

Realistic Car Controller V3.1 by

BoneCracker Games

First of all, thank you for purchasing and using Realistic Car Controller! This documentation will guide you to;

Contents

First to Do!	3
General Information about RCC	4
RCC Settings	6
Configurable Ground Materials	7
Creating New Vehicles	8
Warning	8
Important	8
Collider Shapes Of The Vehicle	12
Driving Assistances	14
Mobile Controllers	14
Mobile Controller (With Unity UI)	15
Mobile Controller (With NGUI)	16
About Mobile Usement on City Scene	17
Dashboard Configuration	18
Damage (Based on Mesh Deformation)	19
Creating Lights, Sounds, Skidmarks, Smoke Effects	19
Variable Ground Tire Grip	20
Adjusting Ground Particles, Wheel Sounds, Damp, Forward and Sideway Stiffness, Slip On Different Grounds	22

RCC Camera System	22
RCC_Customization	24
How The Customization Panel Works	24
AI Configuration	26
Creating NavMesh For Scene	26
Adding AI Controller To Vehicle	28
Adding Waypoints Container To Scene	29
Adding BrakeZones Container To Scene	30
Enter-Exit System	30
F.A.Q.	31
Multiplayer with Photon	32
Credits	35
Extreme Vehicle Pack by Vertigo Games	35
Sofie With Animations by 3DMaesen	35
Sound Effects	35
License	35

You can find more updated details on

<http://www.bonecrackergames.com/realistic-car-controller>

YouTube Playlist

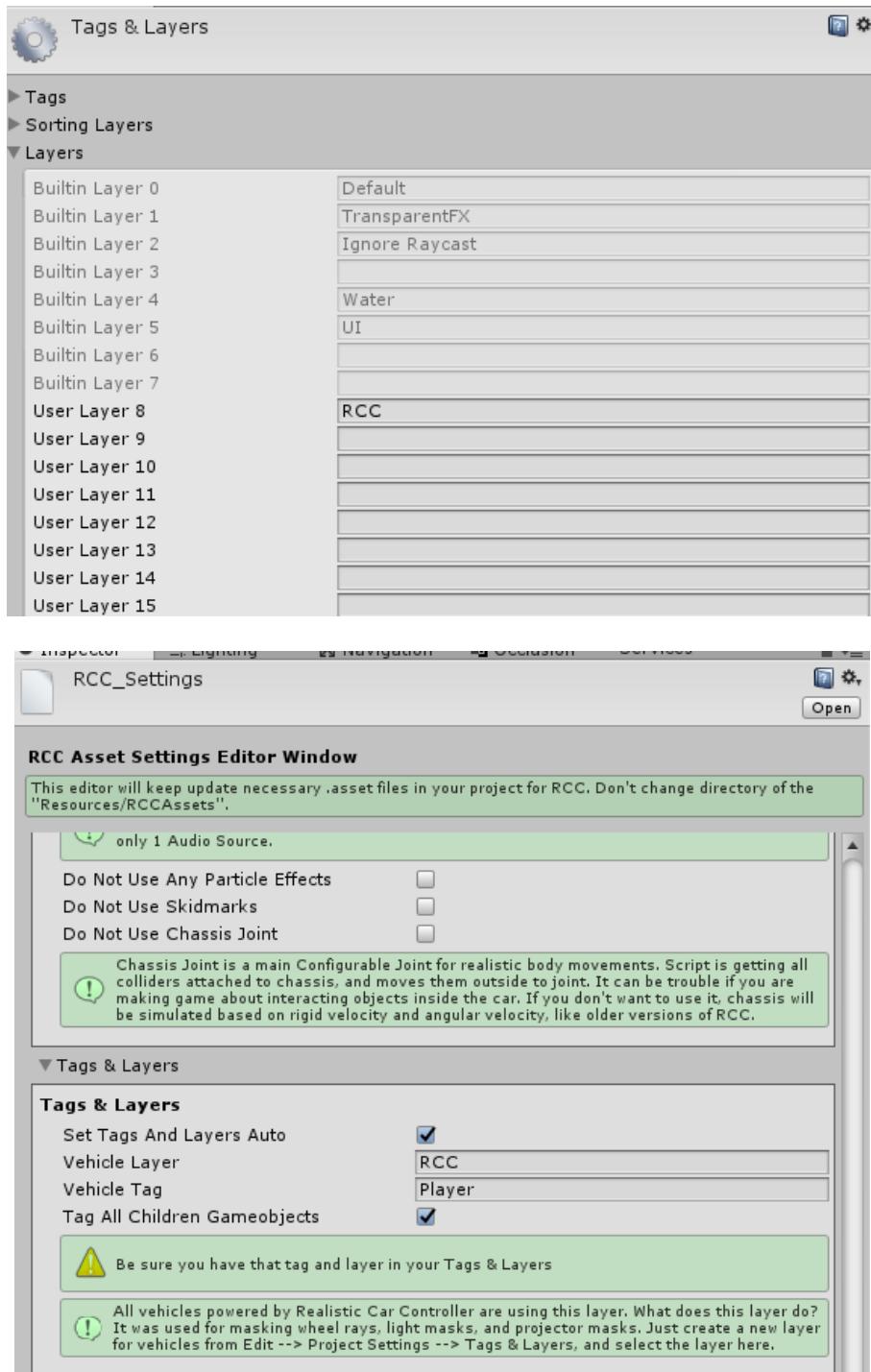
<https://www.youtube.com/playlist?list=PLRXTqAVrLDpoW58IKf8XA1AWD6kDkoKb1>

(You can zoom in with CTRL + ScrollUp for enlarge PDF pages)

First to Do!

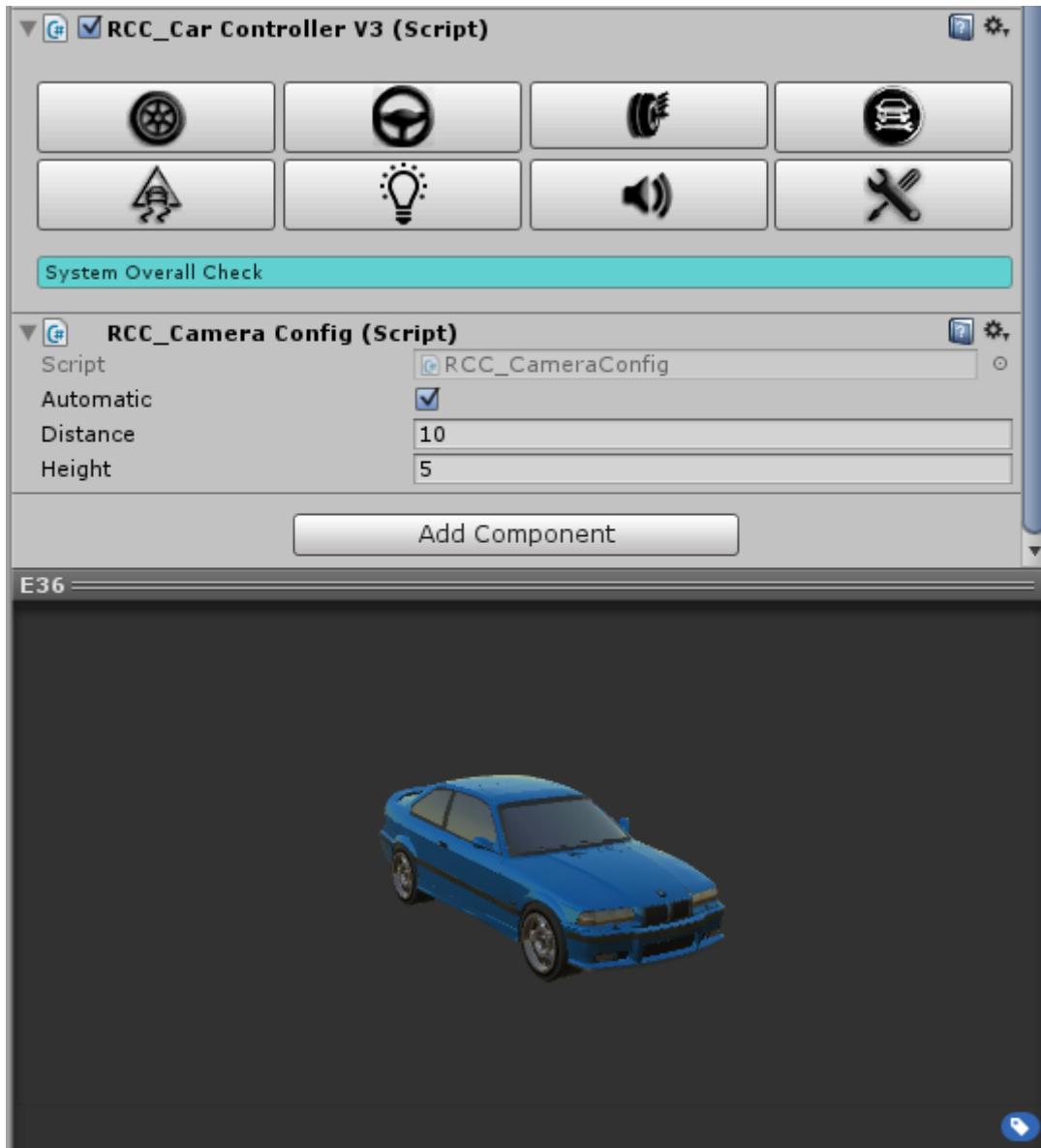
Always backup your project before the update.

This version of the RCC is using **LayerMask** for avoiding unwanted raycast hits and ignoring unnecessary projector layers. Just create one layer for vehicles, and select it in **RCC Settings**.



General Information about RCC

Each vehicle has own [RCC_CarControllerV3.cs](#) script. Each vehicle is responsible for own [RCC_CarControllerV3.cs](#).



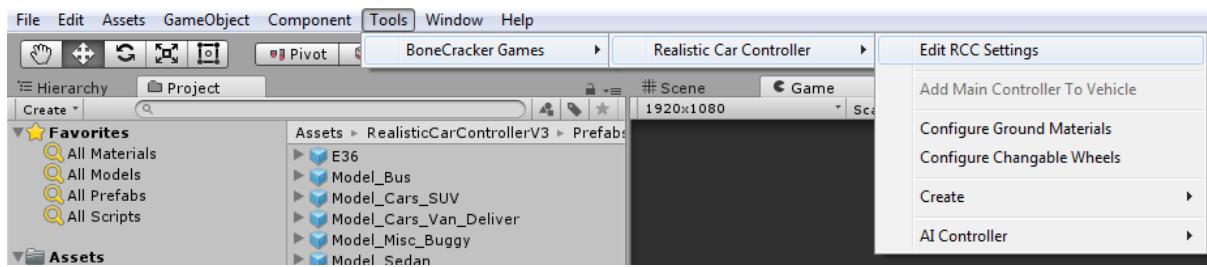
8 Main Categories for Easily and Understandable Creating-Configuring Vehicles.

[Wheels](#), [Steering](#), [Suspensions](#), [Mechanic Configuration](#), [Stability](#), [Lights](#), [Sounds](#), and [Damage](#).

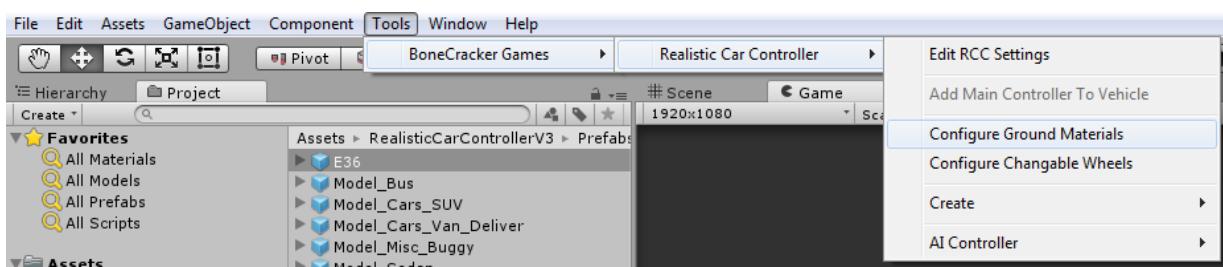
All vehicles are sharing global settings, sounds, configurations via [RCC Settings](#).

Main RCC Settings. It's shared by all vehicles powered by RCC. [Tools → BoneCracker Games →](#)

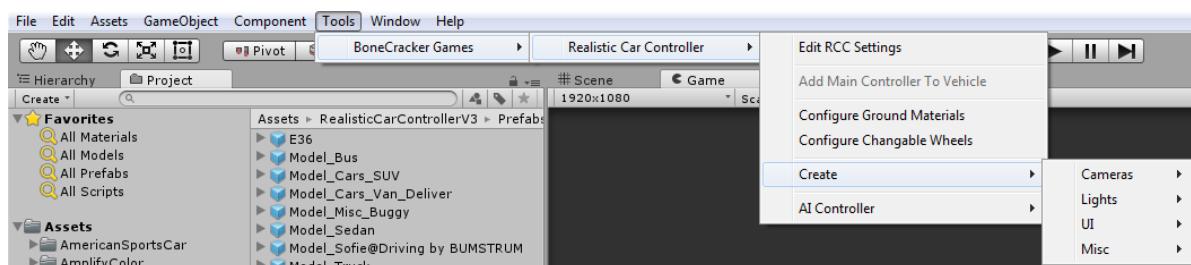
[Realistic Car Controller → Edit RCC Settings.](#)



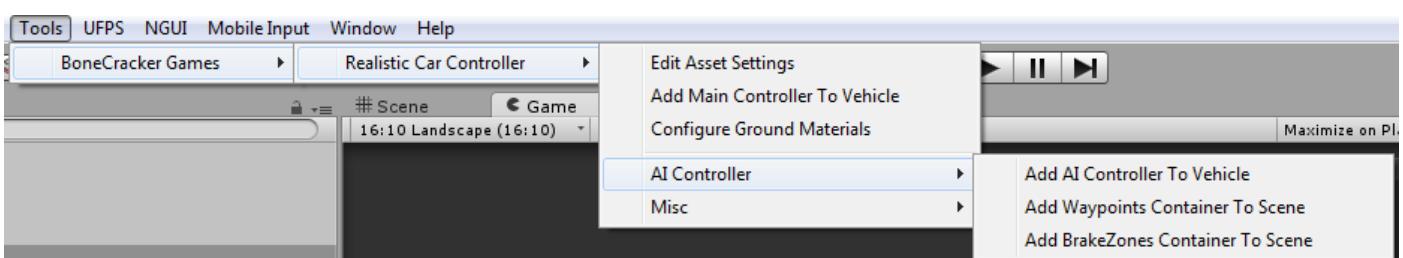
Changing ground materials physics, particles, damps, sounds, etc in [Tools → BoneCracker Games → Realistic Car Controller → Configure Ground Materials.](#)



Creating lights, exhausts, mirrors, enter-exit system, etc in [Tools → BoneCracker Games → Realistic Car Controller → Create.](#)



Making vehicles controlled by AI in [Tools → BoneCracker Games → Realistic Car Controller → AI Controller.](#)



RCC Settings

Main RCC Settings. It's shared by all vehicles powered by RCC. Tools → BoneCracker Games → Realistic Car Controller → RCC Settings.

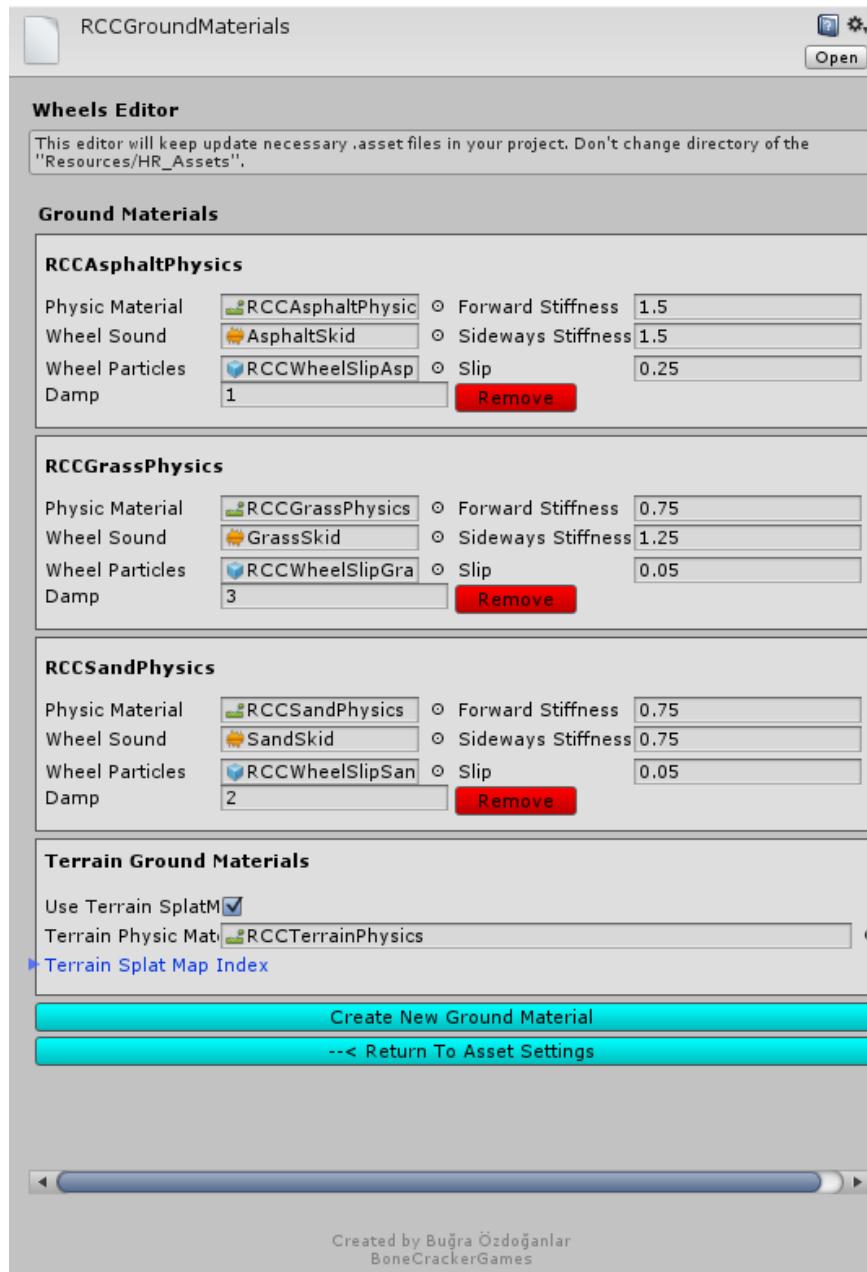
The RCC Asset Settings Editor Window contains several sections:

- General Settings** (Left):
 - Override FixedTimeStep:
 - Fixed Timestep: 0.02
 - Maximum Angular Velocity: 8
 - Behavior Type: Custom
 - Using behavior preset will override wheelcollider settings, chassis joint, antirolls, and other stuff. Using "Custom" mode will not override anything.
 - Use Fixed WheelColliders:
- Controller Settings** (Left):
 - Main Controller Type: Keyboard (selected), Mobile, Custom
 - Keyboard Settings**:
 - Gas/Reverse Input Axis: Vertical
 - Steering Input Axis: Horizontal
 - You can edit your vertical and horizontal input axis in Edit --> Project Settings --> Input.
 - Start/Stop Engine Key: I
 - Low Beam Headlights: L
 - High Beam Headlights: K
 - Change Camera: C
 - Indicator Right: E
 - Indicator Left: Q
 - Indicator Hazard: Z
 - Gear Shift Up: Left Shift
 - Gear Shift Down: Left Control
 - Boost: F
 - Slow Motion: G
 - Get In & Get Out Of The Vehicle: Return
- Main Controller Settings** (Right):
 - Units: KMH
 - Use Automatic Gear:
 - Engines Are Running At Awake:
 - Keep Engines Alive When Player Get In:
 - Auto Reverse:
 - Auto Reset:
 - Contact Particles On Collision: Contact Sparks
- UI Settings** (Right):
 - UI Dashboard Settings: UI Type: UI, Use Telemetry:
- Wheel Physics Settings** (Right):
 - Ground Physic Materials:
 - Ground Physic Materials 0: RCCAsphaltPhysics
 - Ground Physic Materials 1: RCCGrassPhysics
 - Ground Physic Materials 2: RCCSandPhysics
 - Configure Ground Physic Materials
- SFX Settings** (Right):
 - Sound FX:
 - Configure Wheel Slip Sounds
 - Crashing Sounds
 - Gear Shifting Sounds
 - Indicator Clip: Indicator
 - Exhaust Flame Clips
 - NOS Clip: NOS
 - Turbo Clip: Turbo
 - Reset To Defaults
 - Open PDF Documentation
- RCC Asset Settings Editor Window** (Bottom Left):
 - This editor will keep update necessary .asset files in your project for RCC. Don't change directory of the "Resources/RCCAssets".
 - Reset To Defaults
 - Open PDF Documentation
 - Realistic Car Controller V3.0f
BoneCrackerGames
 - Created by Buğra Özdoğanlar
- RCC Asset Settings Editor Window** (Bottom Right):
 - This editor will keep update necessary .asset files in your project for RCC. Don't change directory of the "Resources/RCCAssets".
 - Reset To Defaults
 - Open PDF Documentation
 - Realistic Car Controller V3.0f
BoneCrackerGames
 - Created by Buğra Özdoğanlar
- RCC Asset Settings Editor Window** (Bottom Center):
 - This editor will keep update necessary .asset files in your project for RCC. Don't change directory of the "Resources/RCCAssets".
 - only 1 Audio Source:
 - Do Not Use Any Particle Effects:
 - Do Not Use Skidmarks:
 - Do Not Use Chassis Joint:
 - Chassis Joint is a main Configurable Joint for realistic body movements. Script is getting all colliders attached to chassis, and moves them outside to joint. It can be trouble if you are making game about interacting objects inside the car. If you don't want to use it, chassis will be simulated based on rigid velocity and angular velocity, like older versions of RCC.
 - Tags & Layers (Right):
 - Set Tags And Layers Auto:
 - Vehicle Layer: RCC
 - Vehicle Tag: Player
 - Tag All Children Gameobjects:
 - Be sure you have that tag and layer in your Tags & Layers.
 - All vehicles powered by Realistic Car Controller are using this layer. What does this layer do? It was used for masking wheel rays, light masks, and projector masks. Just create a new layer for vehicles from Edit --> Project Settings --> Tags & Layers, and select the layer here.
 - Resources (Right):
 - Head Lights
 - Brake Lights
 - Reverse Lights
 - Indicator Lights
 - Mirrors
 - Skidmarks Manager
 - Light Projector
 - Exhaust Gas
 - Chassis Joint
 - Car Camera
 - Hood Camera
 - RCC UI Canvas
 - Reset To Defaults
 - Open PDF Documentation
 - Realistic Car Controller V3.0f
BoneCrackerGames
 - Created by Buğra Özdoğanlar

Configurable Ground Materials

Changing ground materials physics, particles, damps, sounds, etc in [Tools → BoneCracker](#)

[Games → Realistic Car Controller → Configure Ground Materials.](#)



How does it work? If WheelCollider hits a collider with one of the physic material in list, changes will be applied.

Creating New Vehicles

Some developers struggling with creating new vehicles. So, i have improved some editor scripts for simplify creating new vehicles.

Warning

Script and behavior depends on vehicle X, Y, Z directions and pivots. So, your vehicle model and wheel models transform directions **should MUST** be correct. Just check the demo vehicles.

Important

Be sure you are in **PIVOT** and **LOCAL** mode while checking directions.



X should **Right**,

Y should **Up**,

Z should **Forward**.



Many designers are making models with wrong directions. This is really painful if you don't know how to fix models directions. 2 ways for fixing this;

1 – Create a empty Gameobject at the center of the model, and rotate the new Gameobject to correct directions. Then parent your model in to this new Gameobject. It's simple.

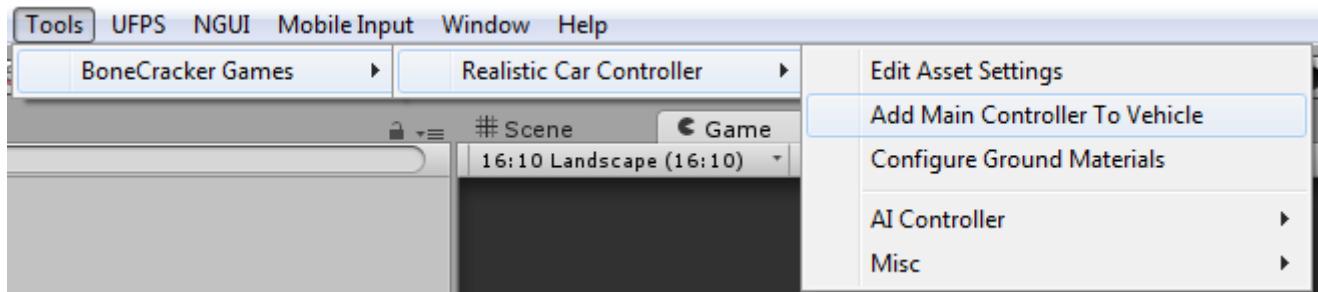
2 – Fix model's directions and pivot positions inside your Designing Software. Check my Youtube Channel for how to do this in 3ds Max.

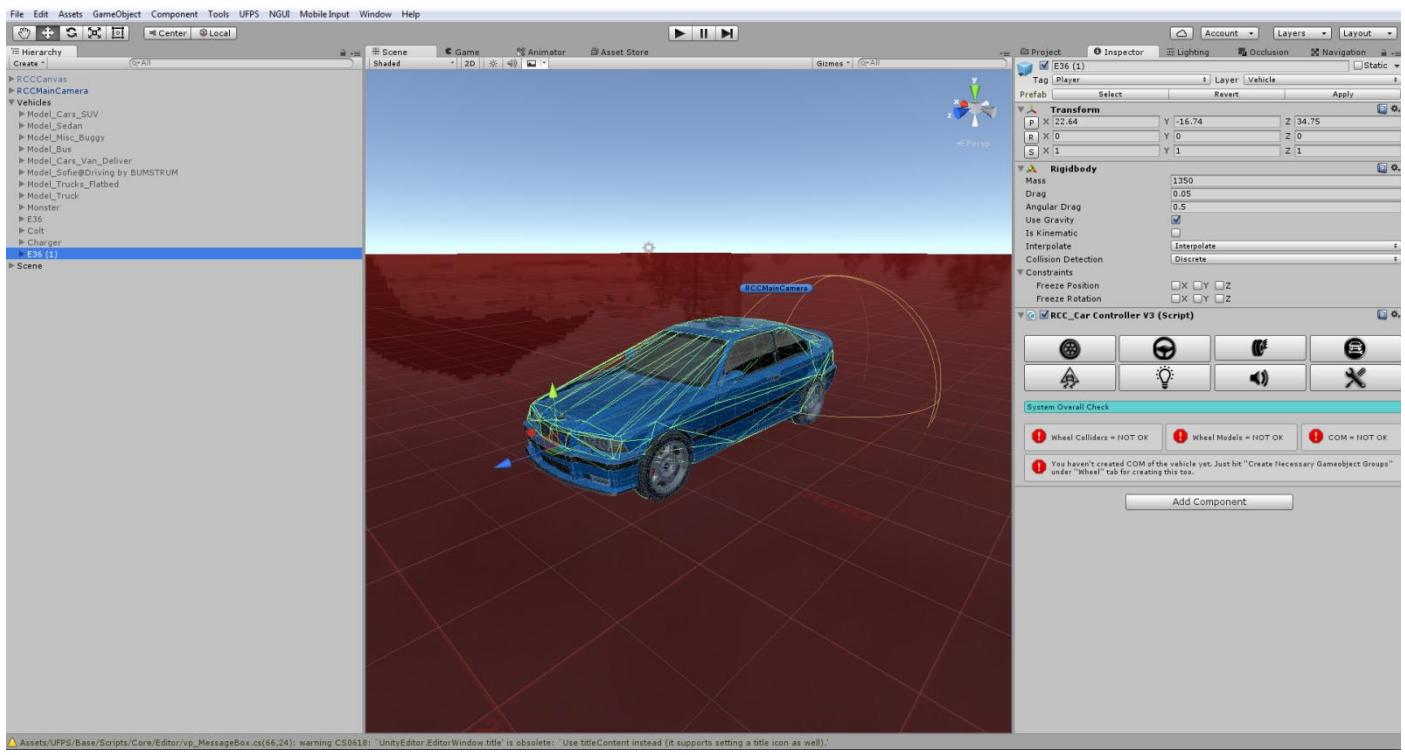
Also scale of your vehicle must be not oversized or miniature. Even if you want to make a toy car game, car size should be nearly same with demo vehicles. So, you have checked inputs, your vehicle and wheel models pivot positions, and their X, Y, Z directions. Everything is OK right?

Then...

Drag and drop your Vehicle Model to your existing Scene and let's get started;

You have to add **Main Controller** to root of the vehicle. Just select your vehicle model on your scene, and click **Tools → BoneCracker Games → Realistic Car Controller → Add Main Controller To Vehicle**.

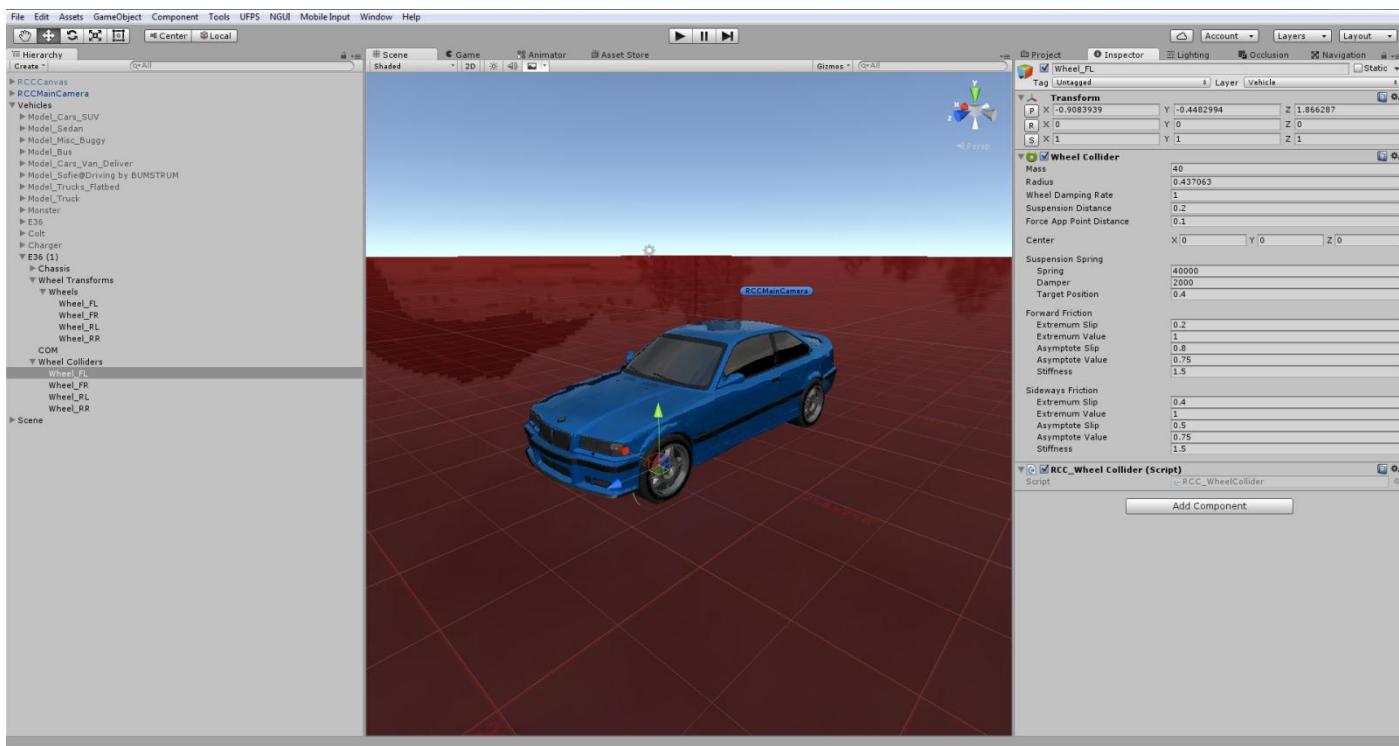
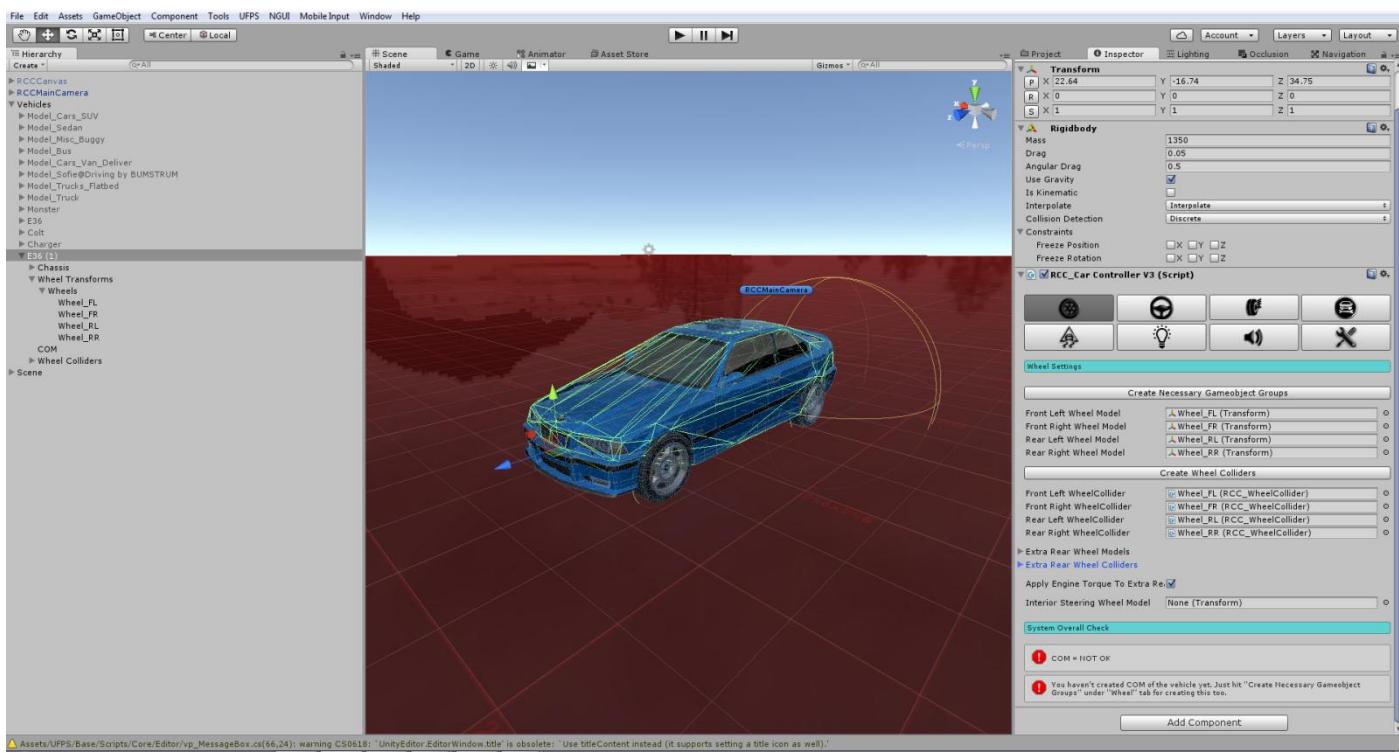




As soon as when you added **RCC_CarControllerV3.cs** script to your car, Rigidbody component will be added automatically. Set your mass to around 1250-1750 for this type of the vehicles. Interpolate Mode = Interpolate, Angular drag is around 0.1 – 0.25 for medium angular velocity.

These rigidbody settings are applied when you create your vehicle automatically.

After rigidbody settings, click “Wheel” tab in the editor script. Select all of your wheels. After selecting your wheel models, hit the “**Create Wheel Colliders**” for creating Wheel Colliders with proper radius, suspension, damper, and friction curves automatically.

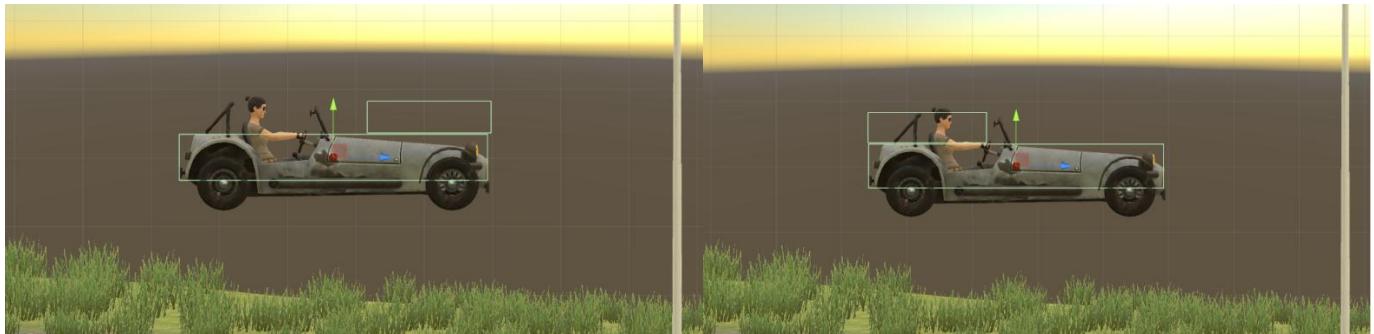


Generated Wheel Colliders settings are fine for 1250-1750 mass vehicles. If you have heavy vehicle such as bus or truck, you must increase Wheel Collider's mass. This value is controlled by [RCC_WheelCollider.cs](#) right now.

Your vehicle must have one of any [Colliders](#) (Such as [Box Collider](#), or [Mesh Collider](#) etc...)
Otherwise, physics won't work.

Collider Shapes Of The Vehicle

This is one of the most important thing for physics behavior. Most of devs are using mesh colliders for their vehicles. Remember that, shape of your vehicle collider will affect physics behavior directly. Let me explain this;



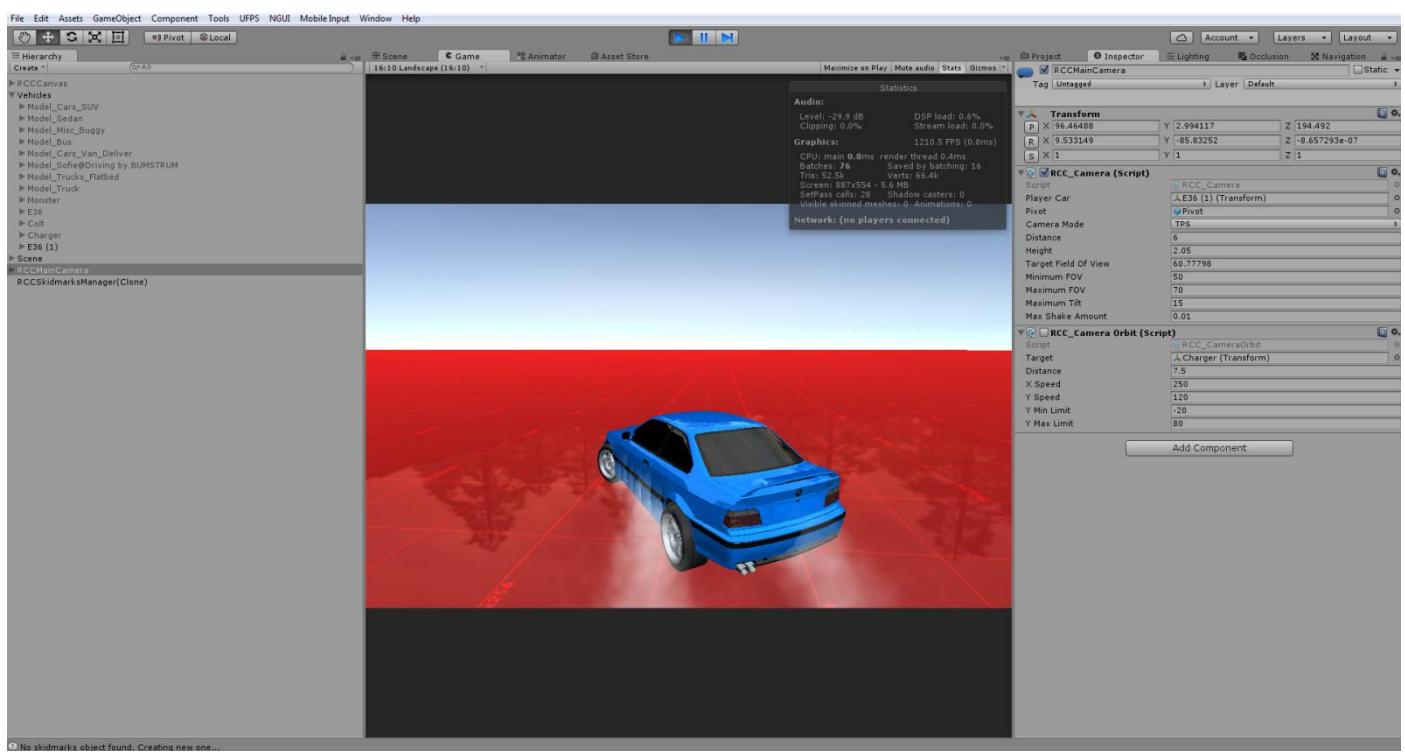
This vehicle has 2 Box Colliders. Second Box Collider is placed at front of the vehicle here on first screenshot. This will bring you more controllable and stable vehicle. Usually, second Box Collider should be at center Z of the vehicle. But this is high speed vehicle, so it must be stable. That's why I did this.

Second Box Collider is placed at rear of the vehicle on second screenshot. This will bring you more unstable and slippy vehicle. Differences between first colliders and second colliders are huge. Just remember that collider shapes are effecting vehicle behavior directly. If you want to make your cars stable, just use one big box collider.

After end up with wheel models and wheel colliders configurations, place your COM to correct place. This is our Center Of Mass. And COM's position is effecting whole behavior. Usually COM of the vehicle is at just below about front seats. Engine and transmission is at front of the vehicle, and they are heavy. This model is RWD. It has shaft at middle and differential at back of the vehicle. So, i'll just set it to just like this;



Runs perfectly after just few clicks.



Configure your vehicle as you wish. If you want to use manual gear, you need to set it from RCC Settings. Also you can select which key to shift up and shift down here.

Driving Assistances

Main Controller has **ABS, TCS, ESP, Steering Assistance, and Traction Assistance**. Threshold means, if wheel slip is equals or higher than this threshold value, corresponding assistance will be engaged immediately. Steering assistance will apply local Y axis torque to vehicle for more easily turns. But results more unrealistic turns like arcade games. 0.1f would be good for all vehicles. (0.2f will be used if behavior type is Racing or Semi-Arcade.)

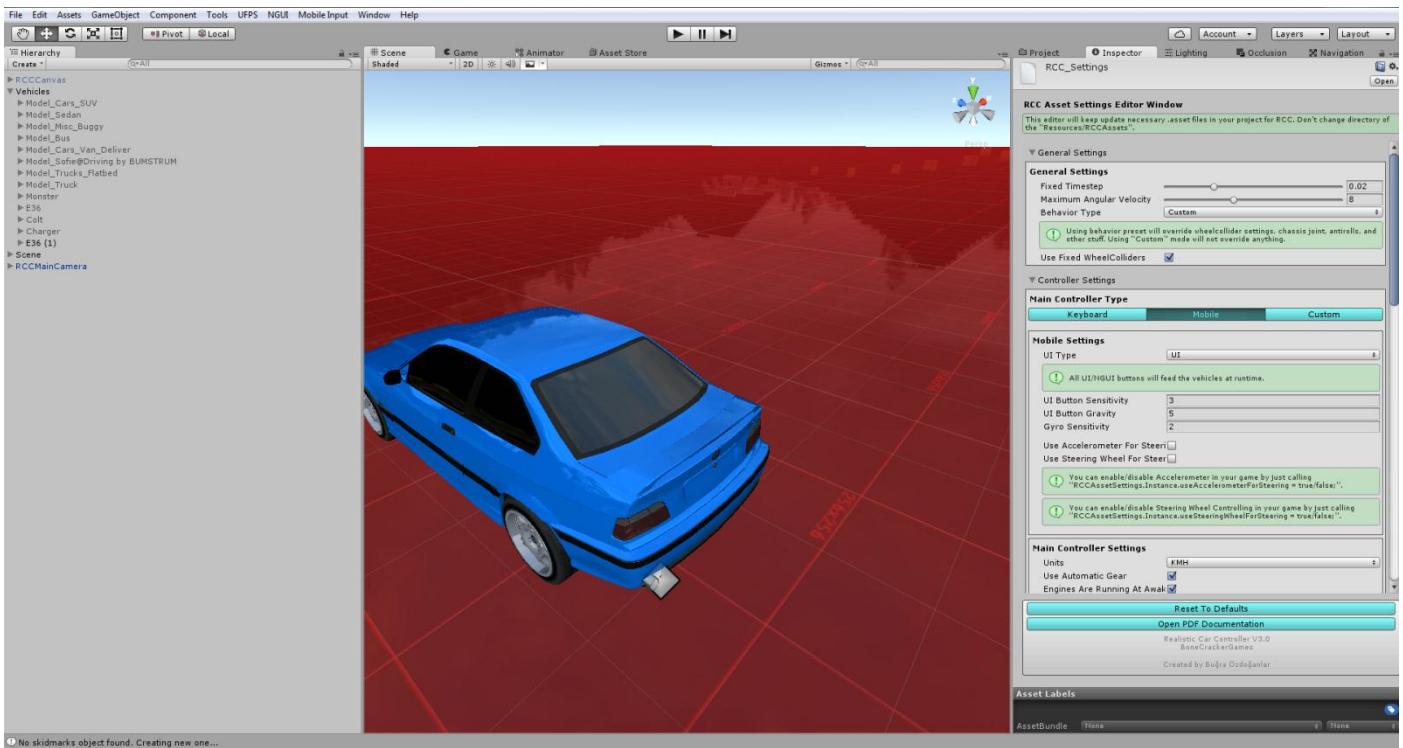
Mobile Controllers

Package contains **Mobile Controller** ready to use with UI and NGUI. Now let's make our vehicles can be controller by Mobile Controller. Just open up RCC Settings from **Tools → BoneCracker Games → Realistic Car Controller → RCC Settings**. You can change your inputs for **Keyboard**, **Mobile Controllers**, and **Custom Controllers** here.

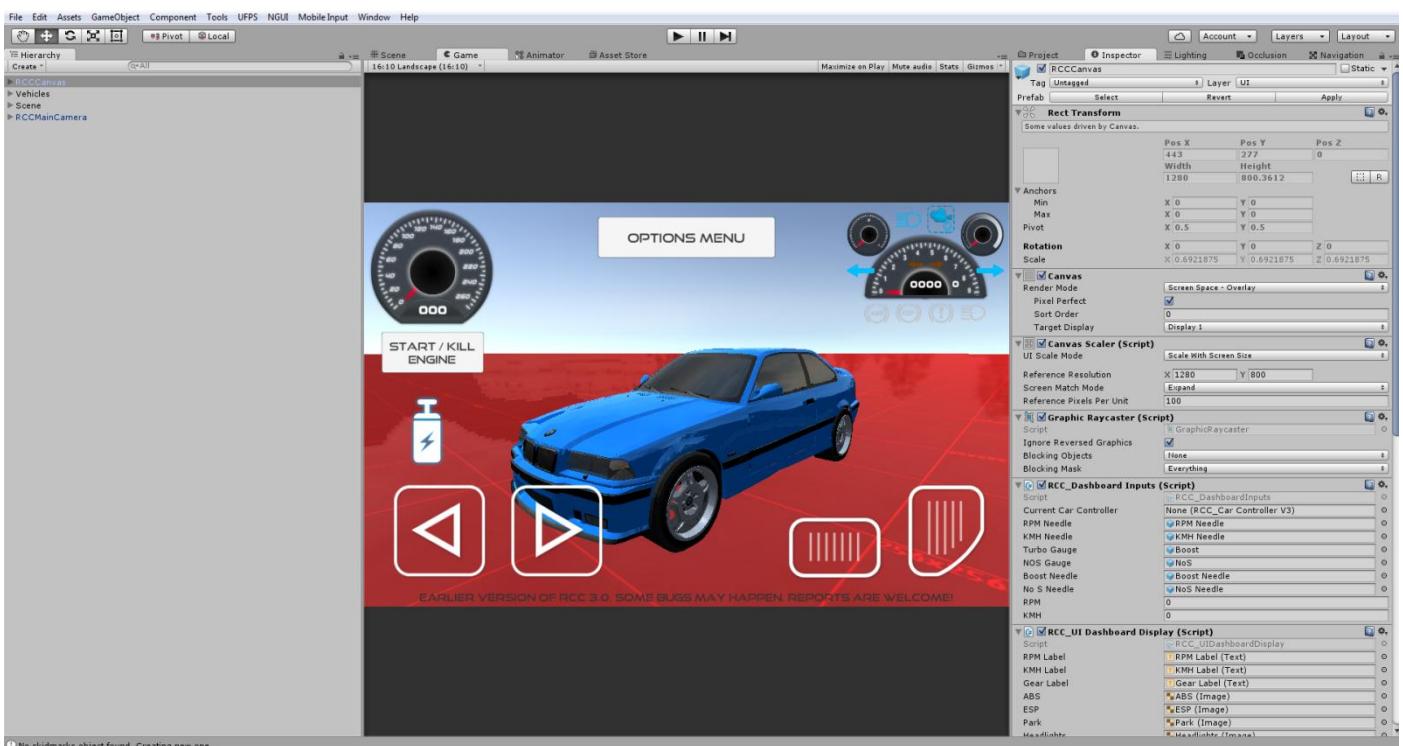
Package supports two type of controller and dashboard display;

- **Unity UI,**
- **NGUI.**

Mobile Controller (With Unity UI)



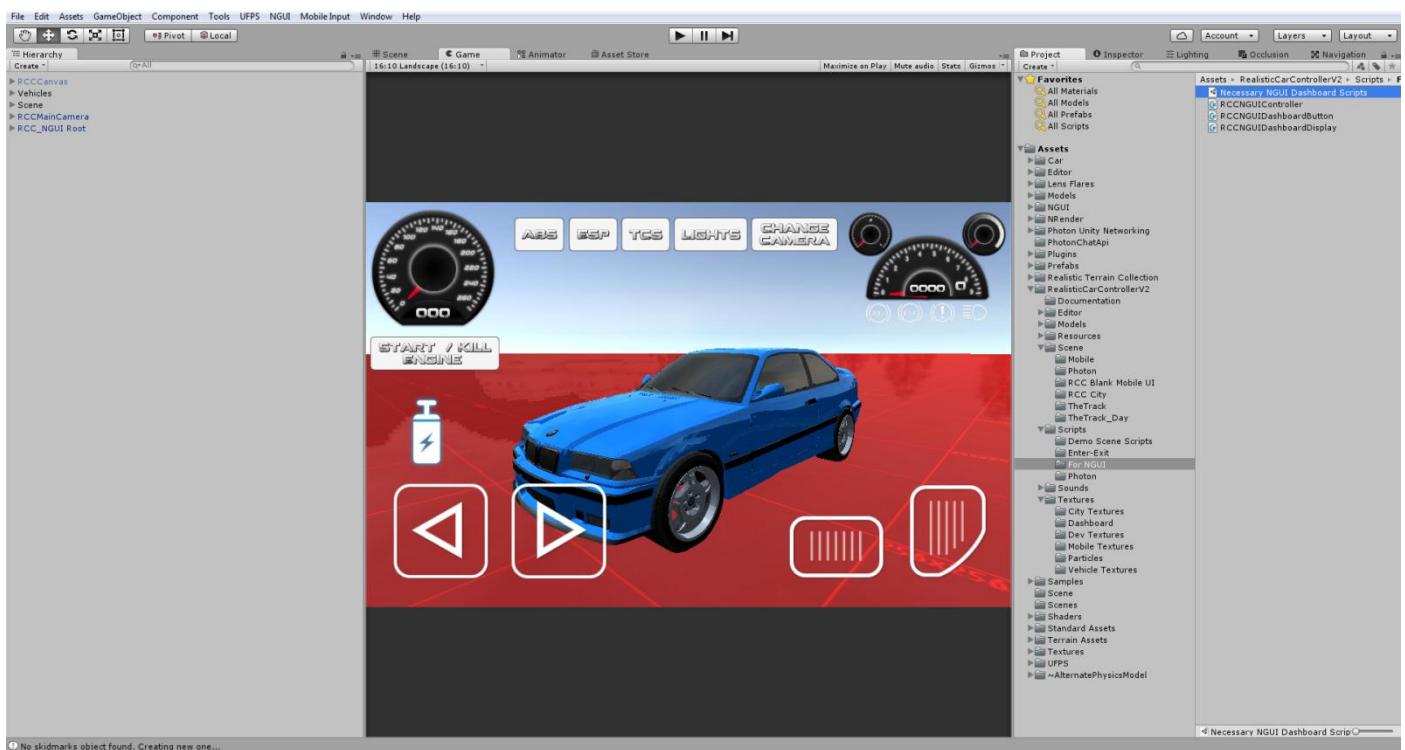
Default Mobile Controller type is “[UI Controller](#)”. You will find “[RCCCanvas](#)” prefab under [RealisticCarControllerV3/Prefabs](#). Drag and drop to your Hierarchy.



Each UI controller gameobject has “[RCC_UIController.cs](#)” script for inputs. These buttons feeds car controller with normalized float values. You can adjust UI buttons sensitivity and gravity from RCC Settings

[Mobile Controller \(With NGUI\)](#)

First, import latest NGUI Package to your Project. Then, import “[Necessary NGUI Dashboard Scripts](#)”(in For NGUI folder) to your Project;



You will find “[RCC_NGUIRoot](#)” under [RealisticCarControllerV3/For NGUI](#). Drag and drop the [RCC_NGUIRoot](#) prefab to your Hierarchy.

This prefab includes [NGUI Controller](#), [NGUI Dashboard](#), and other necessary scripts for Ready to use. Canvas is designed for all aspect ratios. Controller elements won't disappear on other aspect ratios or screen resolutions.

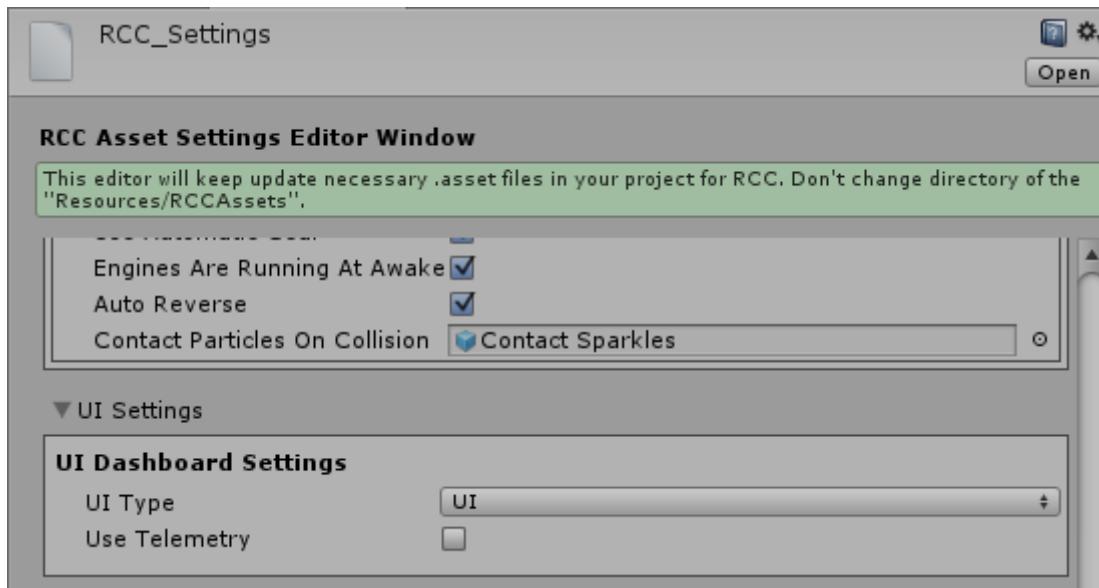


About Mobile Usement on City Scene

City scene has lot of specular maps with alpha channels. Textures with alpha channels are heavy for mobile devices. In Demo APK from my website is not using any texture with alpha channels. Also all standard shaders are replaced with legacy shaders. If you build an APK without editing materials, you will get performance loss.

Dashboard Configuration

You can access dashboard from RCC Settings.



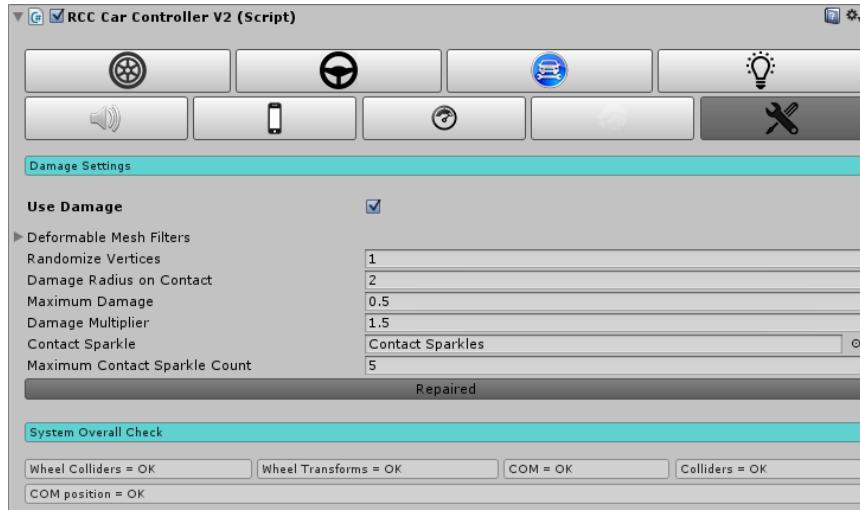
Default Dashboard Type is “[UI Dashboard](#)”. Script will access “[RCC_DashboardInputs.cs](#)” and “[RCC_DashboardDisplay.cs](#)” scripts for displaying dashboard. These scripts are attached on;

[Unity UI](#) - RCCCanvas root.

[NGUI](#) – RCCNGUI Root.

It's extremely easy to create and customize your own dashboard and controller HUD on Unity UI and NGUI instead of writing your own OnGUI() method.

Damage (Based on Mesh Deformation)



Your vehicle body mesh wireframe topology must be reliable for realistic vertices movement. If your vehicle body mesh has broken (unwelded) vertices or bad wireframe topology, mesh will deform buggy and unrealistically.

Creating Lights, Sounds, Skidmarks, Smoke

Effects

These effects are optional.

Create Point Lights for braking and reverse gear, spot lights for headlights. Place them correctly on your vehicle model. You can create all kind of lights from [Tools → BoneCracker Games → Realistic Car Controller → Create](#).

Script doesn't Instantiate, Destroy any smoke particles, or any kind of stuff. Just enabling/disabling particle emitters for avoid garbage memory.

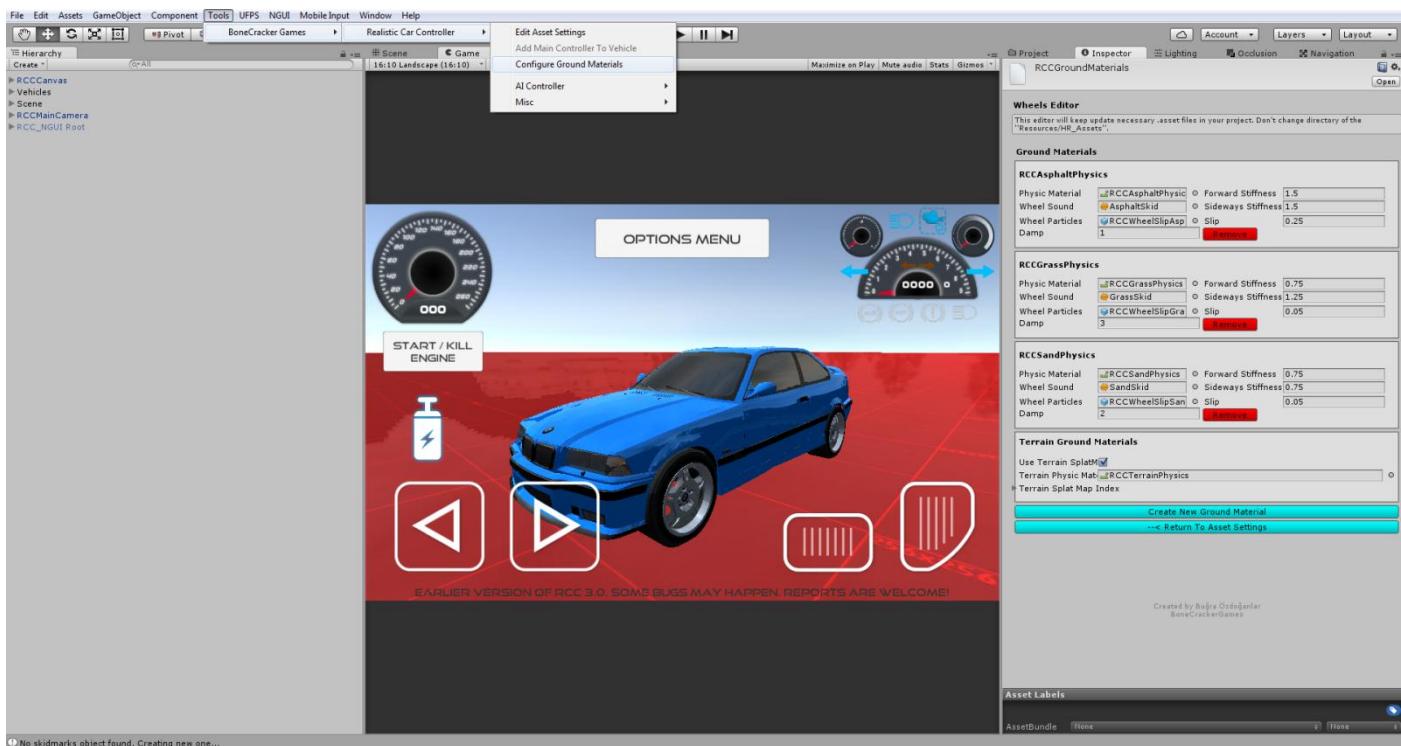
You will find “**RCCWheelSlipAsphalt**”, “**RCCWheelSlipGrass**” and “**RCCWheelSlipSand**” prefabs under Prefabs folder. You can use your own smoke prefab as you wish.

If you want to use exhaust effects, you can create it from **Tools → BoneCracker Games → Realistic Car Controller → Create**. You need to place it to your model correctly. That's it.

You will find “**RCCWheelSkidmarks.cs**” under Scripts folder. Scene must have “**RCCSkidmarksManager**” to create skid meshes. You will find the “**RCCSkidmarksManager**” prefab under Prefabs folder. Drag and drop to your Hierarchy. If your scene doesn't have, **RCC_WheelCollider** will create itself.

Variable Ground Tire Grip

If you want to get variable tire grips on varied surfaces, you can use this feature. Open up **Tools → BoneCracker Games → Realistic Car Controller → Configure Ground Materials**.

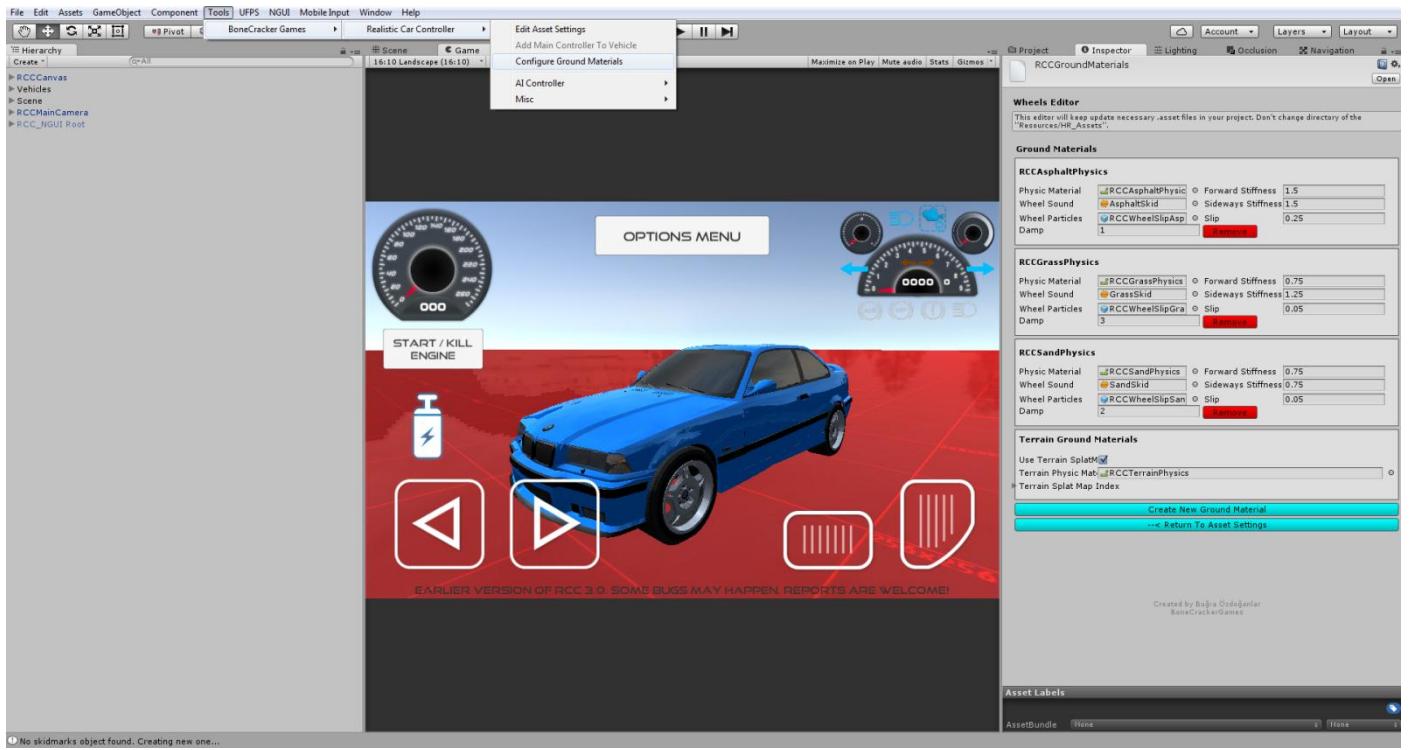


Currently 3 surfaces available such as Asphalt(Default), Sand, and Grass. You will find “**RCCAsphaltPhysics**”, “**RCCGrassPhysics**” and “**RCCSandPhysics**” Physic Materials under

“Resources” folder. If your scene ground is not a Unity Terrain, and made by individual gameobjects, you have to assign each ground gameobject collider’s Physic Material to corresponding one. For ex. Select your grass ground gameobject, and select it’s collider’s Physic Material as “**RCCGrassPhysics**”.

But if your scene has a Unity Terrain as a ground, your Terrain textures will decide which surface your on. Open up **Tools → BoneCracker Games → Realistic Car Controller → Configure Ground Materials**.

Select each index of your terrain texture slot for corresponding physics material. And don’t forget to enable “**Use Terrain SplatMap For Ground Physics**”.



Adjusting Ground Particles, Wheel Sounds, Damp, Forward and Sideway Stiffness, Slip

On Different Grounds

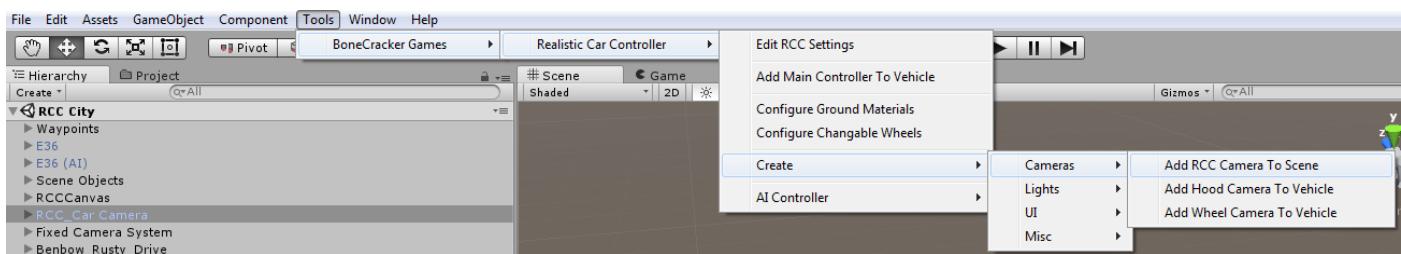
You can adjust each materials Ground Particles, Wheel Sounds, Damp, Forward and Sideway Stiffness, Slip here.

As i said, these are optional effects. If you don't want to use them, just leave.

RCC Camera System

Main Camera system designed for using with RCC. Includes 7 different camera modes with many customizable settings. It doesn't use different individual cameras on your scene like *other* assets. Simply it parents the camera to their positions that's all. No need to be Einstein.

If your scene doesn't have RCC Camera, you can create it from [Tools → BoneCracker Games → Realistic Car Controller → Create → Cameras → Add RCC Camera To Scene](#)



RCC_Camera (Script)

Main Camera designed for RCC. It includes 5 different camera modes. It doesn't use many cameras for different modes like *other* assets. Just one single camera handles them.

Player Car

Pivot of the Camera

Current Camera Mode

TPS

TPS Distance	6
TPS Height	2
TPS Height Damping	5
TPS Rotation Damping	3
TPS Minimum FOV	55
TPS Maximum FOV	70

FPS

Use Hood Camera Mode

(!) Be sure your car has "Hood Camera". Camera will be parented to this gameobject. You can create it from Tools --> BCG --> RCC --> Camera Systems --> Add Hood Camera.

Hood Camera FOV

Wheel

Use Wheel Camera Mode

(!) Be sure your car has "Wheel Camera". Camera will be parented to this gameobject. You can create it from Tools --> BCG --> RCC --> Camera Systems --> Add Wheel Camera.

Wheel Camera FOV

Fixed

Use Fixed Camera Mode

(!) Fixed Camera is overrided by "Fixed Camera System" on your scene.

Select Fixed Camera System

Cinematic

Use Cinematic Camera Mode

(!) Cinematic Camera is overrided by "Cinematic Camera System" on your scene.

Select Cinematic Camera System

Orbit

Use Orbit Camera Mode

RCC_Customization

You can make important changes for your vehicles by just calling a single method. Please take a look at "[Realistic Car Controller V3 Scripts](#)" documentation. Methods are explained there.

How The Customization Panel Works

I've wrote a example script called "[RCC_CustomizerExample.cs](#)". Script is attached to RCCCanvas. UI Buttons in Customization Panel sends methods to thix example script. And this example script uses static methods in [RCC_Customization.cs](#) for making changes. Let me explain it with simple examples;

We want to change front suspension distance of our car. So, we have to call it;

[RCC_Customization](#).[SetFrontSuspensionsDistances](#) ([targetRCC](#), [targetValue](#));

We want to repair our car. So, we have to call it;

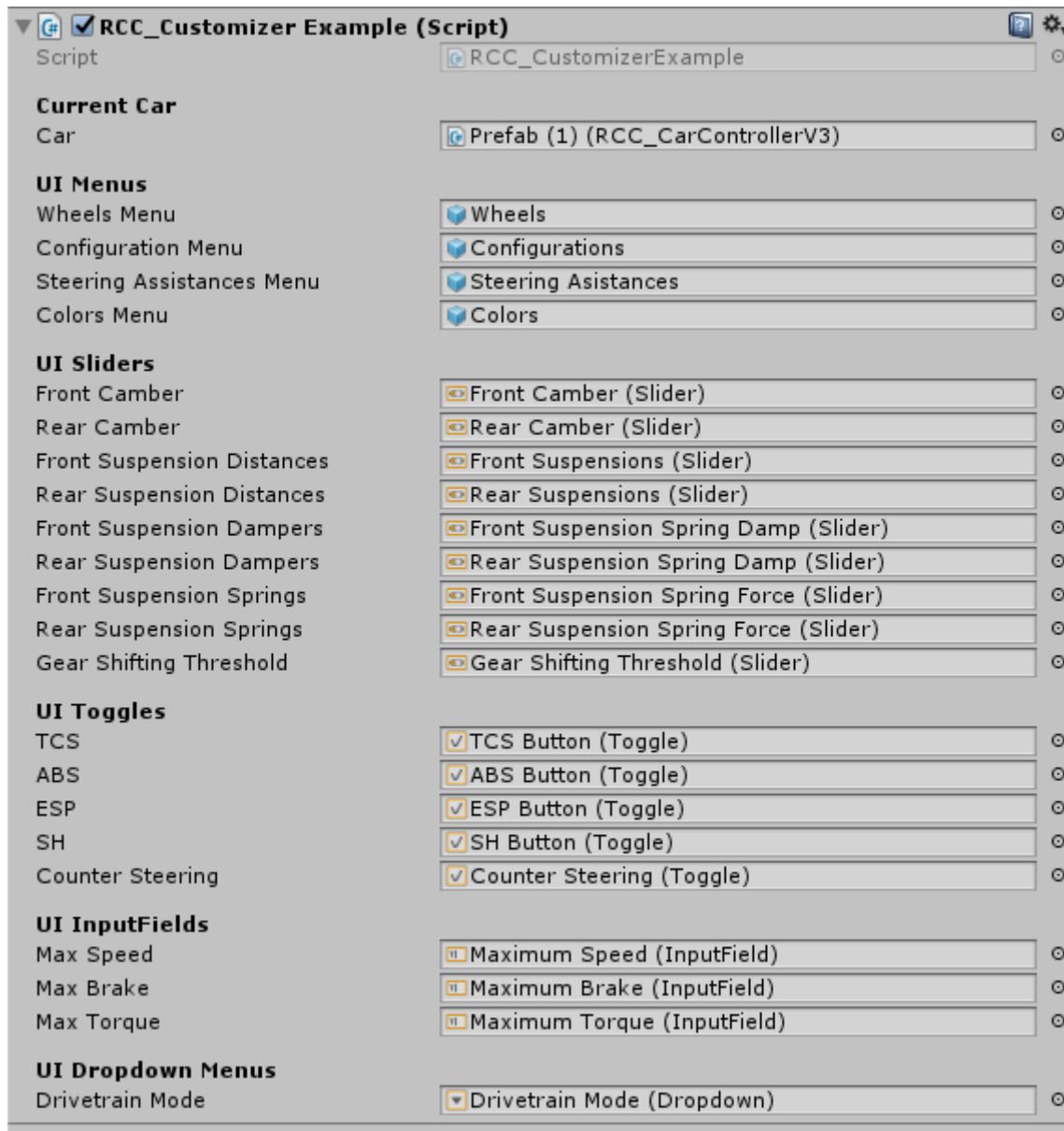
[RCC_Customization](#).[RepairCar](#) ([targetRCC](#));

We want to change drivetrain of our car to AWD. So, we have to call it;

[RCC_Customization](#).[SetDrivetrainMode](#) ([targetRCC](#), [RCC_CarControllerV3](#) [WheelType.AWD](#)) ;

And goes on...

[RCC_CustomizerExample.cs](#) attached to RCC Canvas.

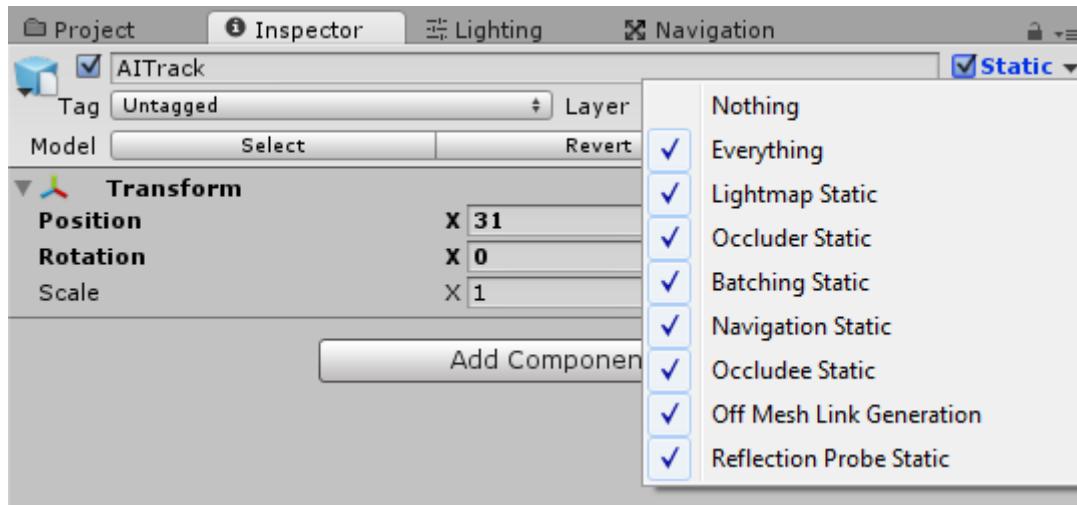


This example script handles all UI menus, buttons, sliders, toggles, inputfields, and dropdown menus. It just receives inputs from UI, and fires necessary action.

AI Configuration

Creating NavMesh For Scene

AI is based on Unity's Nav Mesh. Therefore, you must bake and create navigation mesh for your scene. Select your all static objects (including road too). And set them as "Static".

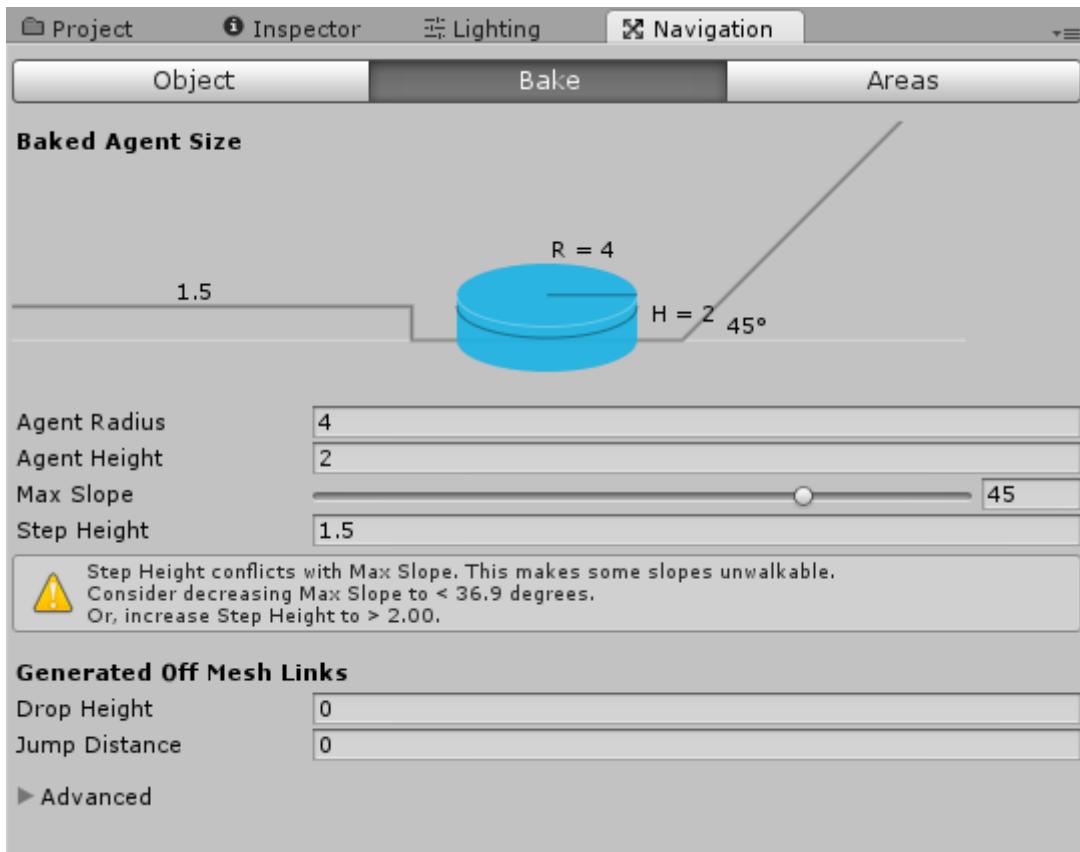


When all your static objects are marked as "Static", then you can bake your navigation mesh.

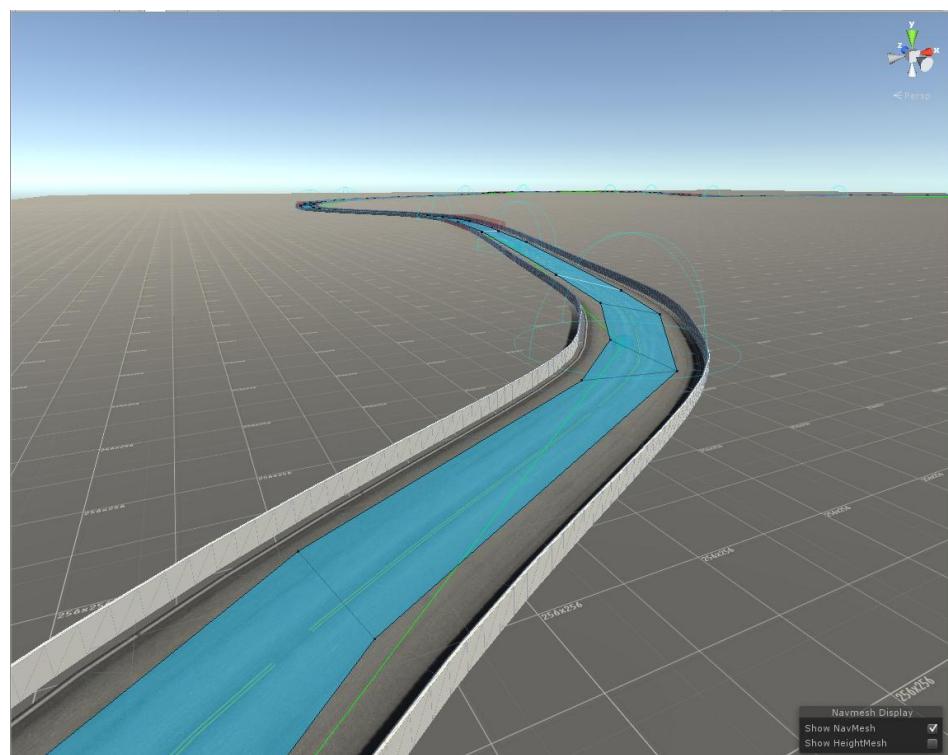
Open "[Navigation](#)" window from [Window → Navigation](#).



Default settings should be like this;

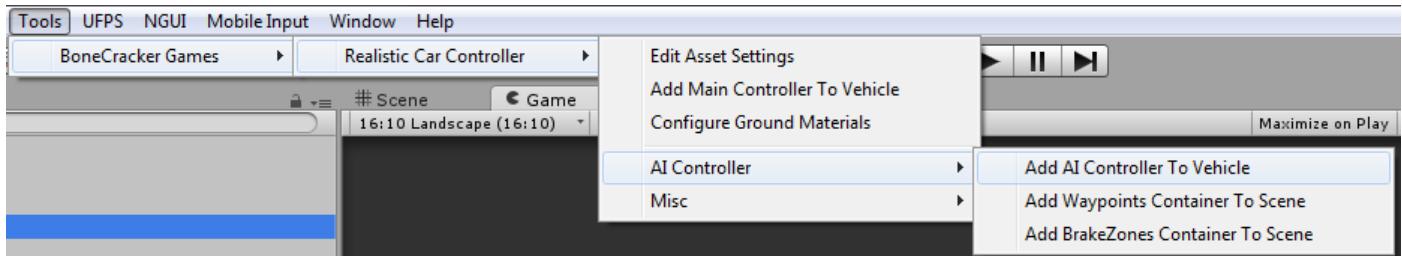


And then, click the bake button and bake your scene. Check your blue navigation mesh. Should be like this;

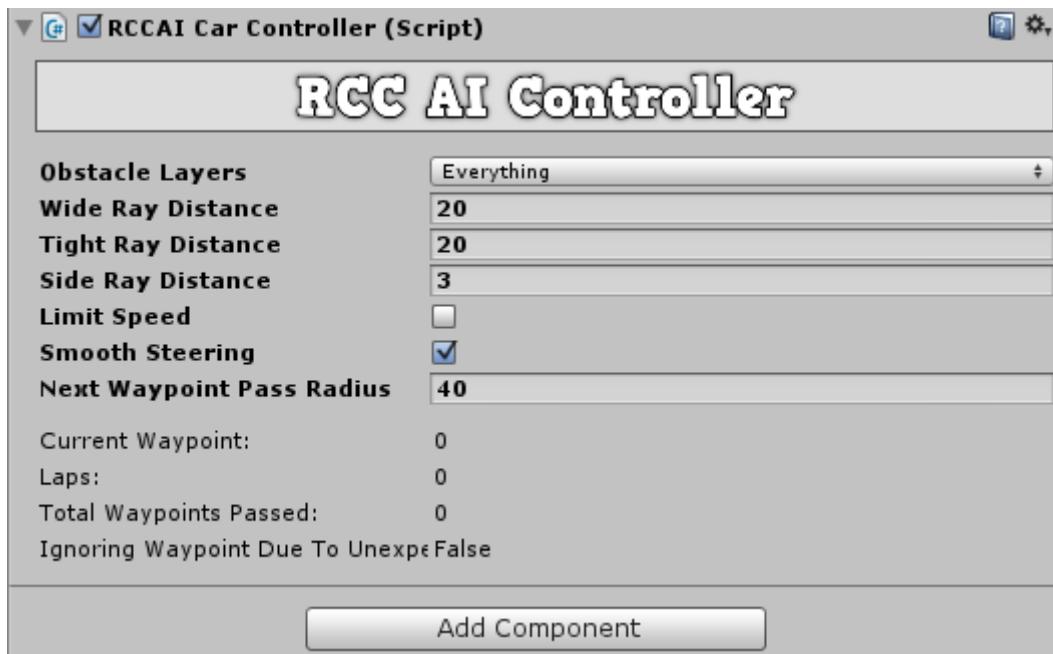


Adding AI Controller To Vehicle

First, build and configure your vehicle. When everything works fine and results are as expected, you can add AI Controller to your vehicle by clicking “Tools → BoneCracker Games → RCC → AI Controller → Add AI Controller To Vehicle”

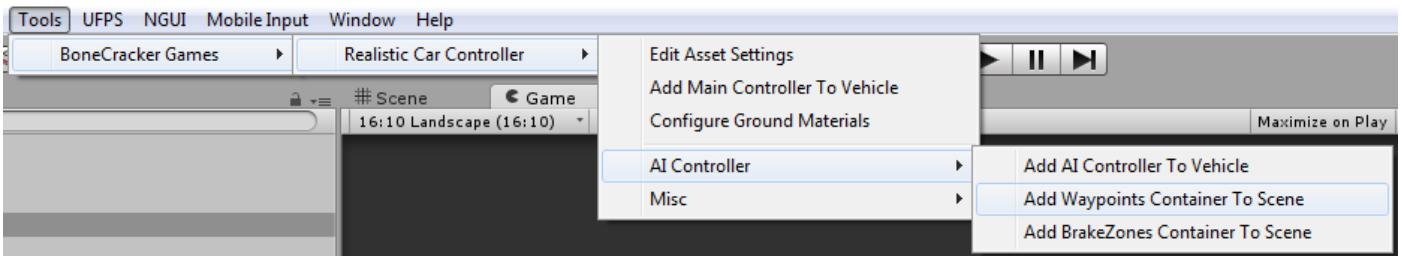


This will add “**RCC_AIController**” to root of your vehicle.

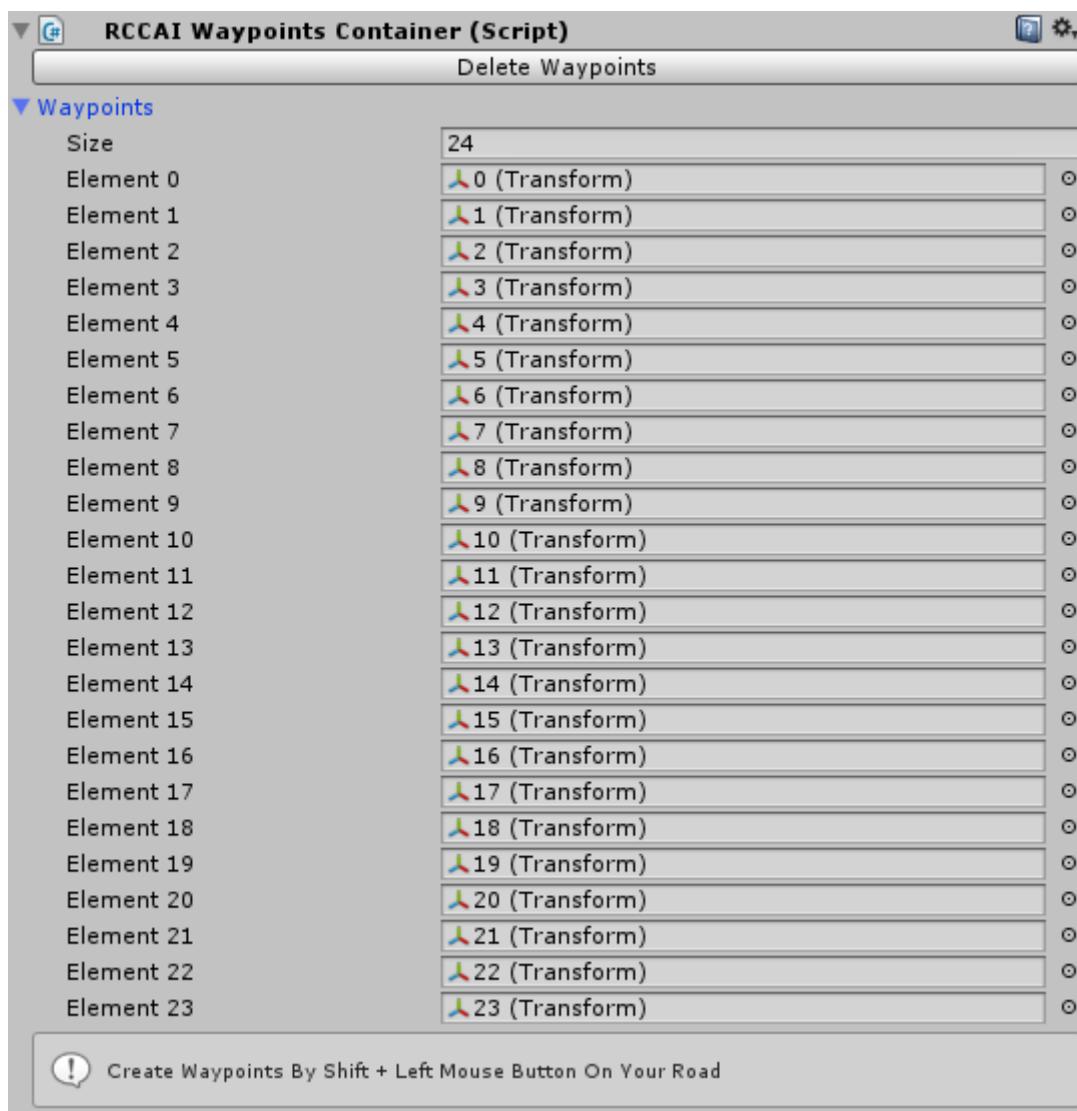


Vehicle will use “**Nav Mesh Agent**” for road path based on your waypoints, and will use raycasts for dynamic objects. If you have specified gameobjects for ignoring raycasts for this gameobject, you can select your gameobject layer from obstacle layers.

Adding Waypoints Container To Scene

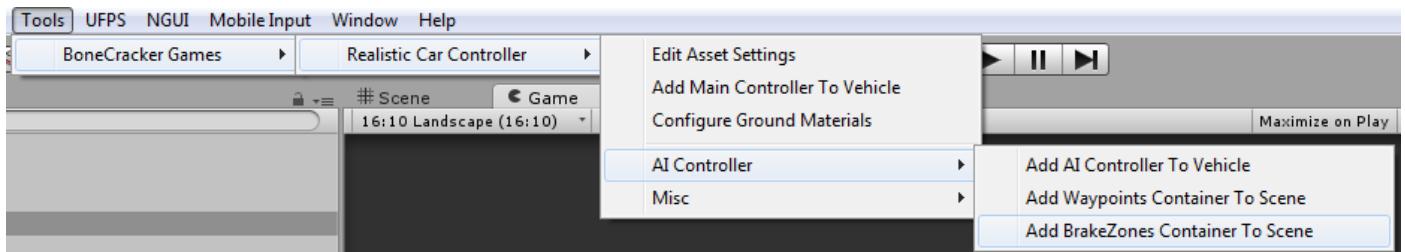


This will add “**RCC AI Waypoints Container**” to your scene. Simply hold Shift and click on your road to create a new waypoint. Create your path with them.

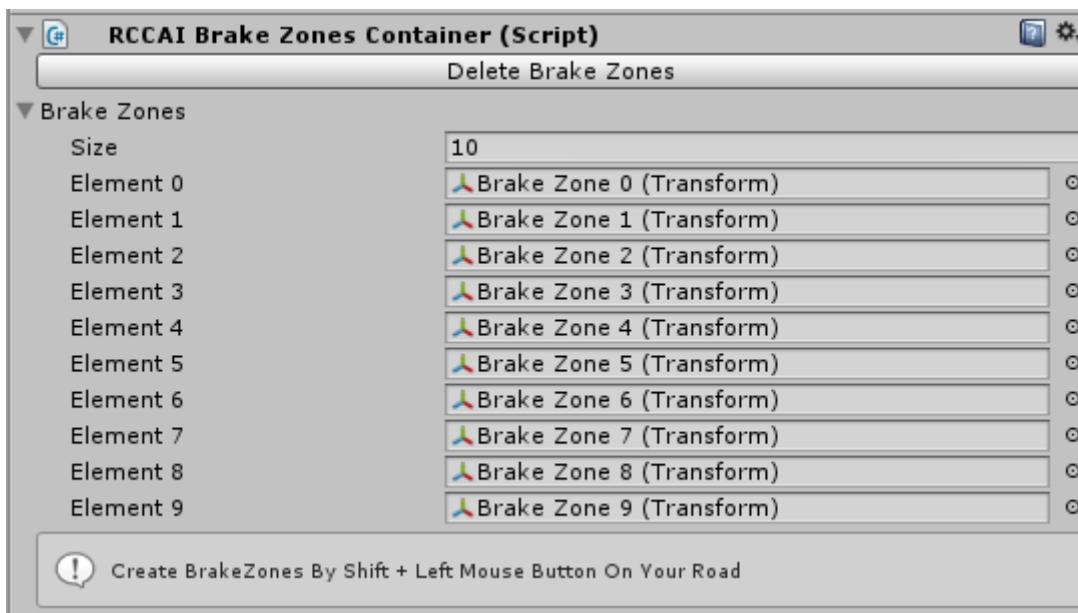


Info: Do not use CTRL+D for duplicate any waypoint.

Adding BrakeZones Container To Scene



This will add “RCC AI BrakeZones Container” to your scene. Simply hold Shift and click on your road to create a new brake zone.



Info: Do not use CTRL+D for duplicate any brake zone.

Each brake zone has a target speed. Vehicle will adapt its speed to this target speed when in this brakezone. You can scale your brakezones to fit to your road section.

Enter-Exit System

You can use your FPS or TPS player for enter-exit vehicles. All you have to do is, add necessary scripts to vehicle and player. Get out positions are created automatically in script if you don't select it in [RCC_EnterExitCar](#).

For FPS characters, it will depend on your camera of the FPS player. For TPS player, it depends on your actual TPS Player, not camera.

F.A.Q.

Q. My vehicle is too unstable!

A. Might be wrongly placed COM. Do not use wrong shaped colliders. Do not use overlapped colliders. Use driving assistances.

Q. My vehicle wheels or wheelcolliders are facing to wrong direction!

A. Be sure your wheel models x, y, z and pivot positions are correct (It's explained above)

Q. Why I can't drift like your video?

A. Many things. First of all, COM. Your center of mass must be at correct position. Turn on Drifting is related with gizmos and check your wheel forces while car stands. Should be %55 front, %45 rear. Check out your collider shape (It is explained above). Apply high engine torque. Be sure you are selected "Drift" behavior from [RCC Settings](#). This will override your vehicle settings. And also [RCC_WheelCollider](#) will use Drift() method to change wheelcollider friction curves at runtime.

Q. I'm getting huge performance loss and red exceptions!

A. Maybe you've found a bug. Send me your error immediately, so I can fix it.

Multiplayer with Photon

You have to import the necessary scripts to the project. First, download and import Photon. Pass your AppID to Photon setup (explained below), and then import “[RCC_PhotonNecessaryScripts](#)” in Scripts/Photon folder. Now you can test the Photon demo scene.

Well, it's quite easy I guess. If you don't know anything about Photon, or even multiplayer based game, here is the events for how it works;

1 – First, you have to connect to server. In this case, our server is Photon Cloud Server.

2 – If your connection to server succeeded, you will be in lobby. Now you can create your room, or join someones room, or join randomly room here.

3 – Congrats, now you are in an online multiplayer based room in realtime.

Multiplayer demo scene is using Photon. You have to create your account from their site. It's free. As soon as you logged in your account, create an AppID for your account. Now, download Photon Free from Asset Store and import it.

As soon as you have imported it to your project, it will ask your AppID. Pass it. And now you are ready to develop your realtime based multiplayer levels. It's free. And of course you are limited with 20 CCU.

Photon has many simple methods in their API. It's extremely easy to understand. Let me explain how the demo scene works;

Demo scene has a gameobject with script named “[RCC_PhotonManager](#)”. This script fires up multiplayer section of the scene. Uses these methods (You can find all methods from Photon’s docs);

“PhotonNetwork.ConnectUsingSettings("RCC V3.1");”

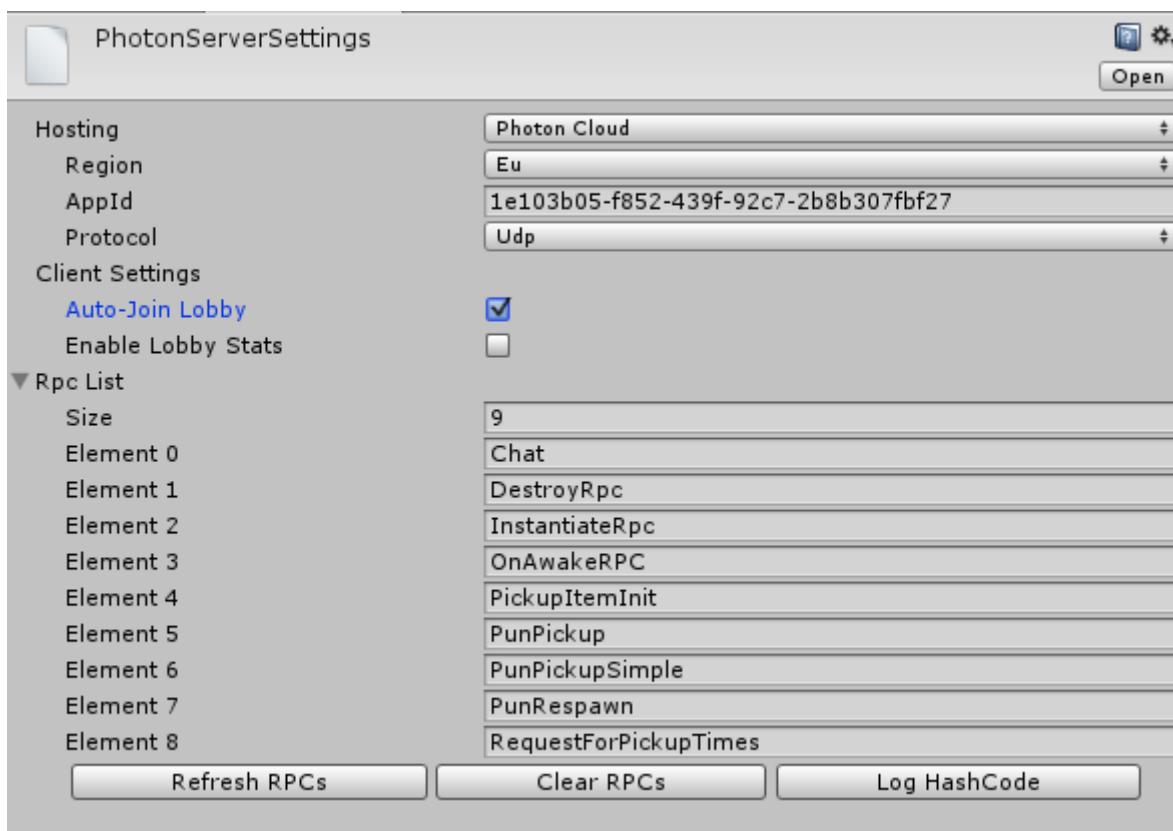
We are connecting to the server first. We can listen which connection status we are on in OnGUI() method. Like this;

“GUILayout.Label("State: " + PhotonNetwork.connectionStateDetailed.ToString());”

OnJoinedLobby()

As soon as we are connected to lobby, we want to join a random room by “PhotonNetwork.JoinRandomRoom();”

Auto Join Lobby must be selected at Photon!



OnPhotonRandomJoinFailed()

If it fails, this means there are no any other room. We are creating the new room by PhotonNetwork.CreateRoom(null); This method needs room name. I didn't use it, because there are no any room list in demo.

I take a string that belongs to player here. And set it by "PhotonNetwork.playerName = name;". Enabling/disabling few UI gameobjects depends on connection state. That's basically, how the demo scene works.

For vehicle sync, each vehicle has [PhotonView.cs](#) and [RCC_PhotonNetwork.cs](#). These scripts are necessary for each vehicle. [RCC_PhotonNetwork.cs](#) is observed by [PhotonView.cs](#).

[RCC_PhotonNetwork.cs](#) is synchronizing all control inputs, transform position, rotation, and rigid velocity smoothly. If vehicle is our vehicle, it will broadcast your data to the server. If vehicle is not our vehicle, it will receive all data from server.

These vehicles are not instantiated or destroyed with regular [GameObject.Instantiate](#) or [Destroy](#). You have to do it with [PhotonNetwork.Instantiate](#) or [Destroy](#). Unfortunately, it won't work with your prefab. It accepts only strings for your vehicle. That means, it will use [Resources](#) folder for accessing your vehicles. Your vehicles must be at Resources folder. Therefore, there are two canvases in resources folder. One of them is using [RCC_Demo.cs](#), other one is using [RCC_PhotonDemo.cs](#).

Credits

Extreme Vehicle Pack by Vertigo Games

Package contains sponsored vehicles by Vertigo Games. You can get this asset from this link;

<http://u3d.as/4xM>

Sofie With Animations by 3DMaesen

Driver Sofie, her animations, and her car model made by 3DMaesen. You can access 3DMaesen asset store from this link;

<http://u3d.as/2vg>

Sound Effects

All sounds in package are completely Royalt Free. You can use them on any personal or commercial projects.

License

You can use this package for unlimited games. Both personal and commercial use. **But you can't resell or redistribute any asset in the package on any store (not even any single asset in package)**. I got many reports from my customers about some fake developers are reselling my package on other stores. **This is strictly forbidden**. You can't resell or redistribute ANY asset from Unity's Asset Store, unless if developer gave you special license for making this.

If anyone violates this, he will be banned, and his revenue from package sellings will be interrupted. You can read Unity EULA from this link;

http://unity3d.com/legal/as_terms

So, how you holding up there? You can ask me anything about my assets! If you want change **minor things** in the package, don't waste your time with editing scripts. Just tell me, I'll do my best with no cost. I don't take any Projects right now, and I'm not available for hire. Please mail me if you used any of my Assets on your game. I'd like to see it in action!

Made by Bugra Özdoganlar

Contact@BoneCrackerGames.com