An object that can be detached from a vehicle (hang on one of several possible joints), then be detached completely. Used mainly for vehicle's body parts. When detached all DamageableObject receive max damage.

Contains fields:

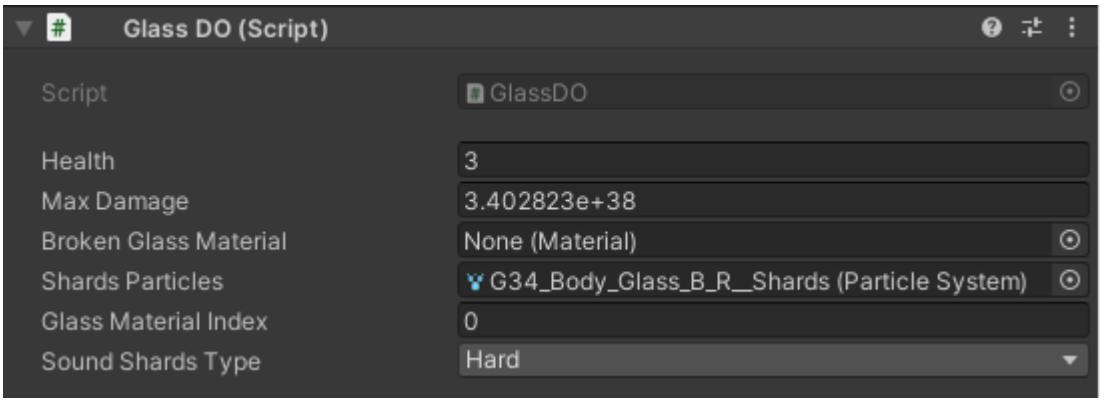


- Mass – mass of an object. This mass is assigned to the part's Rigidbody and subtracts from vehicle's mass
- Drag – is assigned to the corresponding Rigidbody field
- AngularDrag – is assigned to the corresponding Rigidbody field
- LoseForce – hit strength which will partially detach an object (will hang on a random joints) and Rigidbody will be created
- Joints – list of joints. When an object is partially detached a joint is created between an object and a vehicle with the specified parameters from the list (random is selected). Unfortunately, I couldn't figure out a good way to edit these settings. Best way right now is to create a HingeJoint, configure it and copy component values to Joints list. In future I will try to provide a better solution
- DamageCheckPoints – list of points that are checked during a hit to determine whether it will be detached. A list must contain at least one point
- UseLoseHealth – flag used for accumulation of damage. If enabled, then to detach an object there needs to be dealt damage greater or equal to LoseForce in one hit
- SetNewLayerAfterDetach - Whether to assign a new layer to the detachable part. It is necessary, for example, for headlights, after disconnecting they are assigned a layer which does not collide with the car (so that the headlight does not get stuck in the car).

GlassDO

This component was already described in LightObject. It is used to handle the logic for breaking the glass.

Contains fields:



- BrokenGlassMaterial – broken material if you need to show shards after the destruction. If this field is null, then after destruction the object will disappear.
- ShardsParticles – particle system instantiated during destruction.
- GlassMaterialIndex – material index in case Mesh has more than one SubMesh.
- SoundShardsType - sound played when an object is destroyed.

MoveableDO

Object which is moved to the vector of a damage force (depends on the hit strength). Used in wheels and steering wheel

GameController и PlayerController

Utility component for initialization of cars, camera and UI

GameController_WithCarSelector searches for all vehicles and gives you the ability to switch between them, if there are no vehicles in the scene, the first one from GameSetting.AvailableVehicles is instantiated. This class is for demonstration purposes. GameController also has a menu to navigate between scenes and a guide for controls.

GameController_WithEnterExitLogic contains a menu prefab and a SimpleCharacterController prefab, by placing GameController_WithEnterExitLogic on the scene you will be able to enter and exit the car, then there will be a description of the SimpleCharacterController.

PlayerController contains Camera, UI stats of a vehicle and UserInput.

Important! You can remove GameController and PlayerController, manually place a CameraController, CarStateUI and UserInput, assign them a TargetCar and add the AudioListiner component to the player car, after this everything will work, just without initialization. You can also create your own GameController which will initialize PlayerController.

Simple CharacterController

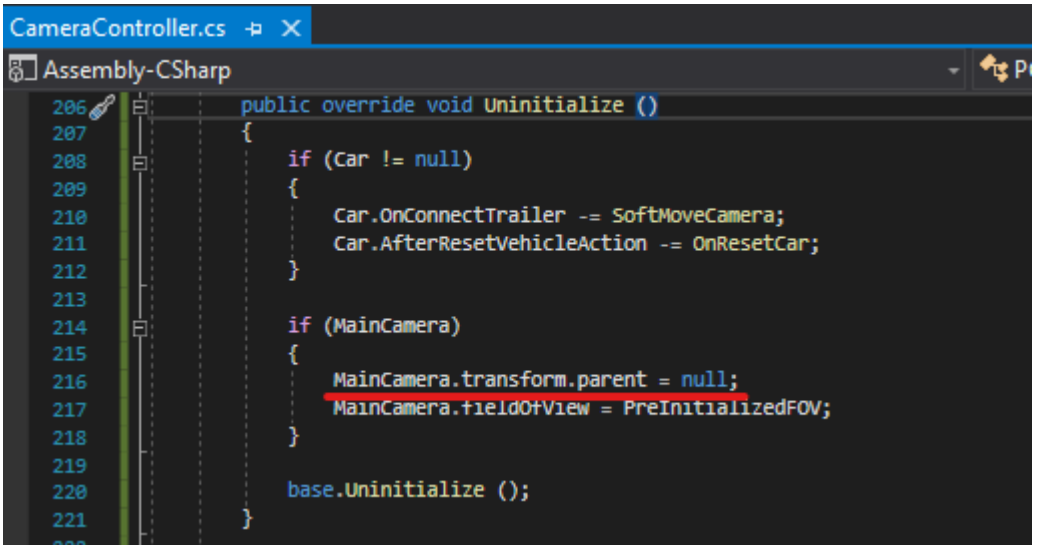
SimpleCharacterController is needed to demonstrate the enter/exit to the car. To use it, just place the GameController_WithEnterExitLogic prefab on the scene in the right place (So that the CharacterController capsule is above the surface). This component has very simple functionality and was created only to demonstrate the possibilities of UVC.

To use a third-party CharacterController with enter/exit functionality, you need to:

- Place a PlayerController or PlayerController_Mobile on the scene
- When entering a car, you can use the PlayerController.EnterInCar(Car) method
- After calling the method, you must disable the CharacterController control (for example, turn off the object or start the animation of entering the car and turn off the control).
- You can also subscribe to the PlayerController.OnExitAction<CarController> car exit event to enable control of the CharacterController.

How this is done you can see in SimpleCharacterController.cs methods TryEnterCar () and OnExitFromCar (CarController car).

You can use the same camera for PG.CameraController and 3rd party CharacterController, when exiting the car CameraController.MainCamera is freed (becomes parentless and remains active in the hierarchy) for smooth camera movement. If you don't need this feature, you can remove this line from the CameraController.Uninitialize() method.

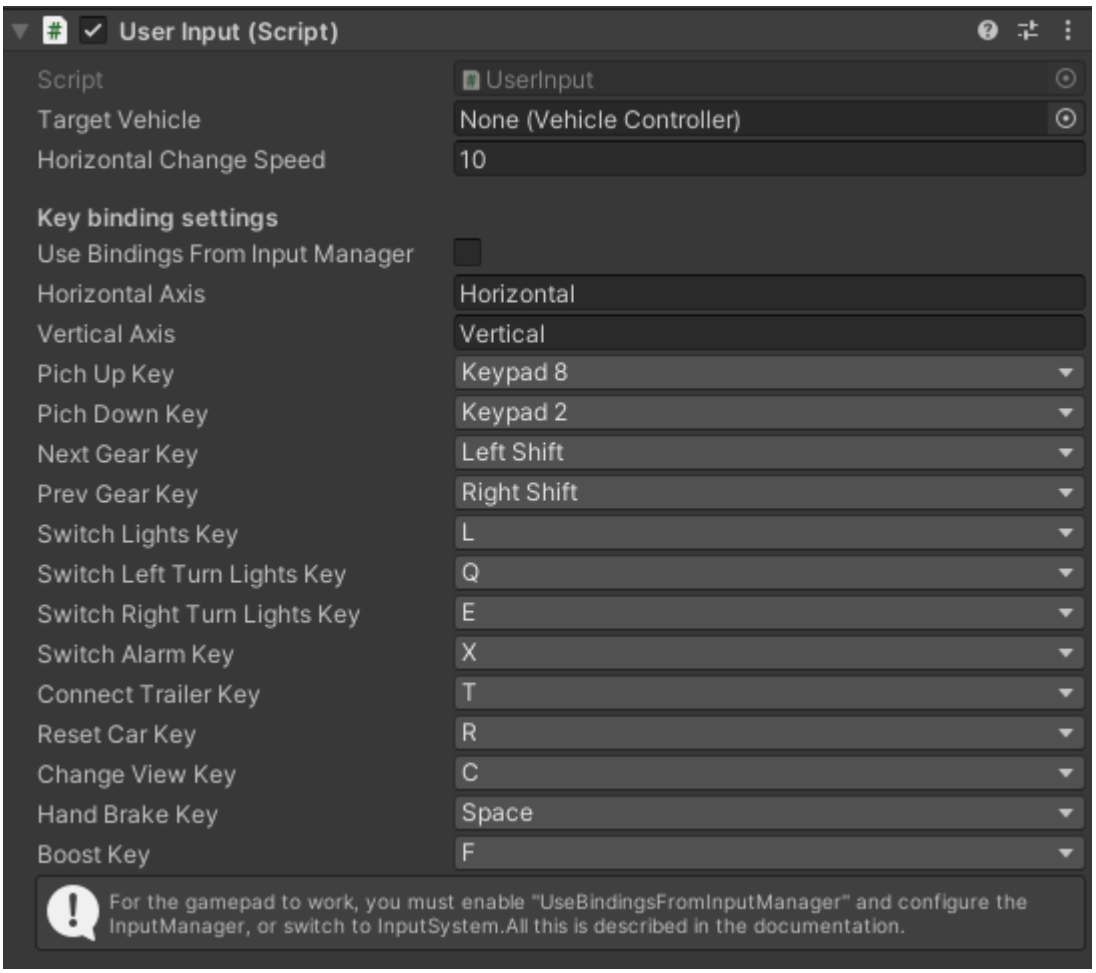


Input

By default, input is available only from the keyboard and from the UI for mobile devices. To work with the gamepad, you need to configure the InputManager or use the InputSystem (How to do this was described in the section "Add-ons>InputSystem")

UserInput.

Contains fields:

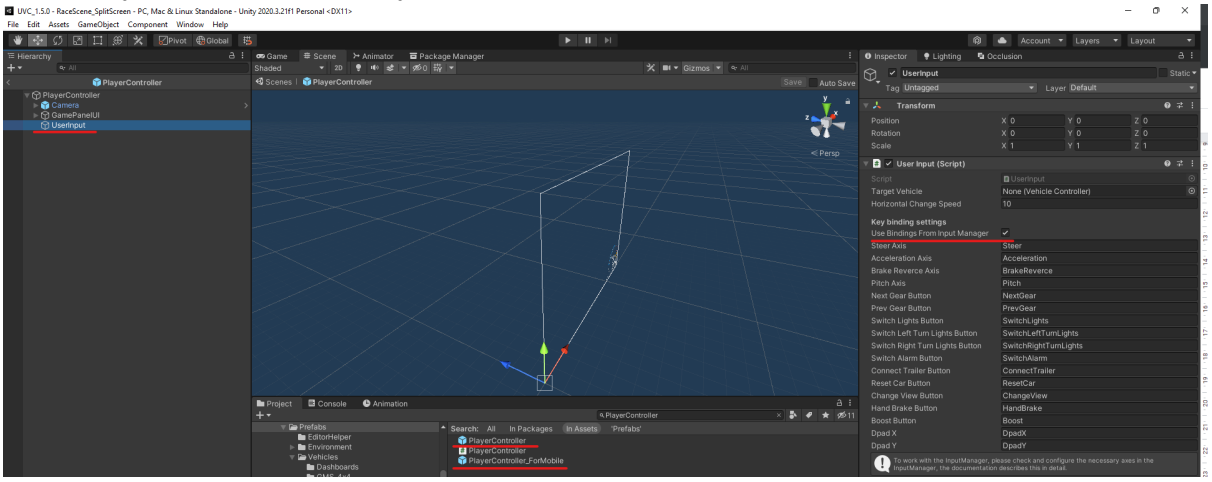


- TargetCar – controlled car, this field is used for manual configuration if UserInput (for ex. the car and UserInput are already in the scene and you do not need to initialize them).
- HorizontalChangeSpeed – to imitate joysticks when hitting the keyboard. Before implementing this the car would accidentally jerk during fast turns using the keyboard
- UseBindingsFromInputManager - Includes a dependency on the InputManager, below will describe how to configure the InputManager to use the gamepad.
- HorizontalAxis - The horizontal axis for steering the vehicle.
- VerticalAxis - Vertical axis to control vehicle acceleration/brake.
- *Keys - The rest of the buttons are necessary to control the car.

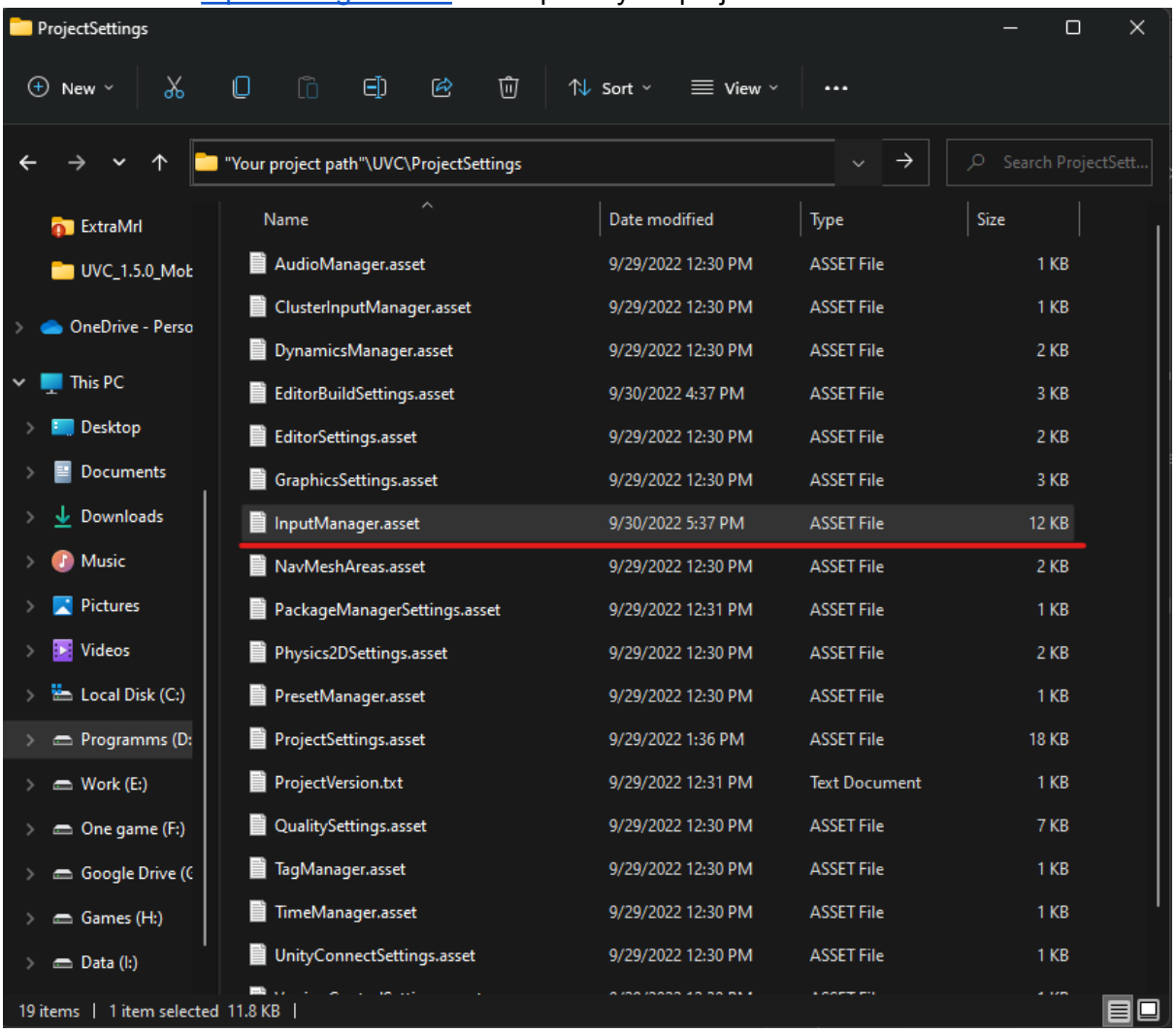
InputManager (Gamepad)

If you are creating a game for a PC or game consoles (Devices without a touch screen), it is better to use the InputSystem dependency, how to do this was described in the section "Add-ons>InputSystem".

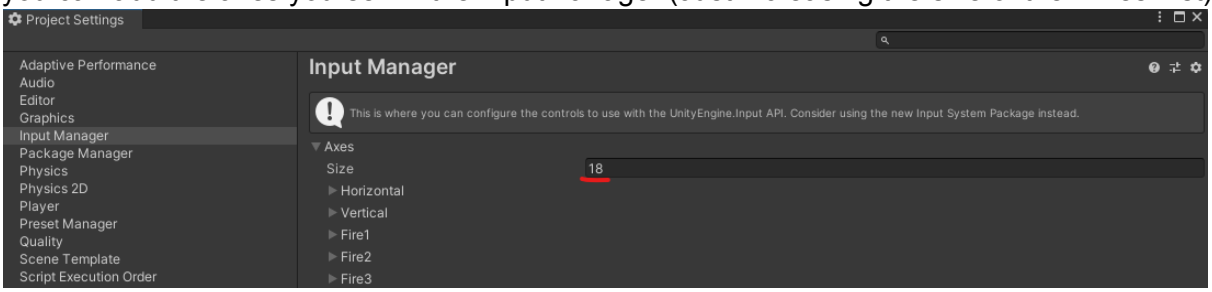
To configure the gamepad, you need to enable the PlayerInput .UseBindingsFromInputManager flags in the PlayerController and PlayerController_ForMobile prefabs, as shown in the screenshot



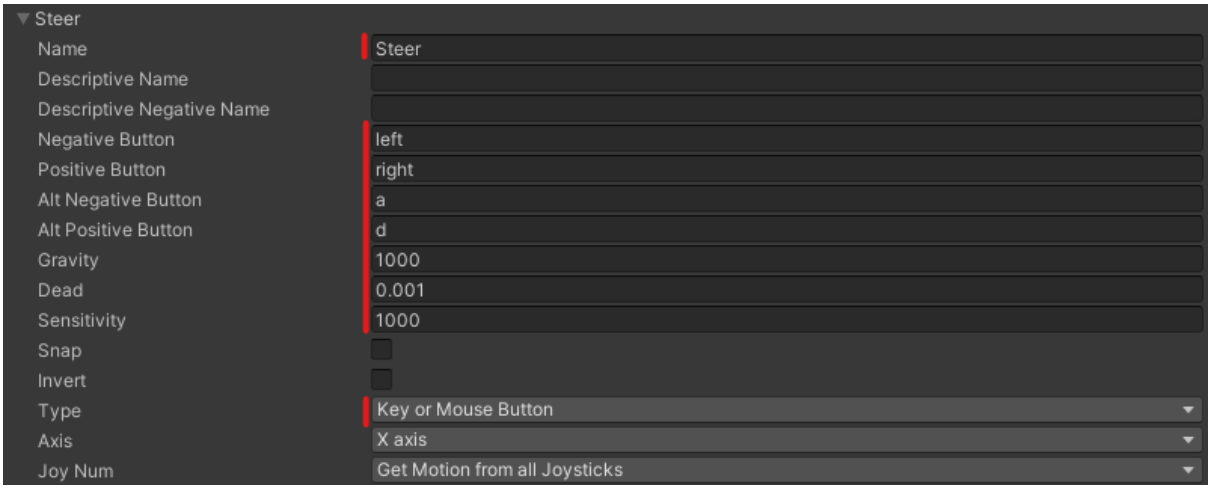
You can download InputManager.asset and replace your project file.



Or you can add the axes yourself in the InputManager (Just increasing the size of the "Axes" list):

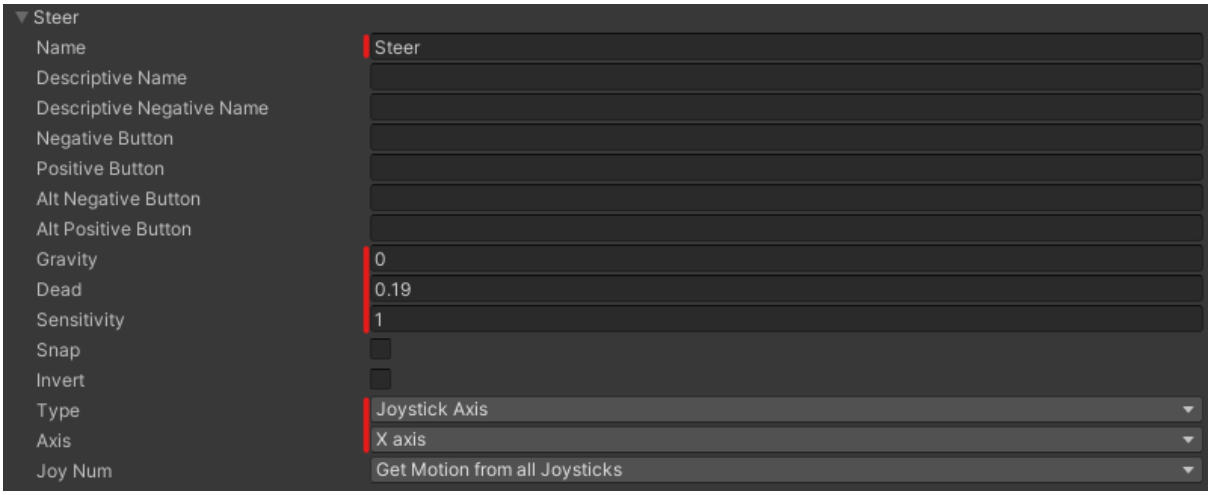


- Axis Steer for the keyboard (The screenshot shows an example, further there will be axes without screenshots).



  - Name = "Steer"
  - Negative Button = "left"
  - Positive Button = "right"
  - Alt Negative Button = "a"
  - Alt Positive Button = "d"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis Steer for the gamepad (The screenshot shows an example, further there will be axes without screenshots).



  - Name = "Steer"
  - Gravity = 0
  - Dead = 0.19
  - Sensitivity = 1
  - Type = JoystickAxis
  - Axis = X axis

- Axis Acceleration for keyboard
  - Name = "Acceleration"
  - Positive Button = "up"
  - Alt Positive Button = "w"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis Acceleration for gamepad
  - Name = "Acceleration"
  - Gravity = 0
  - Dead = 0.19
  - Sensitivity = 1
  - Type = JoystickAxis
  - Axis = 10th axis

- Axis BrakeReverse for keyboard
  - Name = "BrakeReverse"
  - Positive Button = "down"
  - Alt Positive Button = "s"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis BrakeReverse for gamepad
  - Name = "BrakeReverse"
  - Gravity = 0
  - Dead = 0.19
  - Sensitivity = 1
  - Type = JoystickAxis
  - Axis = 9th axis

- Axis Pitch for keyboard
  - Name = "Pitch"
  - Negative Button = "[2]"
  - Positive Button = "[8]"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis NextGear for keyboard and gamepad
  - Name = "NextGear"
  - Positive Button = "left shift"
  - Alt Positive Button = "joystick button 5"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis PrevGear for keyboard and gamepad
  - Name = "PrevGear"
  - Positive Button = "left ctrl"
  - Alt Positive Button = "joystick button 4"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis SwitchLights for keyboard and gamepad
  - Name = "SwitchLights"
  - Positive Button = "l"
  - Alt Positive Button = "joystick button 8"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis SwitchLeftTurnLights for keyboard
  - Name = "SwitchLeftTurnLights"
  - Positive Button = "q"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis SwitchRightTurnLights for keyboard
  - Name = "SwitchRightTurnLights"
  - Positive Button = "e"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis SwitchAlarm for keyboard
  - Name = "SwitchAlarm"
  - Positive Button = "x"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis ResetCar for keyboard and gamepad
  - Name = "ResetCar"
  - Positive Button = "r"
  - Alt Positive Button = "joystick button 2"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis ChangeView for keyboard and gamepad
  - Name = "ChangeView"
  - Positive Button = "c"
  - Alt Positive Button = "joystick button 3"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000
  - Type = KeyOrBouseButton

- Axis HandBrake for keyboard and gamepad
  - Name = "HandBrake"
  - Positive Button = "space"
  - Alt Positive Button = "joystick button 0"
  - Gravity = 1000
  - Dead = 0.001
  - Sensivility = 1000

    - Type = KeyOrBouseButton

- Axis Boost for keyboard and gamepad
    - Name = "Boost"
    - Positive Button = "f"
    - Alt Positive Button = "joystick button 1"
    - Gravity = 1000
    - Dead = 0.001
    - Sensivility = 1000
    - Type = KeyOrBouseButton

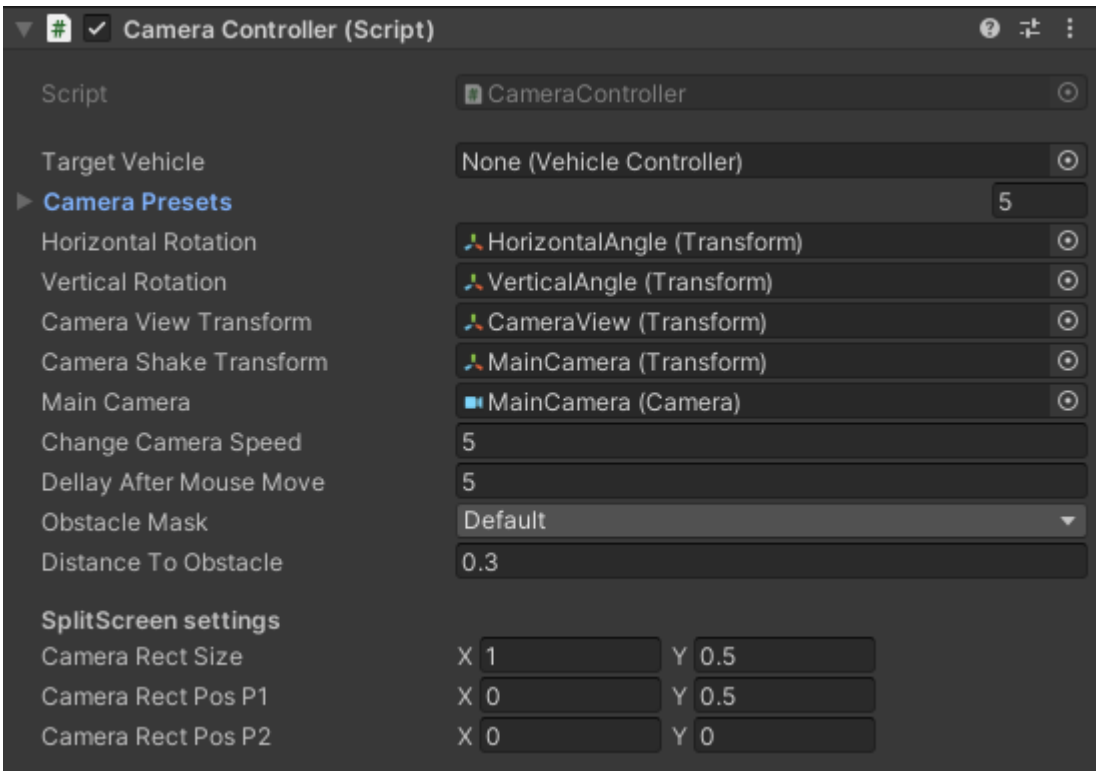
- Axis DpadX for turn signals on gamepad
    - Name = "DpadX"
    - Gravity = 1000
    - Dead = 0.001
    - Sensivility = 1000
    - Type = JoystickAxis
    - Axis = 6th axis

- Axis DpadY for alarm signals and connecting trailers on gamepad.
    - Name = "DpadY"
    - Gravity = 1000
    - Dead = 0.001
    - Sensivility = 1000
    - Type = JoystickAxis
    - Axis = 7th axis

CameraController

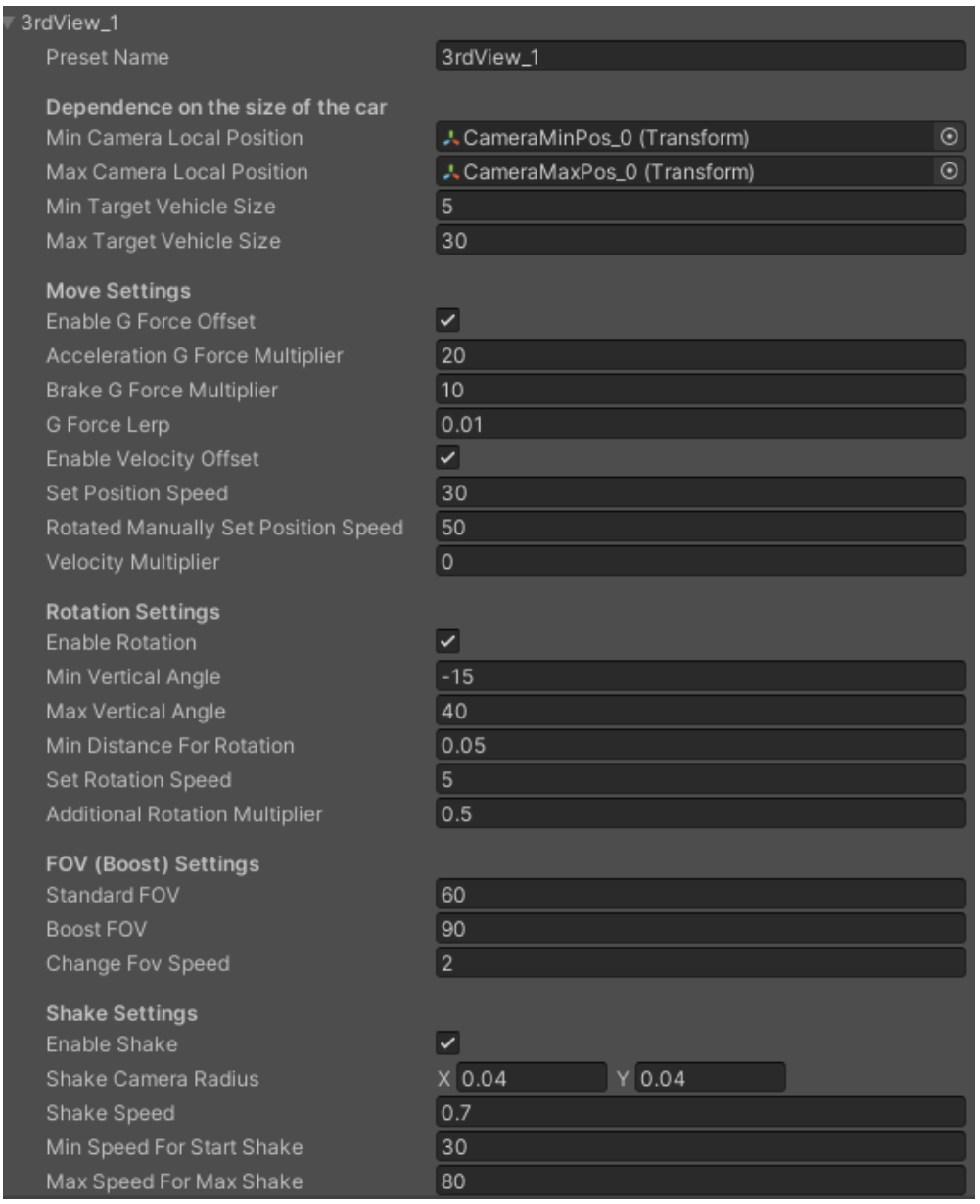
This component has logic for camera controls (follow an object, rotation – automatic and manual), switching different modes

Contains fields:



- TargetVehicle – vehicle to be followed by a camera. This field is used to manually select target vehicle (when you everything in place and you don't need initialization)
- CameraPresets – List of type CameraPreset camera presets, for settings of view types
- HorizontalRotation – object that is rotating horizontally
- VerticalRotation - object that is rotating vertically
- CameraView – Camera object which is placed in CameraLocalPosition depending on selected view mode. Must be a child of VerticalRotation.
- CameraShakeView - The object that shakes while moving must be a child of the CameraViewTransform.
- MainCamera - camera component. may be null, in this case the camera will be searched on the scene, if there is no camera on the scene, a new camera will be created from ResourcesSettings.
- ChangeCameraSpeed – speed of switching between view modes
- DistanceAfterMouseMove – if the camera is moved manually, then after vehicle covers this distance the camera with reset its position

CameraPreset contains fields:



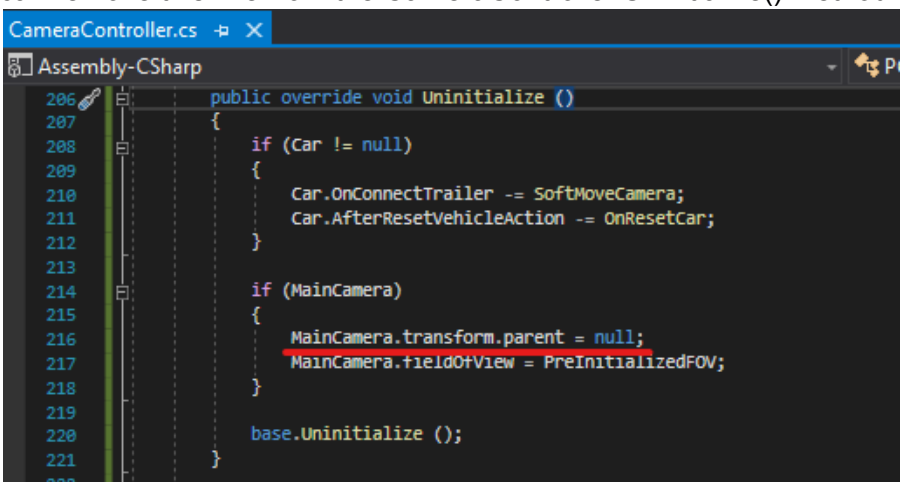
- MinCameraLocalPosition and MaxCameraLocalPosition – depending on size of a vehicle (with a trailer) camera's position and rotation is calculated
- MinCameraLocalPosition and MaxCameraLocalPosition need to be childs of VerticalRotation. After initialization MinCameraLocalPosition and MaxCameraLocalPosition are deleted (position and rotation are stored in a variable)
- MinTargetVehicleSize – min vehicle size, if a vehicle is less or equal this size, then position and rotation for the camera will be MinCameraLocalPosition.
- MaxTargetVehicleSize – max vehicle size, if a vehicle is greater or equal this size, then position and rotation for the camera will be MaxCameraLocalPosition.
- EnableGForceOffset - Enable camera movement during acceleration / braking.
- AccelerationGForceMultiplier - The multiplier to move the camera (Back) when accelerating.
- BrakeGForceMultiplier - The force to move the camera (Forward) when braking.
- GForceLerp - Offset interpolation for smoothness.
- EnableVelocityOffset - Enable offset based on the current speed.

- SetPositionSpeed – camera speed
- VelocityMultiplier – force of advance of following point (for ex. if equal to 0, camera will follow the vehicle depending on its speed)
- EnableRotation – enables and disable camera rotation (manual and automatic)
- MinVerticalAngle – minimum angle of vertical rotation
- MaxVerticalAngle – max angle of vertical rotation
- MinDistanceForRotation – min distance covered by a vehicle to update automatic rotation (to prevent camera from chaotic rotations while the vehicle stands still)
- SetRotationSpeed – rotation speed of a camera
- AdditionalRotationMultiplier – additional rotation towards drift
- StandardFOV - The default FOV.
- BoostFOV - FOV applied while boost is being used.
- ChangeFovSpeed - FOV change rate when boost is turned on / off.
- EnableShake - Enables camera shake while the car is moving.
- ShakeCameraRadius - The radius in which the camera will shake (Points are chosen randomly).
- ShakeSpeed - Shaking speed.
- MinSpeedForStartShake - The speed at which the camera starts shaking.
- MaxSpeedForStartShake - The speed at which the camera is shaken at maximum.

!Important!

The camera has a CameraController.Initialize (VehicleController vehicle) method, if CameraController.MainCamera == null then the camera is searched on the scene (For smooth camera movement), if you do not need this functionality, then just place the camera in the CameraController.prefab and specify it in CameraController.MainCamera.

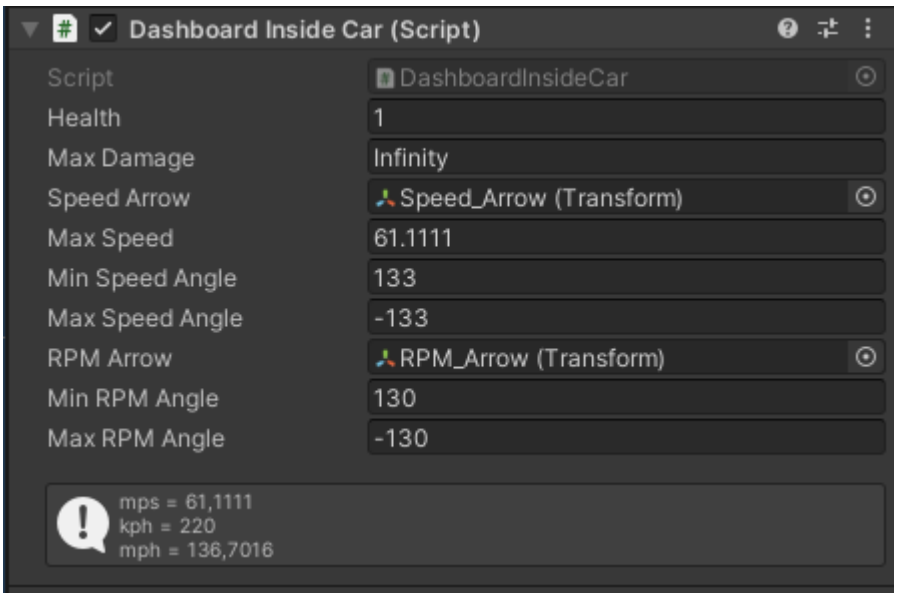
The camera has a method CameraController.Uninitialize (VehicleController vehicle), when this method is executed, CameraController.MainCamera is released (becomes parentless and remains active in the hierarchy), for smooth movement of the camera. If you don't need this feature, you can remove this line from the CameraController.Uninitialize() method.



DashboardInsideCar

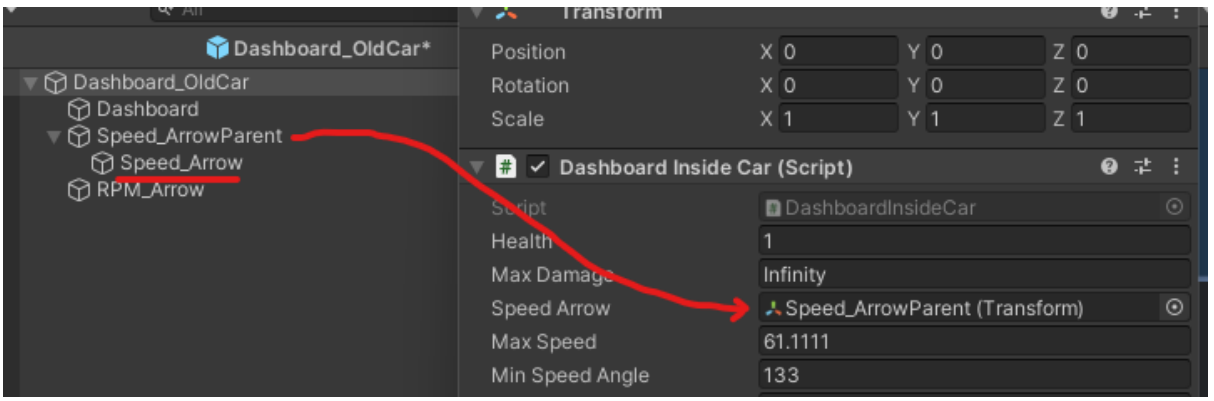
A component for displaying arrows on a car dashboard. Inherits DamageableObject, arrows do not update their rotation when damaged.

contains fields:



- SpeedArow - Arrow indicating speed (The arrow rotates along the Z axis).
- MaxSpeed - The maximum speed on the dashboard, units of measure are meters per second (Bottom is how much this is in kph and mph).
- MinSpeedAngle - Speed arrow angle at 0 speed.
- MaxSpeedAngle - The angle of the speed arrow at maximum (MaxSpeed) speed.
- RPMArrow - Arrow showing RPM (The arrow rotates along the Z axis).
- MinRPMAngle - Rotation arrow angle at 0 rpm.
- MaxRPMAngle - RPM arrow angle at maximum RPM (Value from vehicle settings).

If your arrow needs to rotate along a different axis, then you can place it inside the object that will rotate, and rotate the arrow as you need.



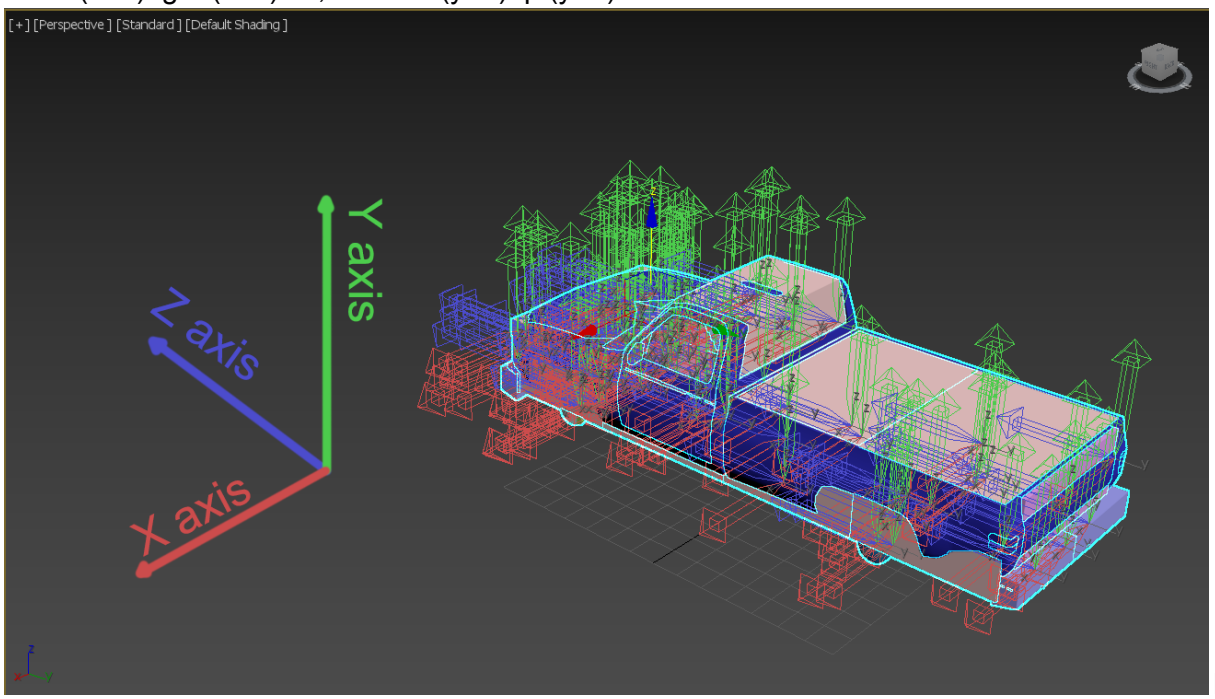
There are 2 dashboard prefabs in the asset, you can take them as an example for your cars.

Car Creation (CreateCarWindow)

Preparing car models.

1) The vehicle and all of its parts must have proper scale (like in real life) transform.localScale for all the parts should be equal to Vector3.one (1, 1, 1), otherwise VehicleDamageController (Damage) will not work correctly and will need to be redone.

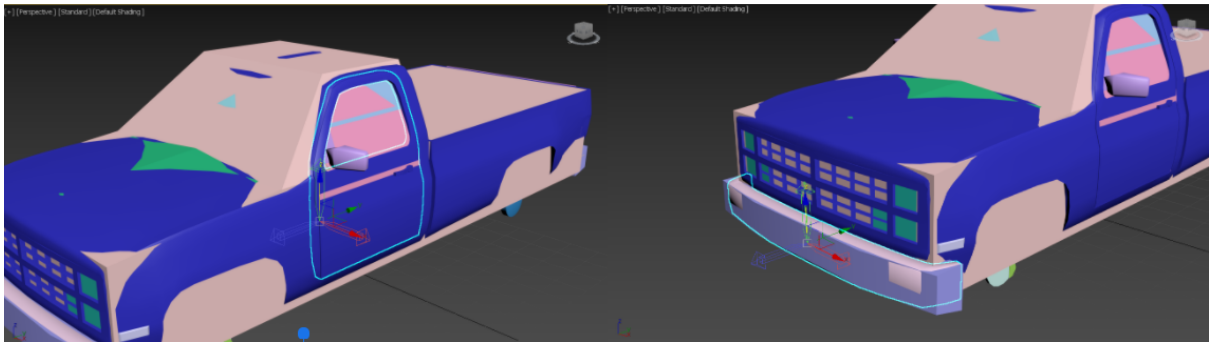
2) The vehicle and all of its parts must have proper pivot orientation, Z axis - (z>0)front/(z<0)back, X axis - (x>0)right/(x<0)left, Y axis - (y>0)up/(y<0)down.



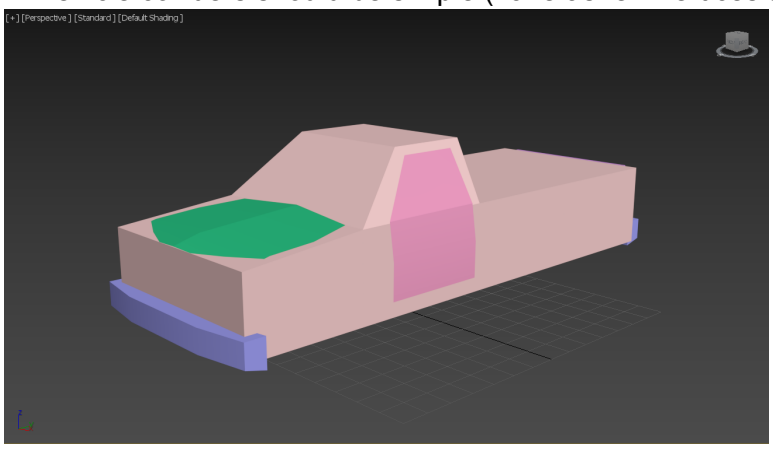
If pivot orientations are wrong there might be difficulties with the wheel determination, detachable parts and some other unexpected problems might appear.

3) Detachable parts should have the correct position of pivot point, for example the door should have pivot point where the door opens (for ease of setup), bumper should have pivot

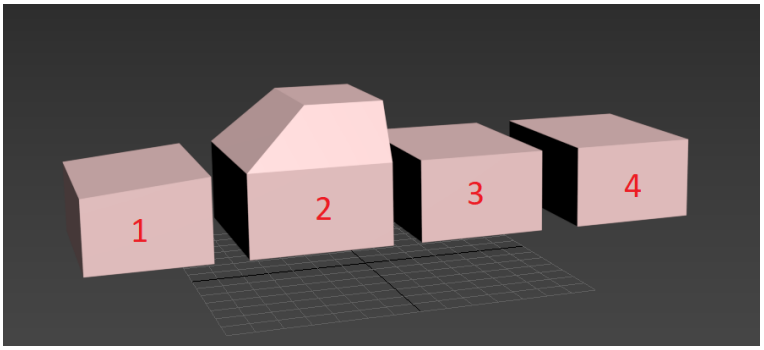
pivot point near Center of Mass (COM) etc.



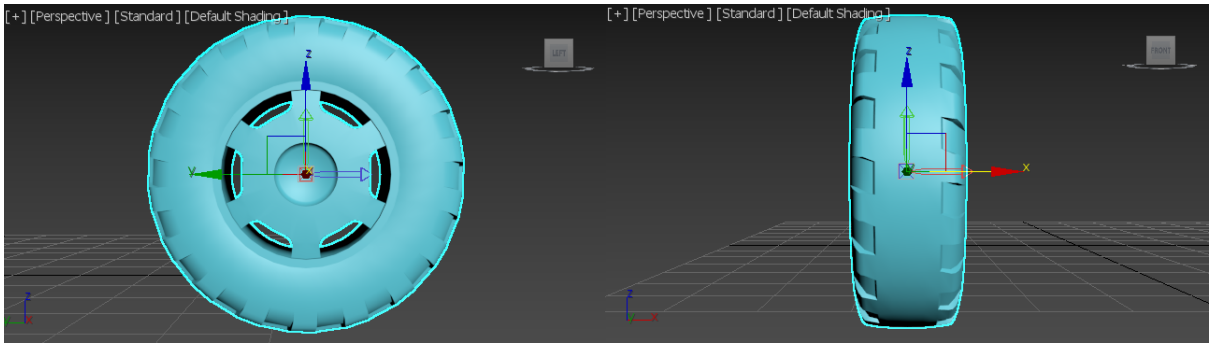
4) All vehicle colliders should be simple (have as few vertices as possible).



5) All colliders must be convex, therefore separate it in places where you don't need it to be convex, for example Body collider:

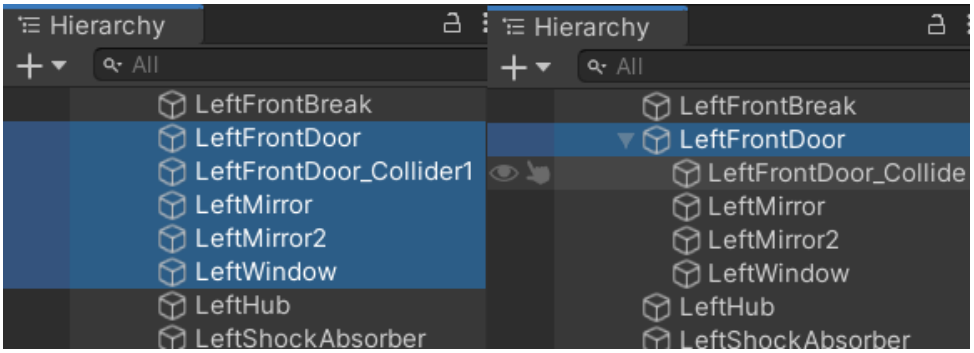


6) Wheel pivot point must be in the center of the wheel, this is where WheelCollider will be placed.

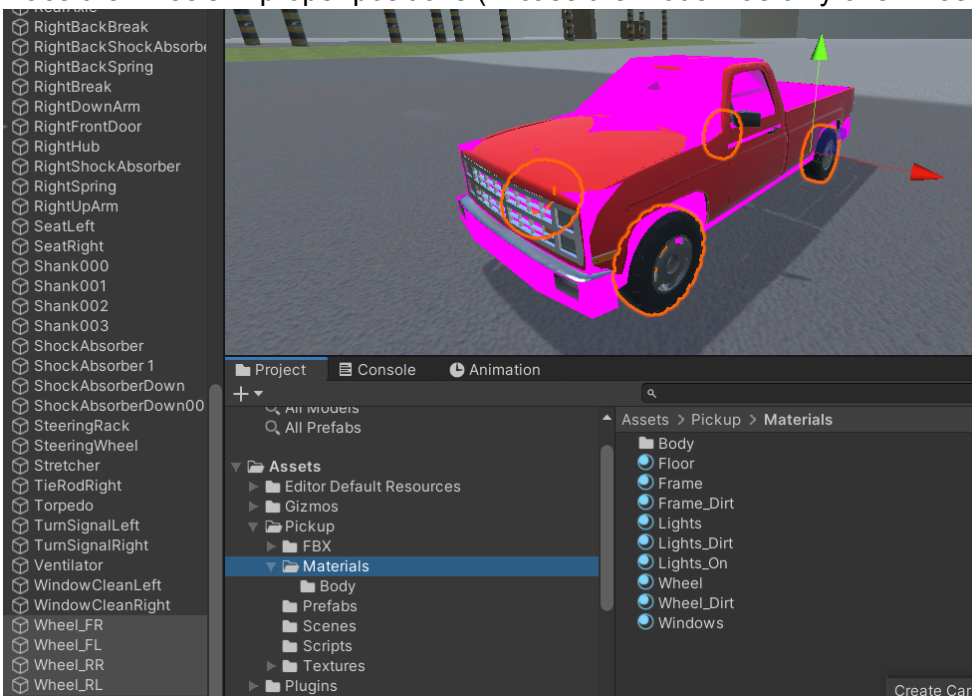


Preparing prefab (Object) for car creation.

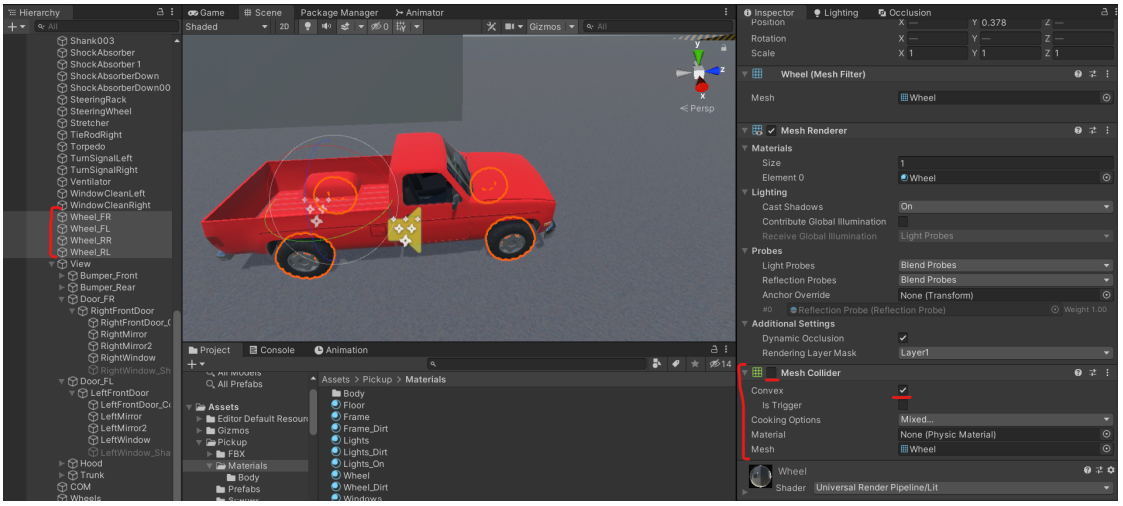
7) For ease of use you can make all parts from one group to be children of that object (for example make mirror and glass children of the door) to not look for them through the whole object hierarchy.



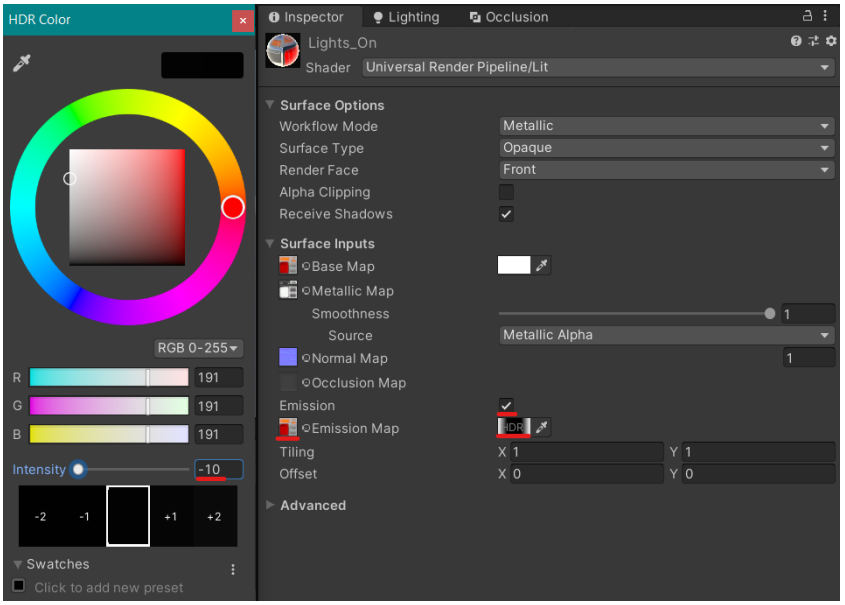
8) Place the wheels in proper positions (in case the model has only one wheel)



9) Add MeshCollider and turn it off (to prevent the wheel from falling through the ground when it's destroyed).



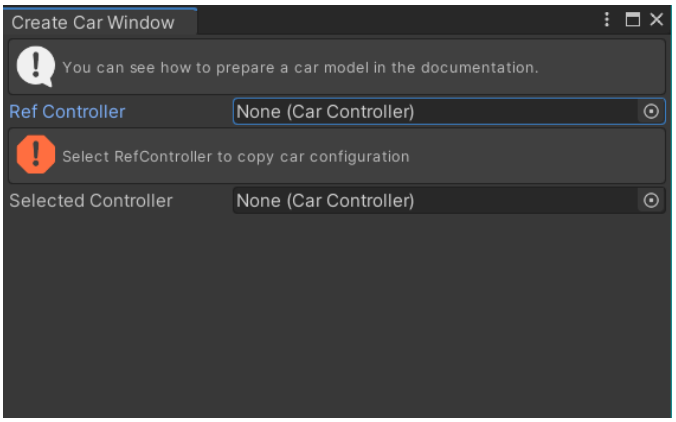
10) Duplicate Lights material and make it Emissive (if you are going to use light for your vehicle).



Create Car Window.

There is a video tutorial covering the whole process [How to create a car in UVC]

To open it from the menu go to: Window > Perfect Games > Create Car Window.

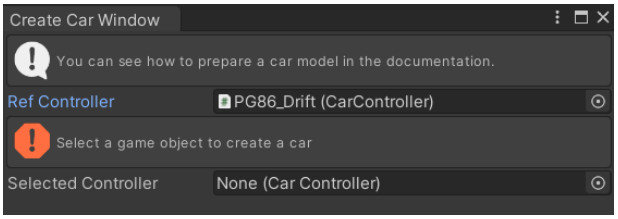


RefController - Reference to existing vehicle from which engine, transmission, steering and wheels configuration will be copied. When creating a new car it is recommended to not leave this field blank.

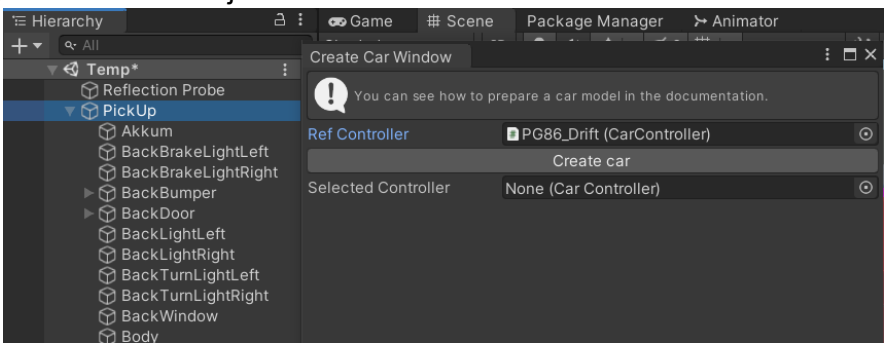
SelectedController - Vehicle that is being edited, automatically assigned when the "Create car" button is pressed, you can also select an existing vehicle to edit.

Car creation:

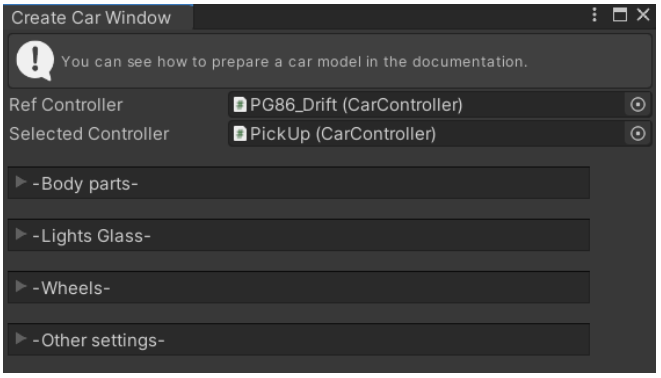
11) Select existing vehicle to copy values from.



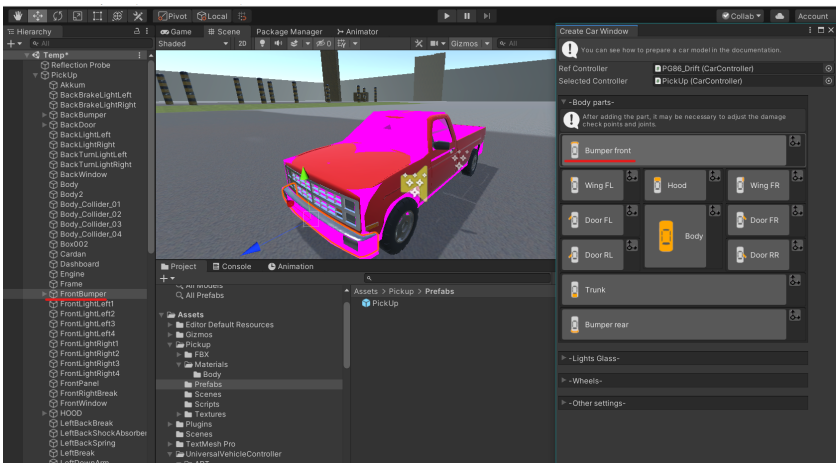
12) Select the object which will be the base for the new vehicle.



13) Click "Create car". After that all necessary components, objects and prefabs will be added to the newly created vehicle.



14) Configure all the vehicle parts ("Body parts" tab), select the object (Objects) and hit the appropriate button.

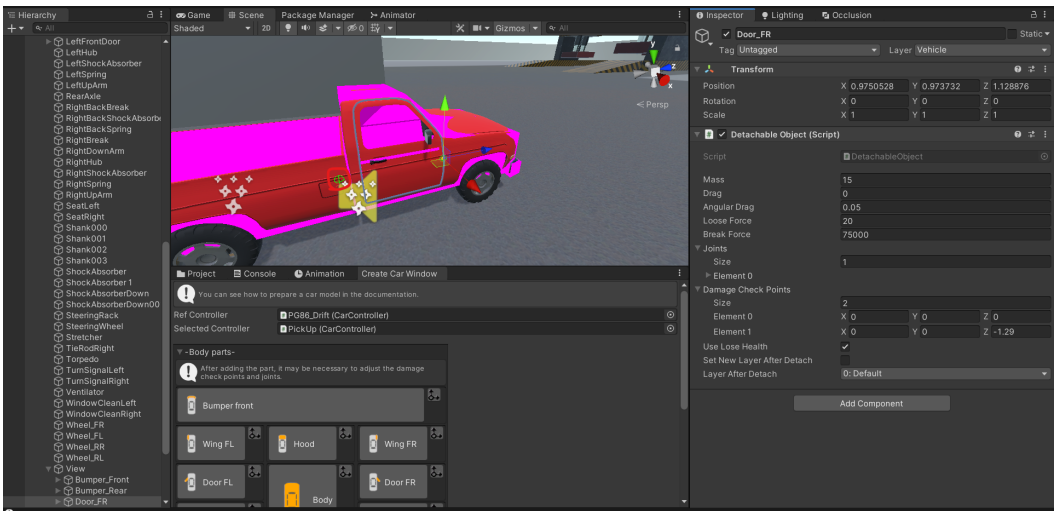


If there are multiple objects - the pivot point might be in the wrong place, to fix that select the object with the right pivot and click  , after that pivot point for that part will be in the selected place.

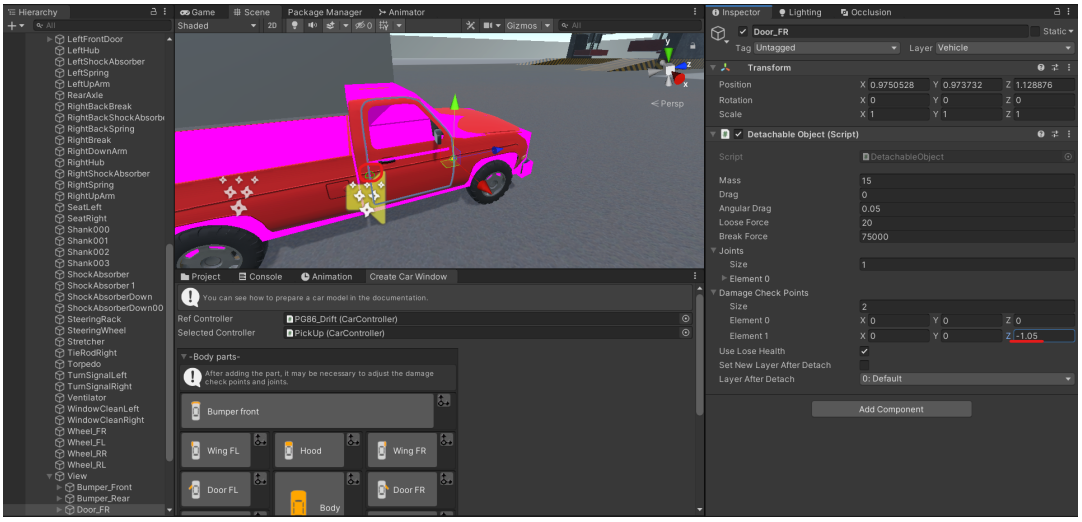
After creation of the part there might be a need to configure DetachableObject DamageCheckPoints and Joints.

Examples:

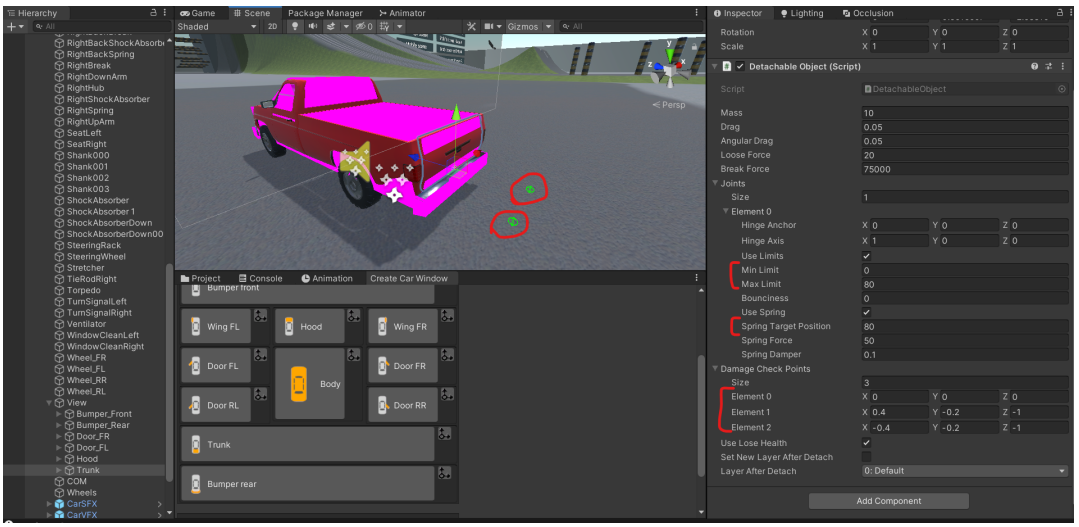
Door after creation



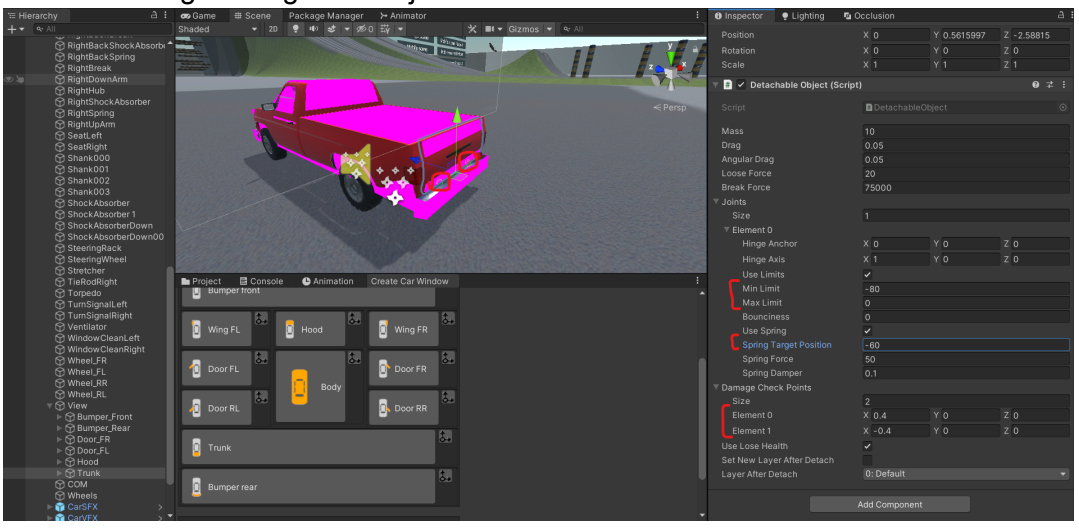
Door after editing DamageableObject



Trunk after creation



Trunk after editing DamageableObject



Similar to this all other components should be created (If the vehicle doesn't have any of them, just skip them).

15) You can select all other parts and hit "Body" button.

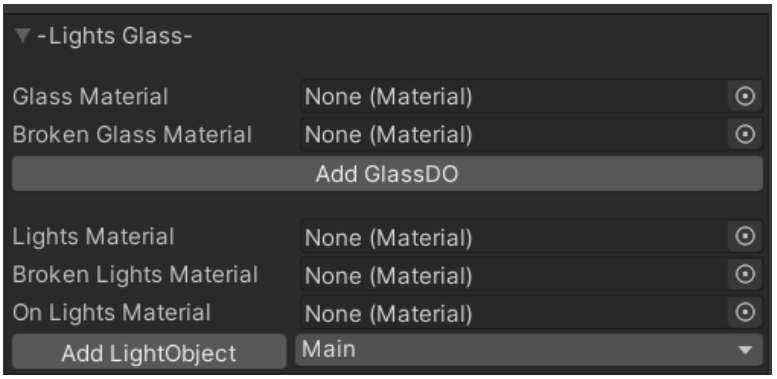
16) If your model has separate meshes for colliders then click on the "Convert all MeshColliders" button, if your collider prefix is different then you need to change the "ColliderPrefix".

!!!IMPORTANT!!! Colliders are required for all vehicles. If your model does not have collider meshes, you can:

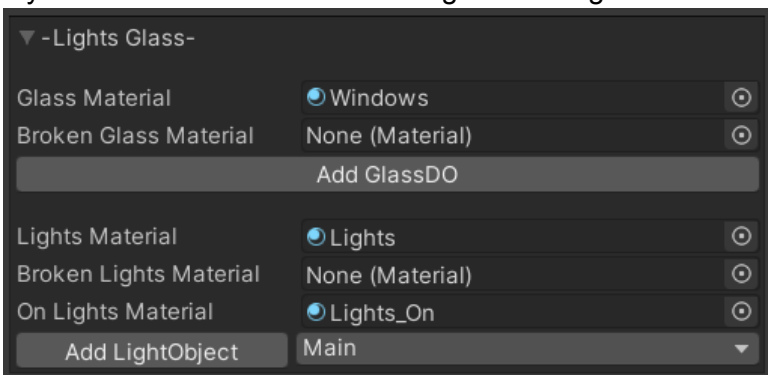
- Try to use the existing mesh of the part (If the mesh has a lot of polygons then there may be a performance problem).
- Use primitive colliders (for example box collider), in this case the collider will not change shape when damaged.

I recommend using meshes specially prepared for colliders (you can see how this is done in standard UVC models).

17) Glass and lights ("Lights Glass" tab).



First you need to select materials for glass and lights.



Broken material is used when object is destroyed, for example glass shards. If no material is provided then the object will disappear after it is destroyed.

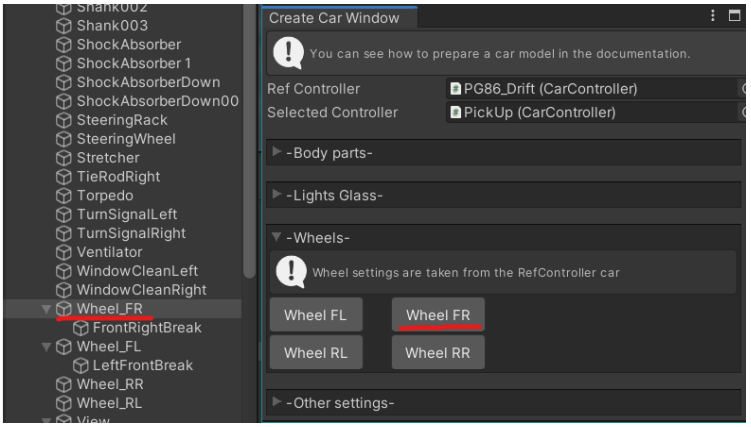
18) Select the glasses and press "Add GlassDO", glass shards particles are created, which will appear when an object is destroyed, you can adjust the angle if needed, by default the angle is based on the pivot point of the vehicle.

19) Like glasses, now select all the lights of the same type and hit "Add LightObject", repeat for each type of lights (Main, Turn left, Turn right, brake, reverse).

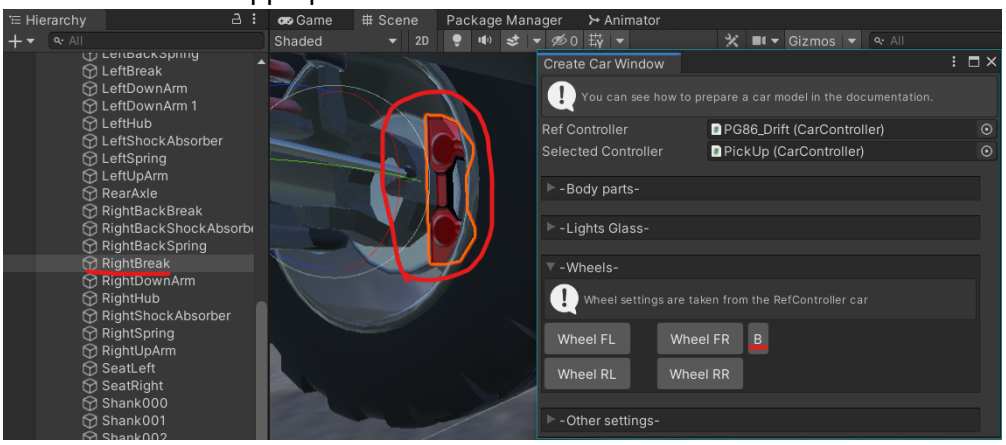
20) After lights are created you can add light sources for selected objects (for headlight or for invisible objects without MeshRenderer).

21) ("Wheels" tab)

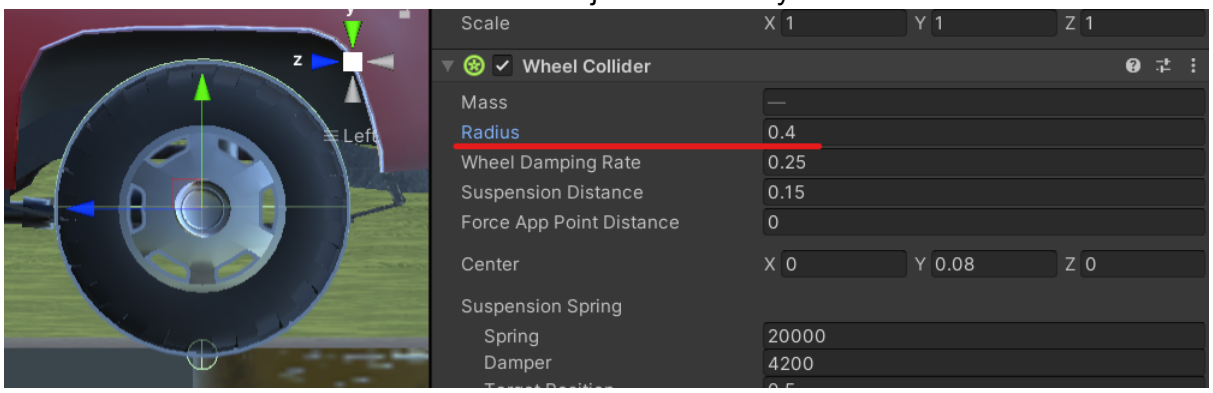
Select the wheel and all of its parts (rim, tire, brake pad), it is important that all of these parts have the same pivot, if it is not so then make all of them children of the wheel and highlight the wheel. Hit the appropriate button, all the settings come from RefController.



After wheel creation there will be option to add brake supports. Select your object and press the "B" button next to the appropriate wheel.

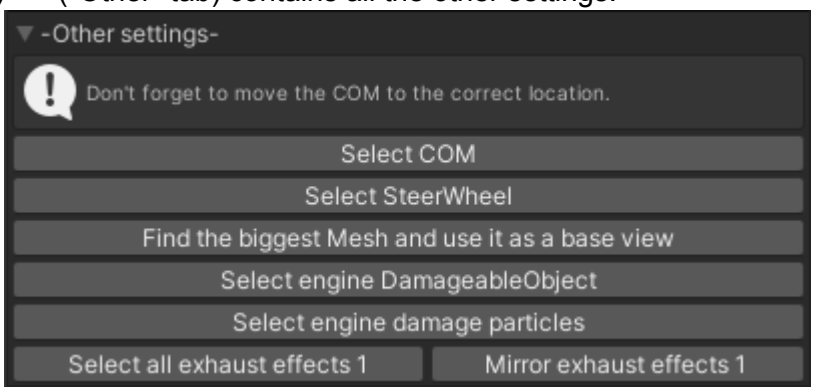


Also wheel radius of WheelCollider should be adjusted if it's any different



Other wheels are configured accordingly.

22) ("Other" tab) contains all the other settings.



"Select COM" button makes the COM object active, position it as you like, this highly affects vehicle behavior, it is suggested to be in the center of the vehicle and slightly lower the wheelbase (see how it is done in other vehicles from the asset).

23) Select the steering wheel (if your model has one) and click "Select SteerWheel", after that it will rotate along the Z axis towards the turn direction.

24) Click "Find the biggest Mesh and use it as a base view", the biggest MeshRenderer will be found and will be added to CarController.BaseViews, you can choose custom view options if needed, it is used to determine the visibility of the vehicle, when it is not visible there are no effects and wheel spin.

25) The "Select engine DamageableObject" button selects or creates in the absence of an object responsible for engine damage. The location of this object determines the strength of damage to the engine during impacts.

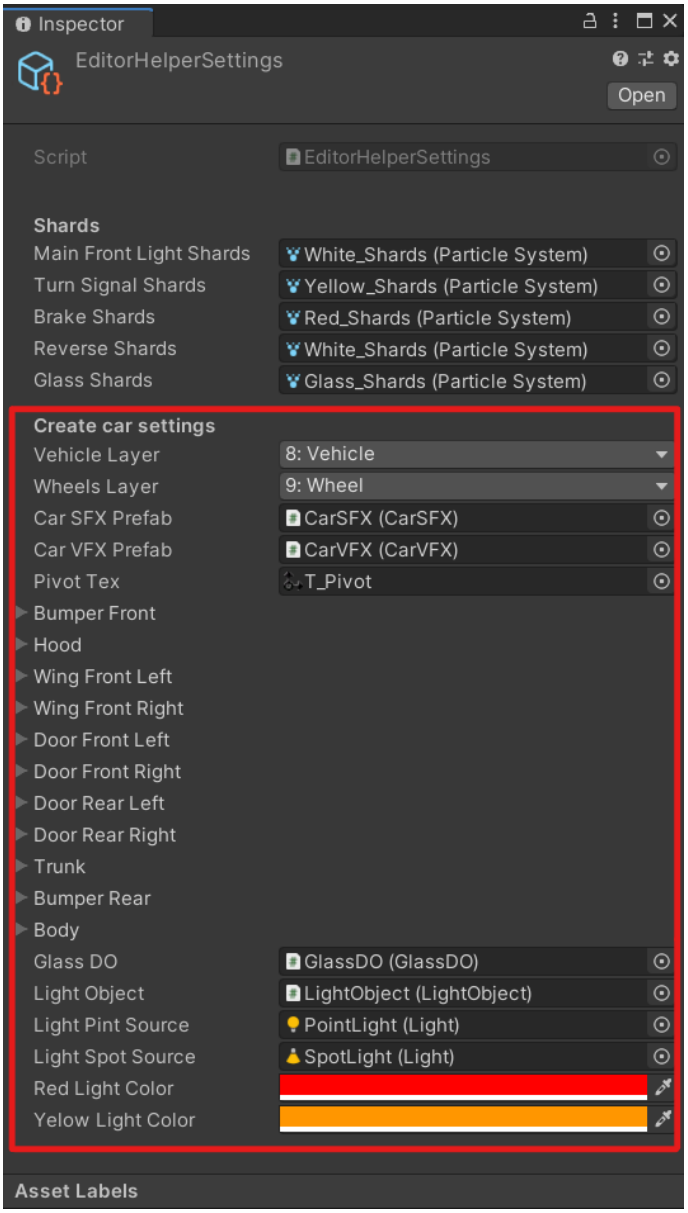
26) The "Select engine damage particles" button selects all engine damage particle systems, if these effects are present in the CarVFX prefab.

27) The "Select all exhaust effects n" button selects all exhaust effects to help you position them.

28) The "Mirror exhaust effects n" button duplicates the effects along the X axis.

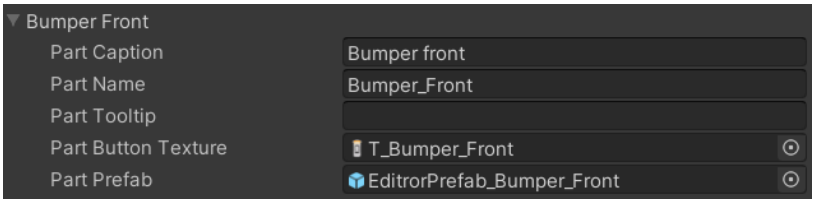
Vehicle creation settings

All setting are in the EditorHelperSettings



- Shards - ParticleSystems for creating lights and glass shards effect. Automatically added upon creation of the lights or glasses
- VehicleLayer - Layer to which the vehicle will be assigned.
- WheelsLayer - Layer to which the wheels will be assigned..
- CarVFXPrefab - Visual Effects prefab for creating lights and glass shards effect. Automatically added upon creation of the vehicle.
- CarSFXPrefab - Sound Effects prefab. Automatically added upon creation of the vehicle.
- PivotTex - Texture used to mark pivot point (Editor only).
- BumperFront (and other parts) - Field of type CarPart, contains the info about the part

CarPart contains fields:



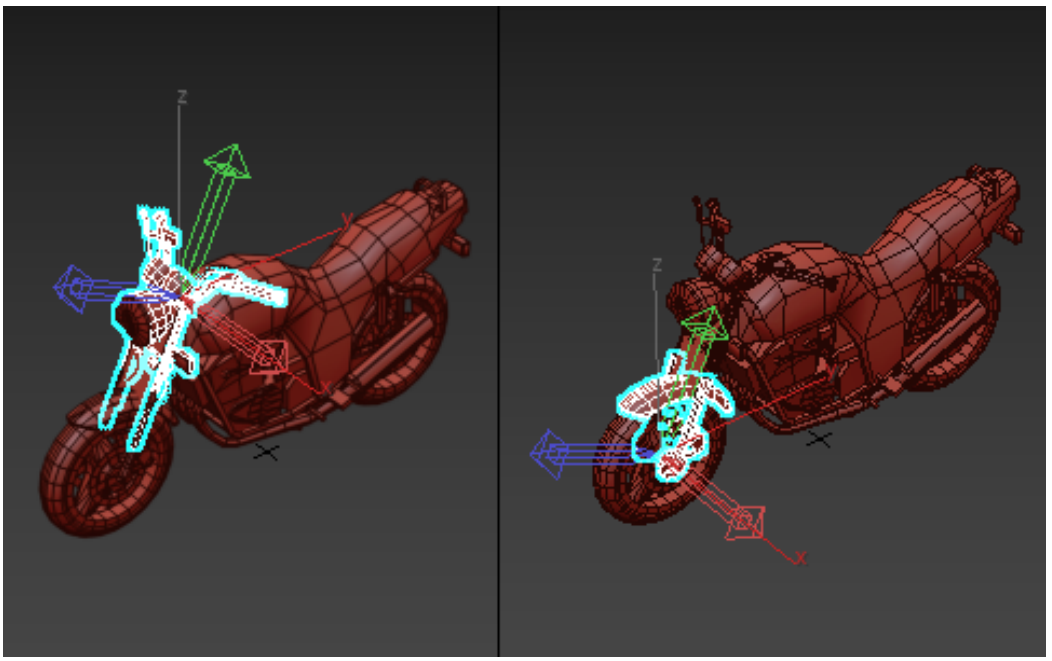
    - PartCaption - Part’s label which will be displayed on the part.
    - PartName - Part’s name, this is the name used in object hierarchy.
    - PartTooltip - Tooltip for the icon.
    - PartButtonTexture - Texture for the button.
    - PartPrefab - Part’s prefab which will be copied to the selected part, all such parts are located in "Assets\UniversalVehicleController\Editor\EditorPrefabs", you can change them and create vehicles with your custom parts.
- GlassDO - Glass settings prefab (Health, sound etc.).
- LightObject - Light settings prefab (Health, sound etc.).
- LightPointSource - Point light prefab;
- LightSpotSource - Spot light prefab (for headlights);
- RedLightColor - Color for the light source, chosen based on the light type (Brake light, rear main light).
- YelowLightColor - Color for the light source, chosen based on the light type (Left blinker, Right blinker).

Bike Creation (CreateBikeWindow)

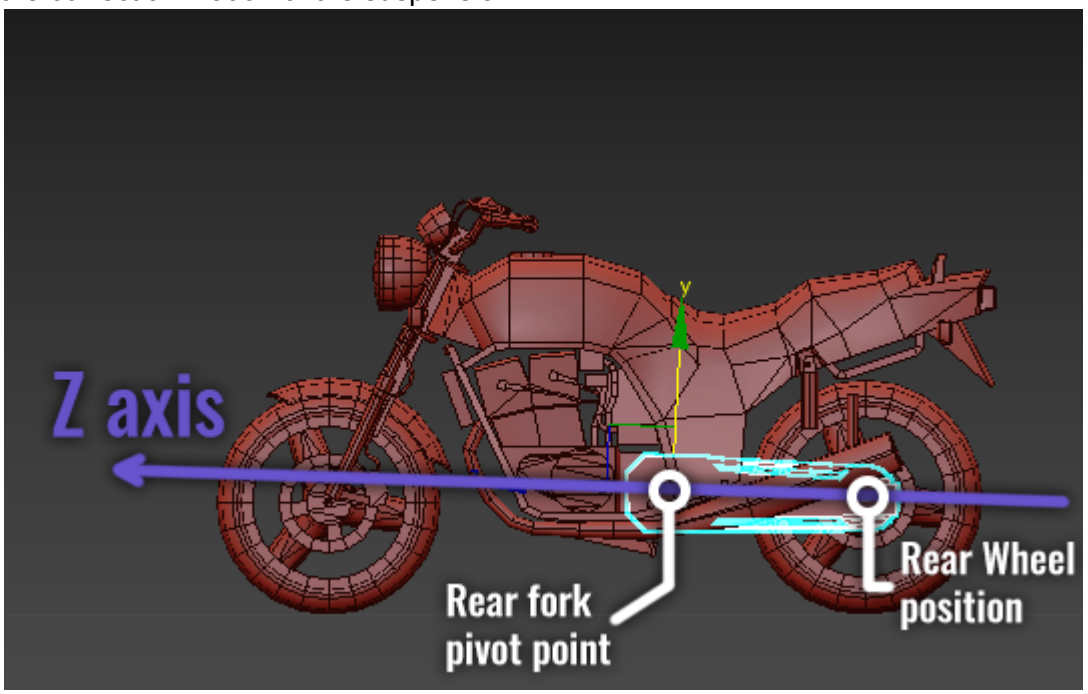
Preparing bike models.

The bike model is configured in the same way as the car model, you can see 1 - 6 in the section "Preparing car models".

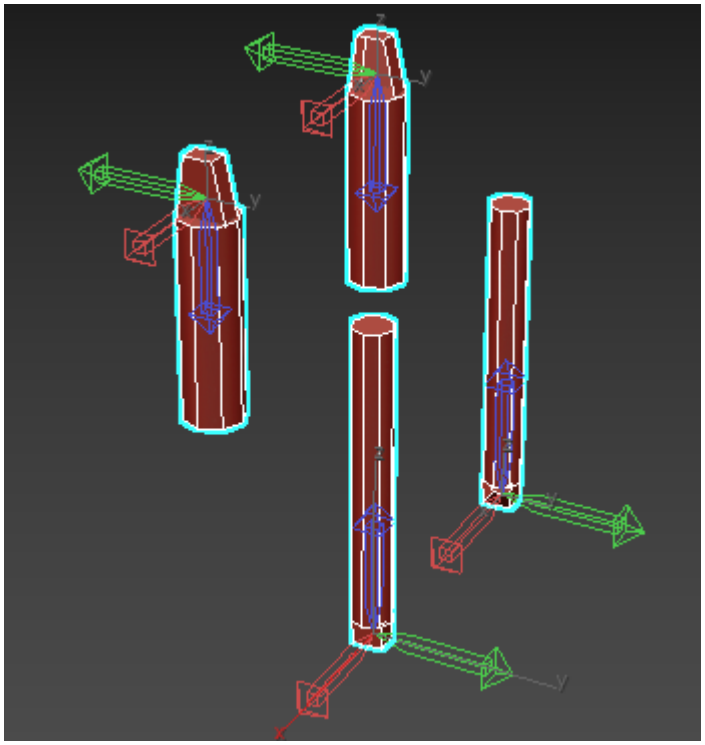
7) The rotation of the pivot point of the handlebar and the front fork must be the same for the suspension to animate correctly. It is desirable that the Y-axis coincides with the handlebar axis, if this is not the case, then for the handlebar axis, you can set the angle manually. It is desirable that the position of the handlebar pivot point should coincide with the steering axis of the bike.



8) The position of the pivot point of the rear fork should be in the axle of the rear fork attachment, the direction of the pivot point should be such that the rear wheel is on the Z-axis of the rear fork for the correct animation of the suspension.

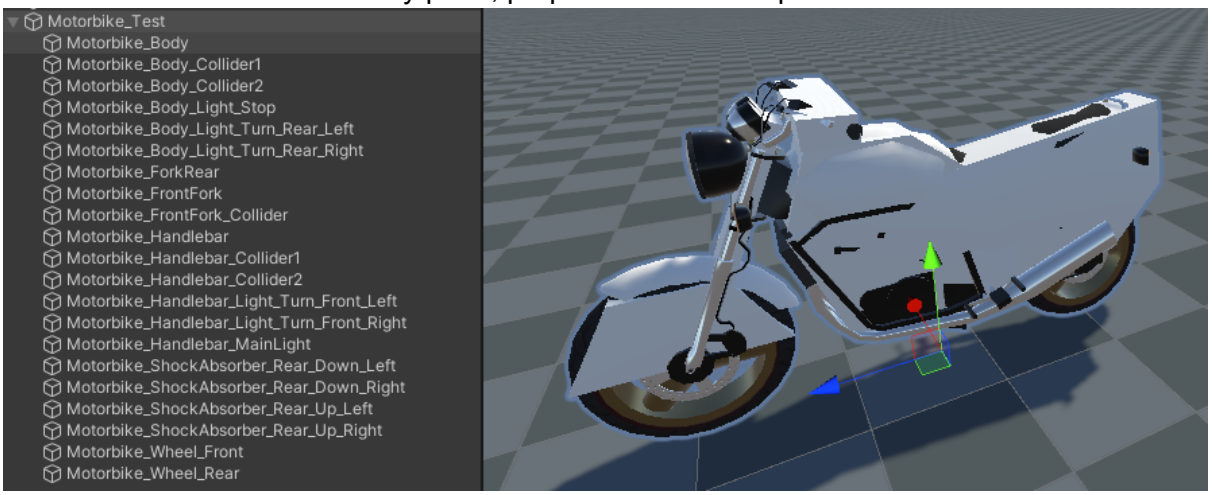


9) If your model has shock absorbers, then the Z-axis of the pivot point must coincide with the shock-absorbers axis, the Z axis should be directed towards the tracking object.



Preparing prefab (Object) for car creation.

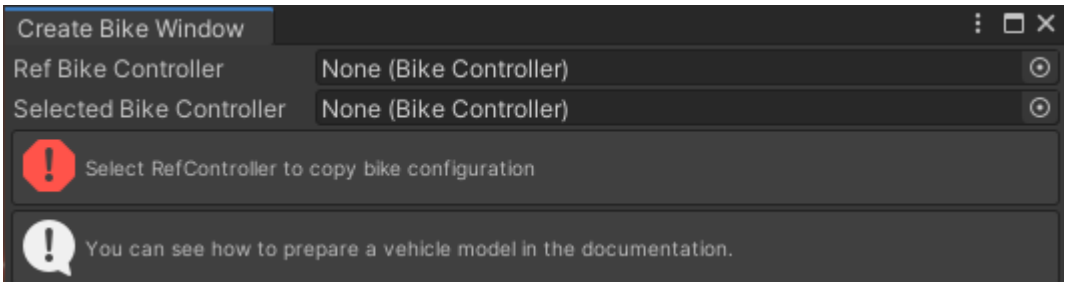
Since the bike does not have many parts, preparation is not required.



Create Bike Window

There is a video tutorial covering the whole process How to create a bike in UVC

To open it from the menu go to: Window > Perfect Games > Create Bike Window.

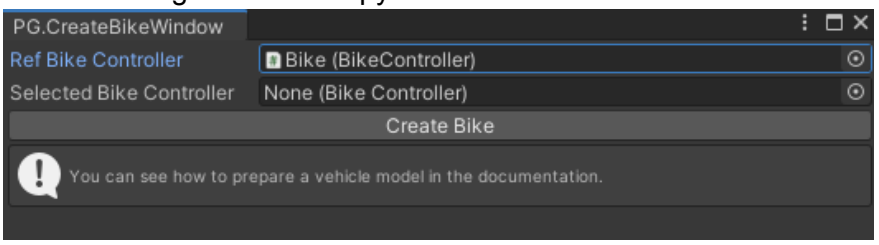


RefBikeController - Reference to existing vehicle from which engine, transmission, steering, wheels and BikeConfig configuration will be copied. When creating a new car it is recommended to not leave this field blank.

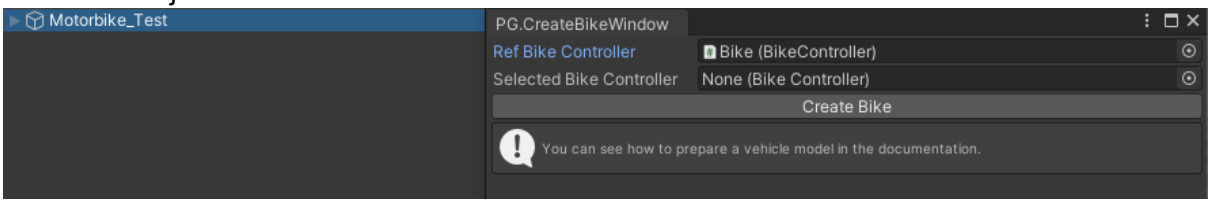
SelectedBikeController - Vehicle that is being edited, automatically assigned when the "Create bike" button is pressed, you can also select an existing vehicle to edit.

Bike creation:

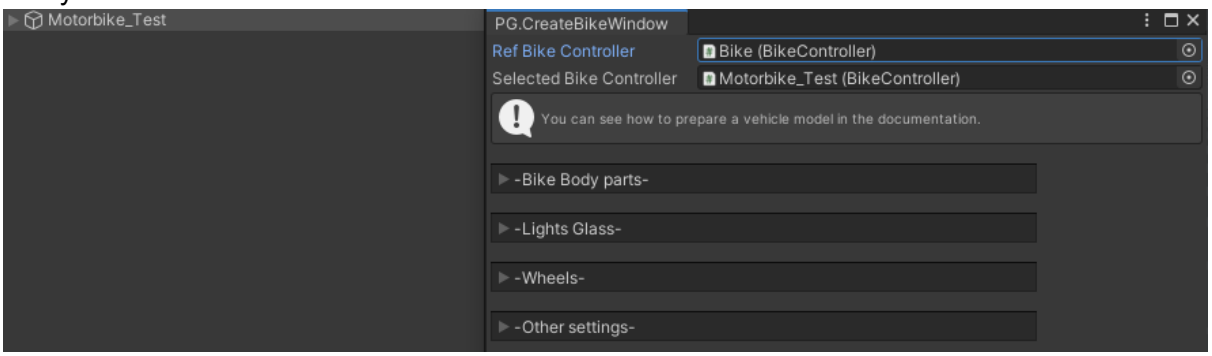
1) Select existing vehicle to copy values from.



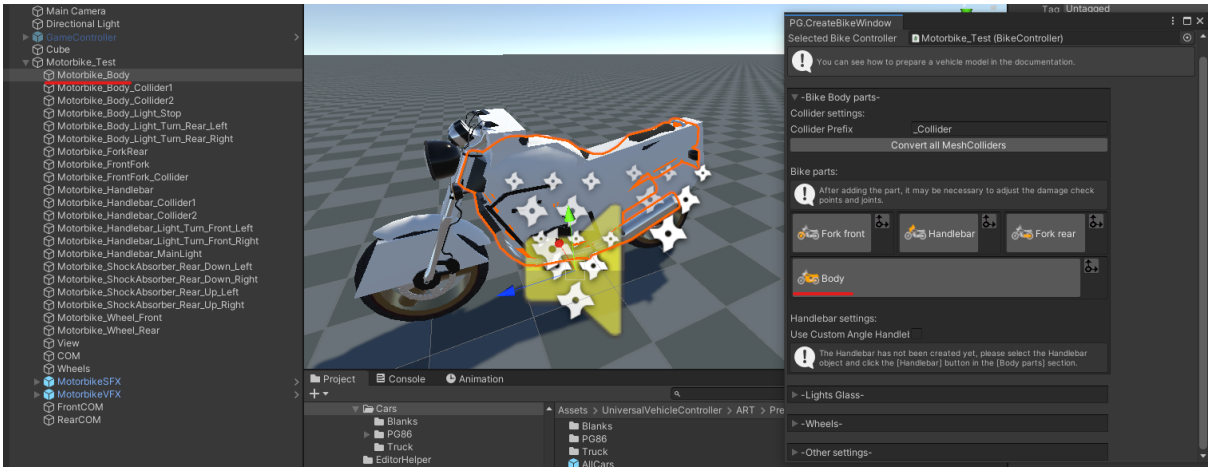
2) Select the object which will be the base for the new vehicle.



3) Click "Create bike". After that all necessary components, objects and prefabs will be added to the newly created vehicle.



4) Configure all the vehicle parts ("Body parts" tab), select the object (Objects) and hit the appropriate button.



If there are multiple objects - the pivot point might be in the wrong place, to fix that select the object with the right pivot and click  , after that pivot point for that part will be in the selected place.

5) If your model has separate meshes for colliders then click on the "Convert all MeshColliders" button, if your collider prefix is different then you need to change the "ColliderPrefix".

!!!IMPORTANT!!! Colliders are required for all vehicles. If your model does not have collider meshes, you can:

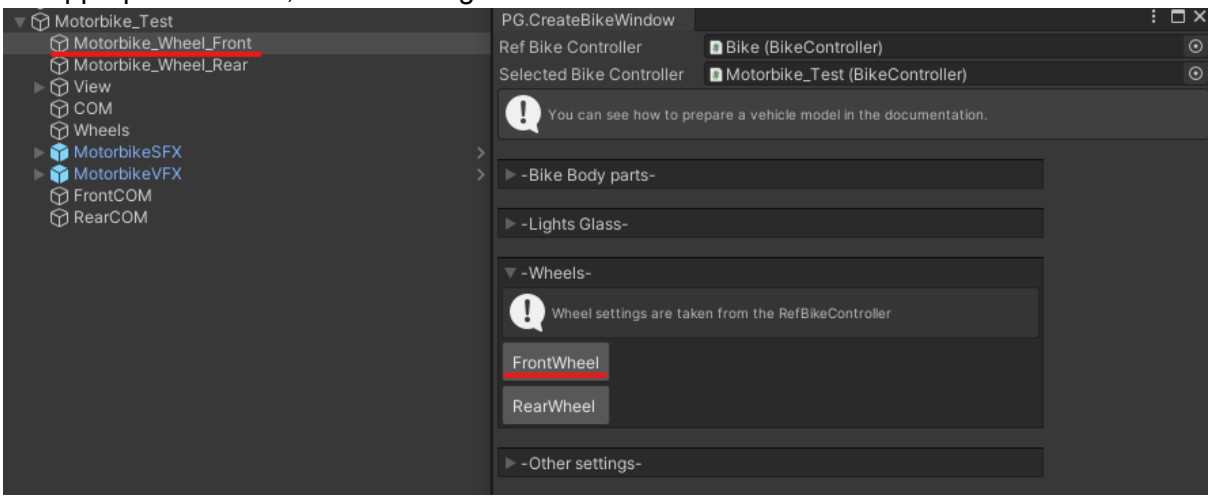
- Try to use the existing mesh of the part (If the mesh has a lot of polygons then there may be a performance problem).
- Use primitive colliders (for example box collider), in this case the collider will not change shape when damaged.

I recommend using meshes specially prepared for colliders (you can see how this is done in standard UVC models).

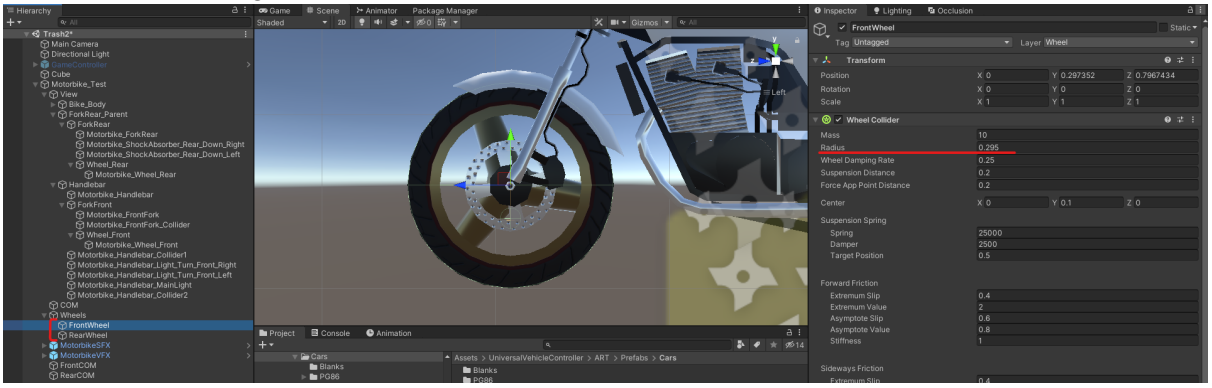
6) Glass and lights ("Lights Glass" tab) can be adjusted in the same way as in a car.

7) ("Wheels" tab)

Select the wheel and all of its parts (rim, tire, brake pad), it is important that all of these parts have the same pivot, if it is not so then make all of them children of the wheel and highlight the wheel. Hit the appropriate button, all the settings come from RefBikeController.



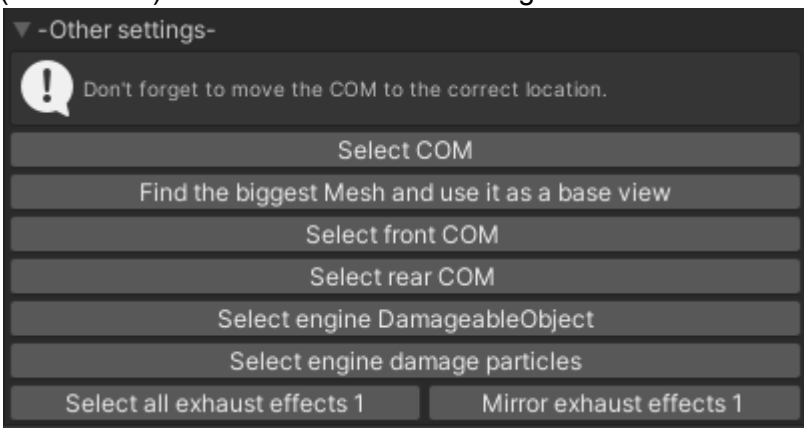
You also need to change the radius of WheelCollider if it is different



!!!IMPORTANT!!! WheelColliders and WheelViews are in different parent objects (To simulate the suspension of a bike).

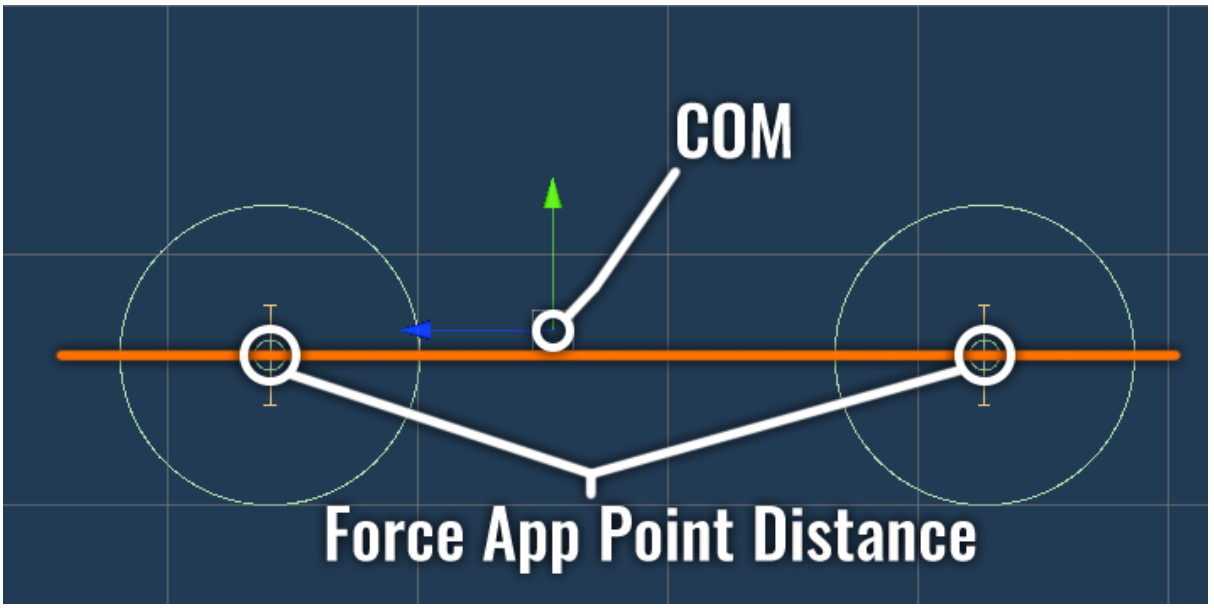
The rear wheel is adjusted in the same way.

8) ("Other" tab) contains all the other settings.



"Select COM" button makes the COM object active, position it as you like, this highly affects vehicle behavior, it is suggested to be in the center of the vehicle and slightly lower the wheelbase (see how it is done in other vehicles from the asset).

!!!Attention!!! The COM position greatly affects the behavior of the bike. The closer the COM position is to the line between the ForceAppPointDistance of the wheels, the more stable the bike behaves when changing RB.Velocity (Acceleration, braking, turns).



9) Click "Find the biggest Mesh and use it as a base view", the biggest MeshRenderer will be found and will be added to CarController.BaseViews, you can choose custom view options if needed, it is used to determine the visibility of the vehicle, when it is not visible there are no effects and wheel spin.

10) The "Select engine DamageableObject" button selects or creates in the absence of an object responsible for engine damage. The location of this object determines the strength of damage to the engine during impacts.

11) The "Select engine damage particles" button selects all engine damage particle systems, if these effects are present in the CarVFX prefab.

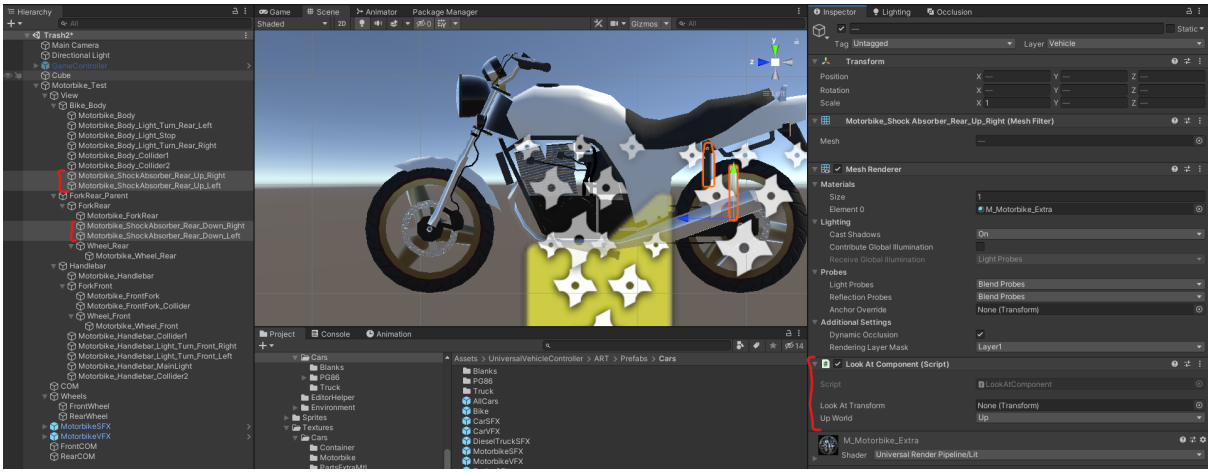
12) The "Select all exhaust effects n" button selects all exhaust effects to help you position them.

13) The "Mirror exhaust effects n" button duplicates the effects along the X axis.

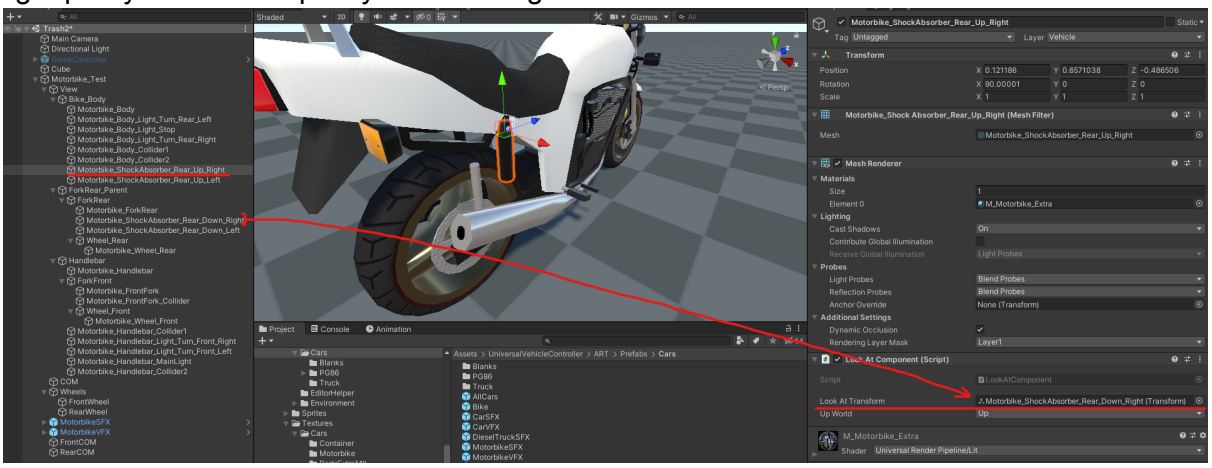
LookAtComponent

LookAtComponent has been added to animate the bike suspension (Shock absorbers). Correct operation of the shock absorber (Correct positioning) depends on the correct setting of the shock absorber pivots, how to set up correctly is described in the section "Preparing bike model".

1) If your bike has shock absorbers, select all shock absorber parts (Upper / Lower) and add this component to them.

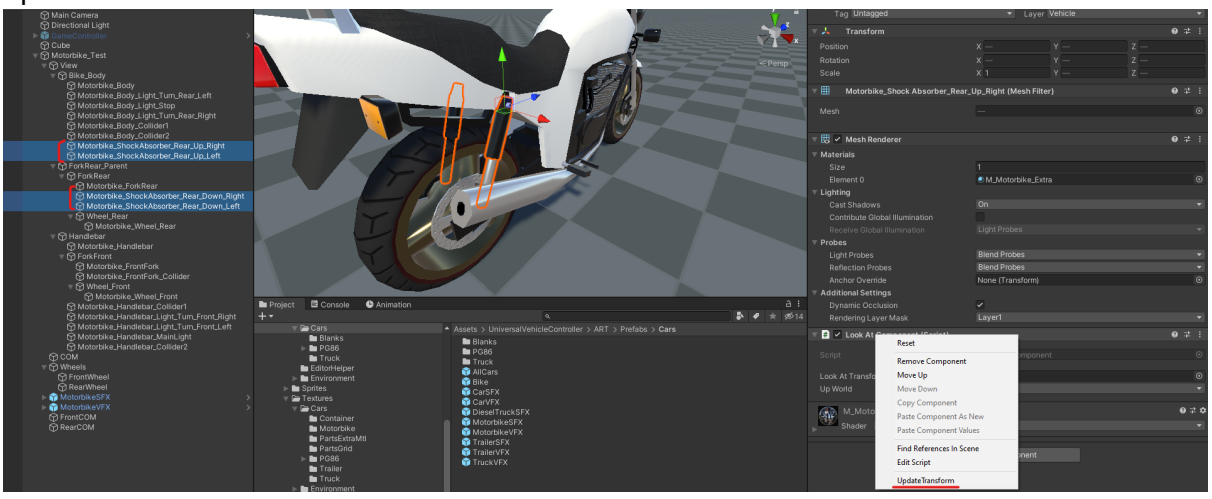


2) In LookAtTransform specify the opposite part of the shock absorber, for example: for the upper right part you need to specify the lower right.



3) In some cases it is necessary to change UpWorld, for example for this bike you need to change UpWorld to Down for the upper parts of the shock absorbers.

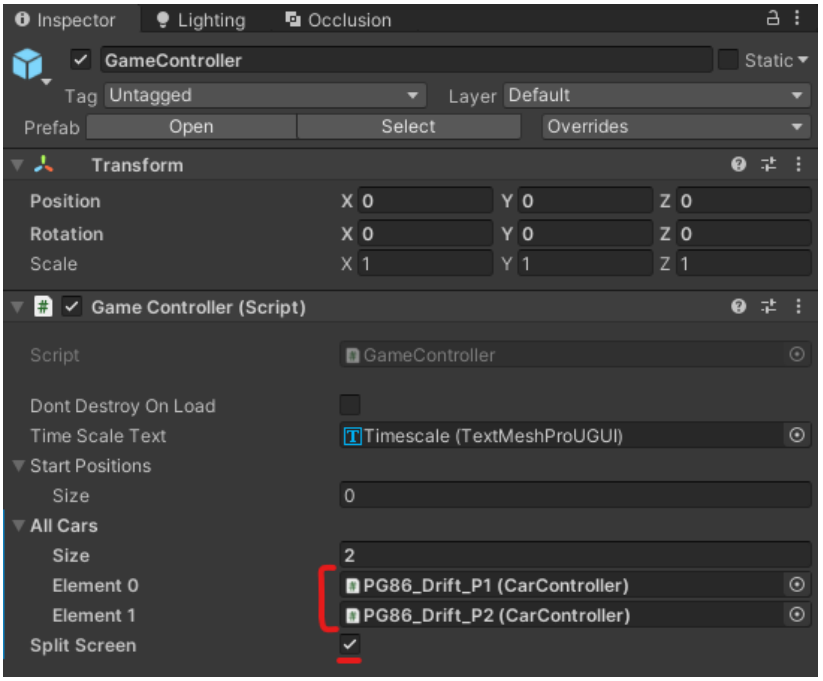
4) After adjusting all parts of the shock absorbers, you can execute the "Context Menu / UpdateTransform" command.



If the setting is correct, all the parts will be in the correct position.

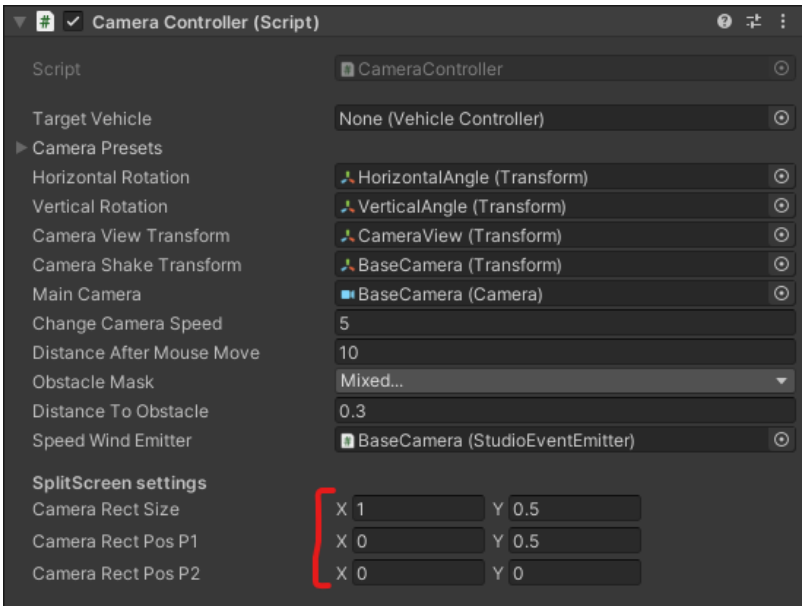
Local multiplayer (Split screen)

The asset can be used for split-screen gaming. To do this, you need to install the add-ons FMOD (For the ability to use two AudioListiner), InputSystem (For convenient separation of control), set the GameController.SplitScreen flag and place 2 cars in the AllCars list, the 1st car will be for the first player, the 2nd car for the second player.



Split screen settings:

Camera is set to split horizontally



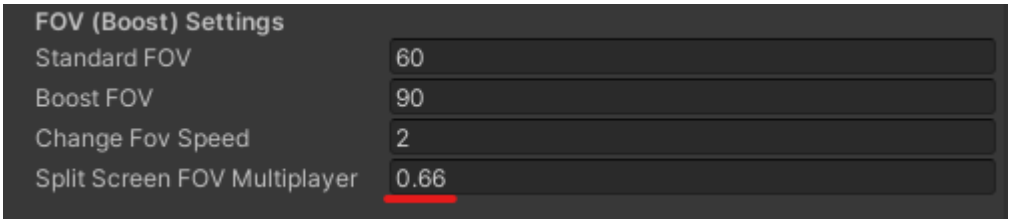
If desired, you can make vertical split by replacing the values with:

CameraRectSize = 0.5, 1

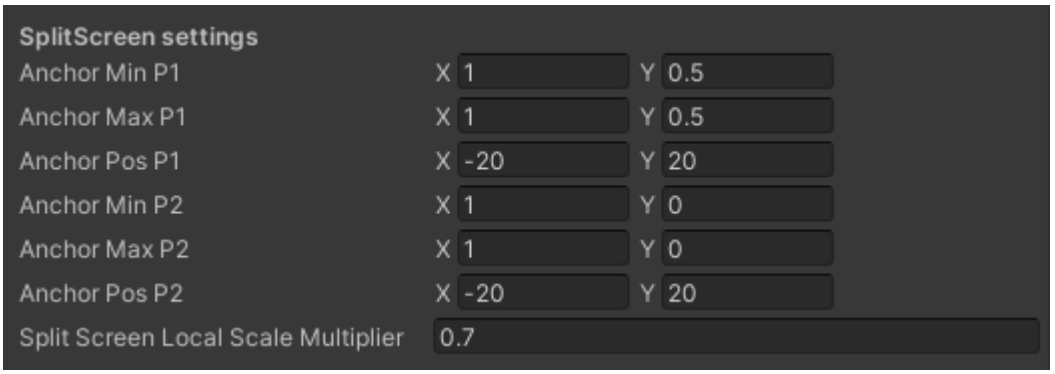
CameraRectPosP1 = 0, 0

CameraRectPosP2 = 0.5, 0

You can also adjust the FOV multiplier for the split screen in the camera presets (so that the image is not stretched at the edges).



CarStateUI is set to split horizontally (CarStateUI is in the PlayerController prefab)



If desired, you can make vertical split by replacing the values with:

AnchorMinP1 = 0.5, 0

AnchorMaxP1 = 0.5, 0

The cars are controlled using one keyboard (the main keyboard of the first player and the NUM block for the second player), you can also select gamepads for different players in the menu.

Mobile version

The asset can be used to develop mobile projects, all scripts are well optimized and work even on very low-end devices.

The main GPU and CPU resources are used for graphics quality and content (Models, textures, particles, etc.).

A few tips for optimization:

- Avoid using Terrain unless it is really needed. Even with low terrain quality settings, performance is heavily affected on welow-endak devices.
- In models, use as few polygons and detachable parts as possible, this will speed up the calculation of vehicle damage and reduce the load on the GPU.
- I also want to advise you to study the manual on optimization, for example this, or similar.

In the asset, mobile control is carried out using the "CustomButton" buttons.

The control UI is located in the "MobileUI" prefab. Prefab "MobileUI" is a nested prefab in "GameController", object "MobileUI", enabled / disabled in the "Start" method if the application is launched on a mobile device or TargetPlatform == Android | iOS. All control types are located in the MobileUI / Controls prefab.

Also, another PlayerController is created on mobile devices, it differs from the main one only in the position of CarState and camera settings, in the future there may be other differences.

"CustomButton" - This is the successor to "Button" with press testing.

Vehicles in the project

You can easily create your own vehicle, video on how create car, video on how create bike.

There are 3 car variations and 1 motorbike in the project:

1. Stock PG86_Stock – Slow, hard to control
2. Drift PG86_Drift – Relatively fast with low max speed, easy to control, easy to drift
3. Racing S34_Race – Fast with high max speed, easy to control on high speeds
4. For a track with vertical g-forces S34_Race_ForCrazyroad, Vehicle with strong suspension and modified wheel friction settings
5. GMS_4x4 - SUV.
6. Motorbike - Motorbike with 2 wheels.
7. Truck – heavy, has 4 drive wheels and 2 steering wheels, has high friction wheels, engine and transmission are configured for very high torque but relatively low speed
8. Trailer – truck trailer, very heavy, has 6 uncontrollable wheels (only braking is working)

Differences in PG86, which you can compare:

- All WheelColliders, spring force, size, damper, friction
- RigidBody – parameter Drag, for achieving high speeds
- CarController:
    - CarController.COM – center of mass
    - Engine.MaxMotorTorque – max engine torque
    - Steer.MaxSteerAngle – steering angle of the wheels
    - Steer.SteerChangeSpeedToVelocity and SteerChangeSpeedFromVelocity – speed of turning the steering wheels
    - Steer.HelpDriftIntensity – intensity of steering towards drifting
    - Steer.PositiveChangeIntensity and OppositeChangeIntensity – intensity of changing the AngularVelocity
    - GearBox.GearsRatio and MainRatio – gears ratio

You can change other parameters like drive wheels, mass, torque curve, speed of shifting and many more. Next comes the balance. The proper balance heavily affects the feeling from the car

Contacts

If you have any questions regarding the project you are free to contact me in any suitable way.

E-mail: PerfectGamesStudio@gmail.com

Discord: Discord server