Realistic Car Controller V3.2 by BoneCracker Games

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

Contents

You can find more updated details on

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

YouTube Playlist

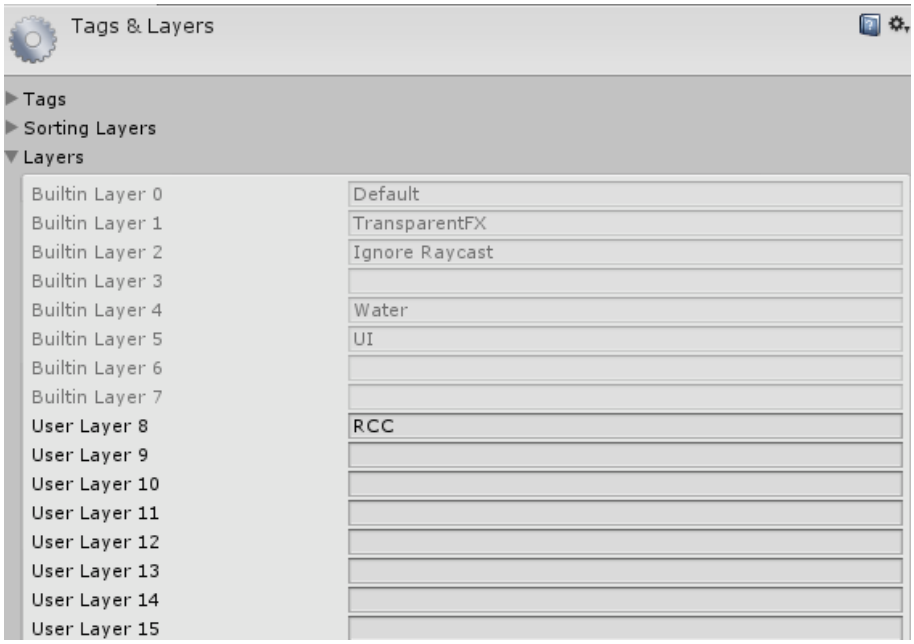
https://www.youtube.com/playlist?list=PLRXTqAVrLDpoW58lKf8XA1AWD6kDkoKb1

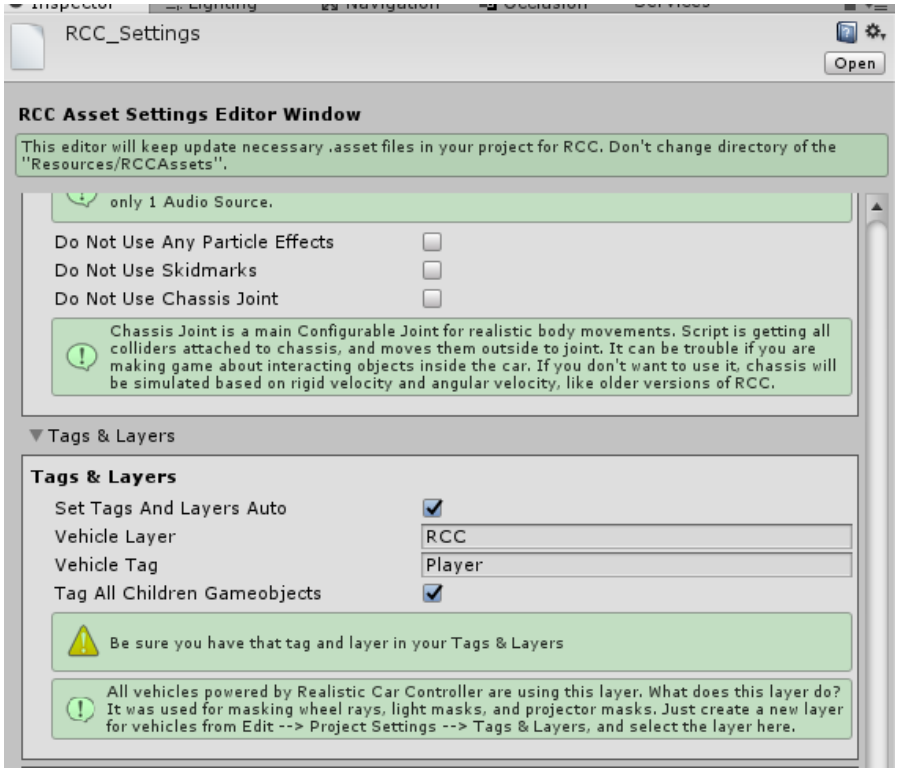
(You can zoom in with CTRL + ScrollUp for enlarge PDF pages)

First to Do!

Always backup your project before the update.

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

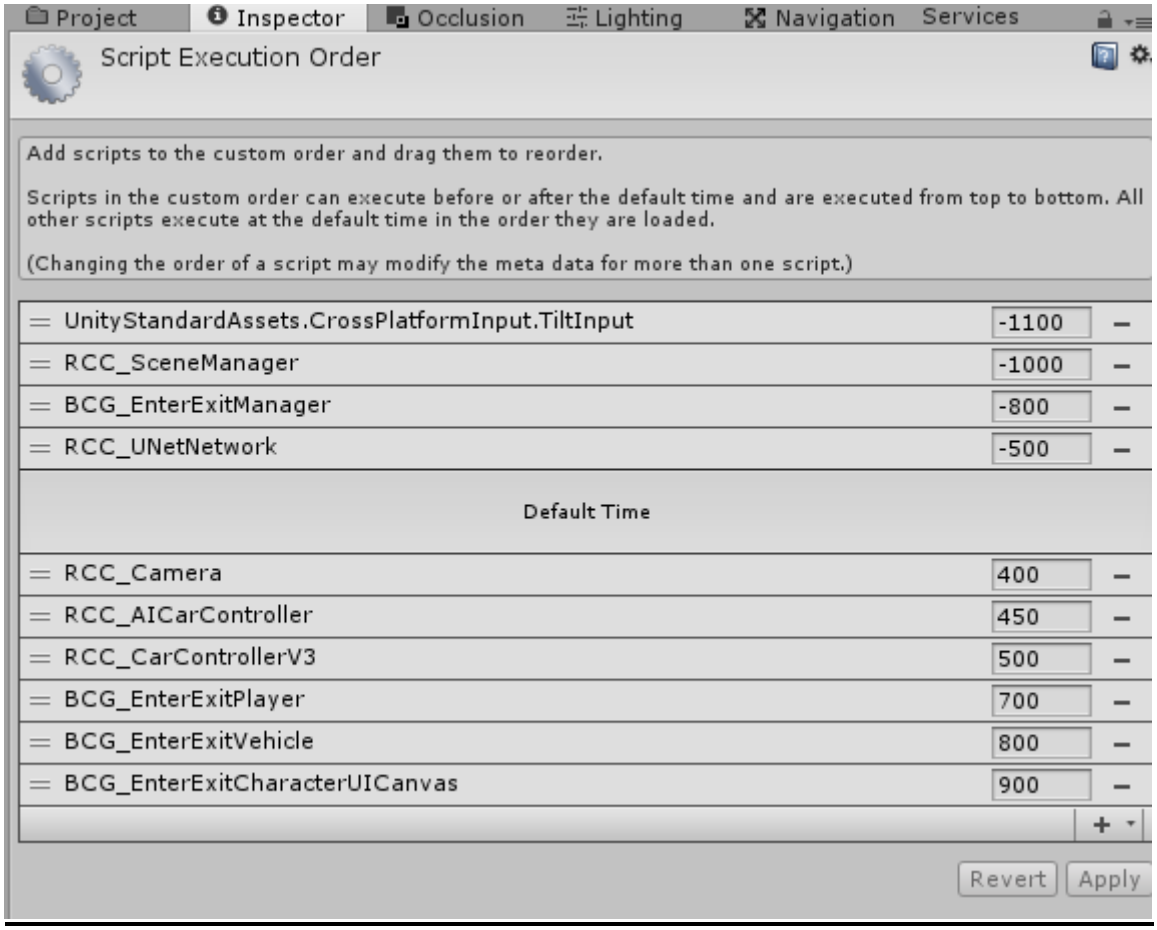




(Tools → BoneCracker Games → Realistic Car Controller → RCC Settings)

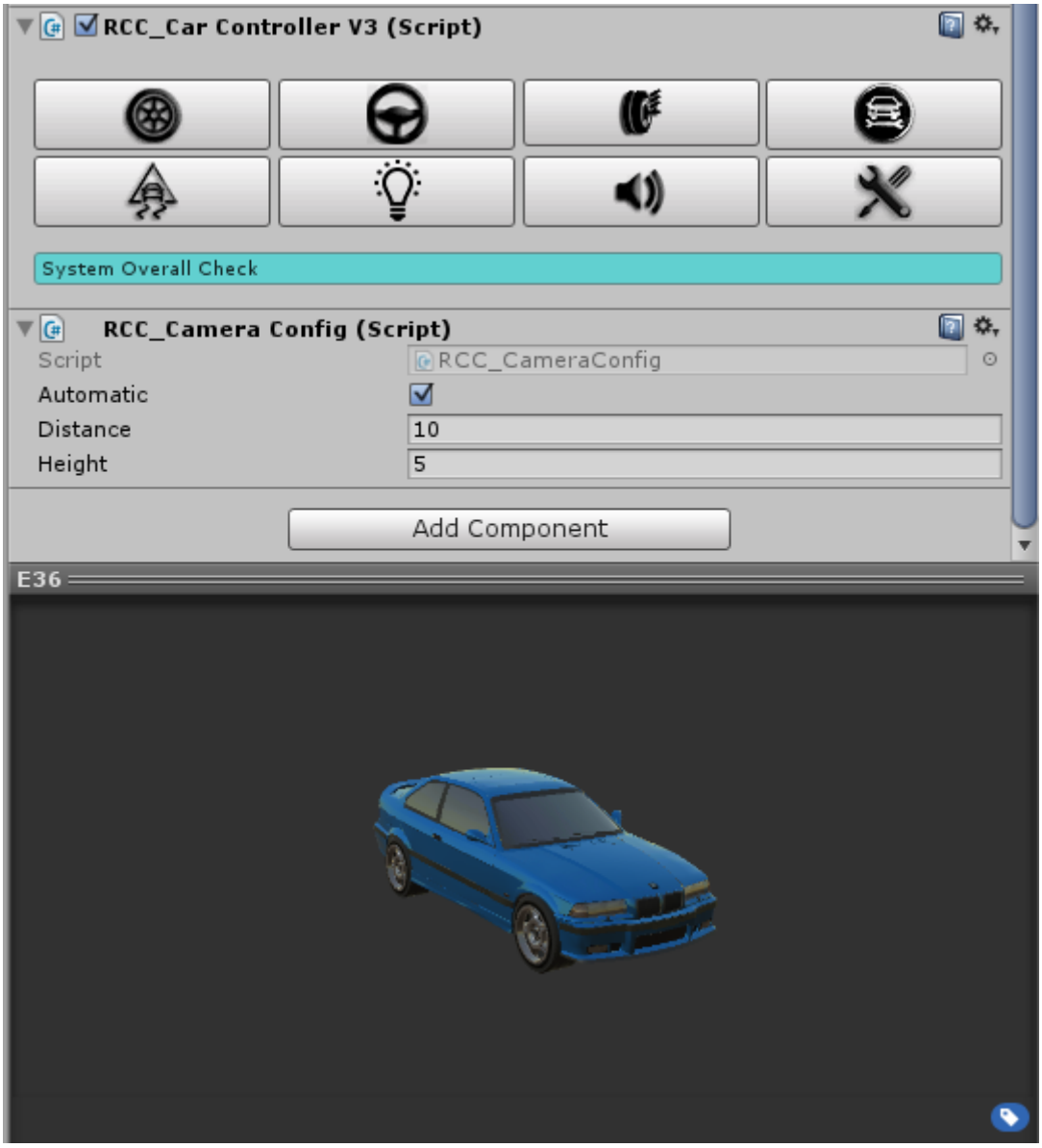
Script Execution Order

This version of the RCC is using Script Execution Order for avoid unexpected event conflicts. This must be imported and doesn't require any action. Just make sure you have this order. You can check it in Edit → Project Settings → Script Execution Order.



General Information about RCC

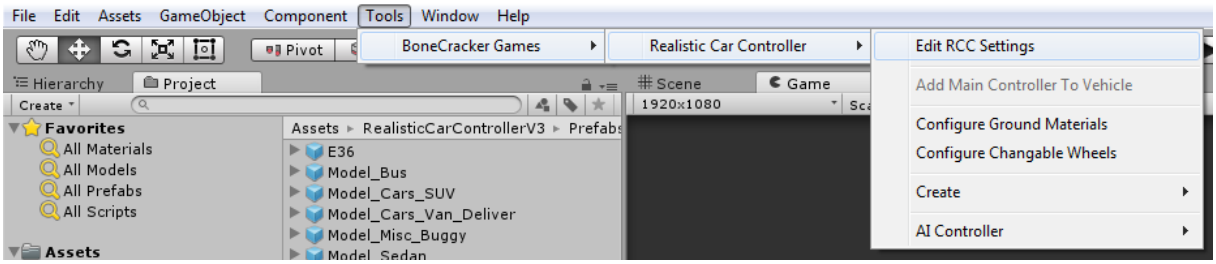
Each vehicle has it's own RCC_CarControllerV3.cs script. Each vehicle is responsible for own RCC_CarControllerV3.cs.



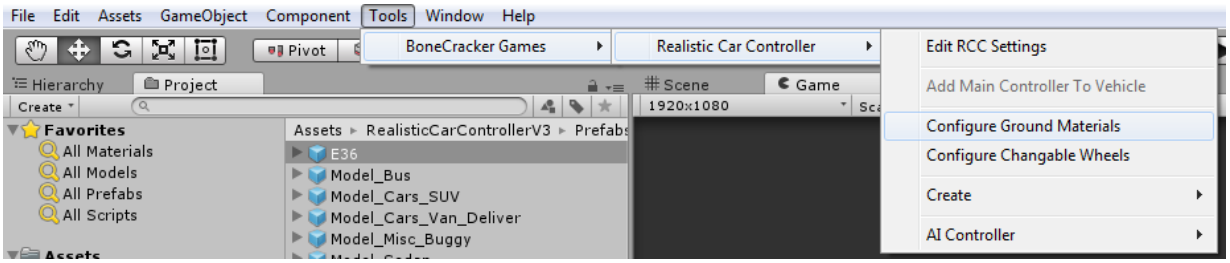
8 Main Categories for Easily and Understandable Creating-Configurating Vehicles.

Wheels, Steering, Suspensions, Mechanic Configuration, Stability, Lights, Sounds, and Damage.

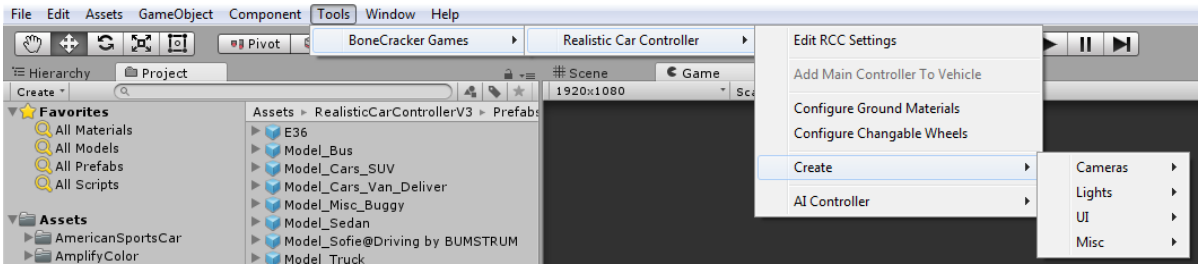
All vehicles are sharing global settings, sounds, configurations via RCC Settings.



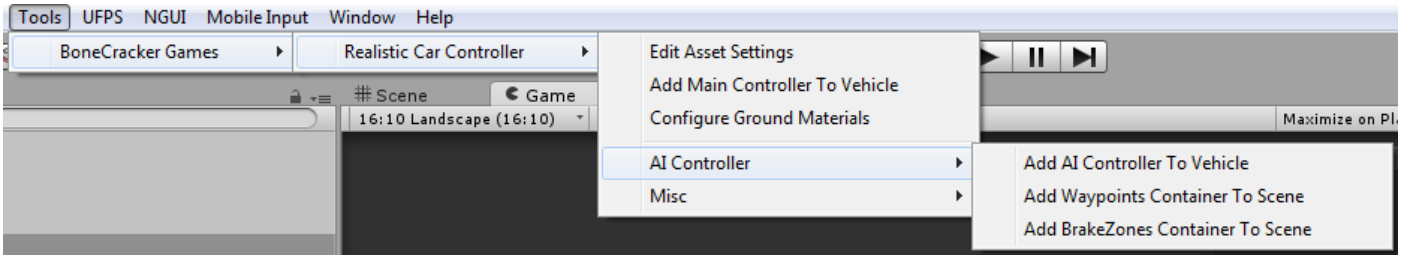
Changing ground materials physics, particles, damps, sounds, etc in Tools → BoneCracker Games → Realistic Car Controller → Configure Ground Materials.



Creating lights, exhausts, mirrors, cameras, etc in Tools → BoneCracker Games → Realistic Car Controller → Create.

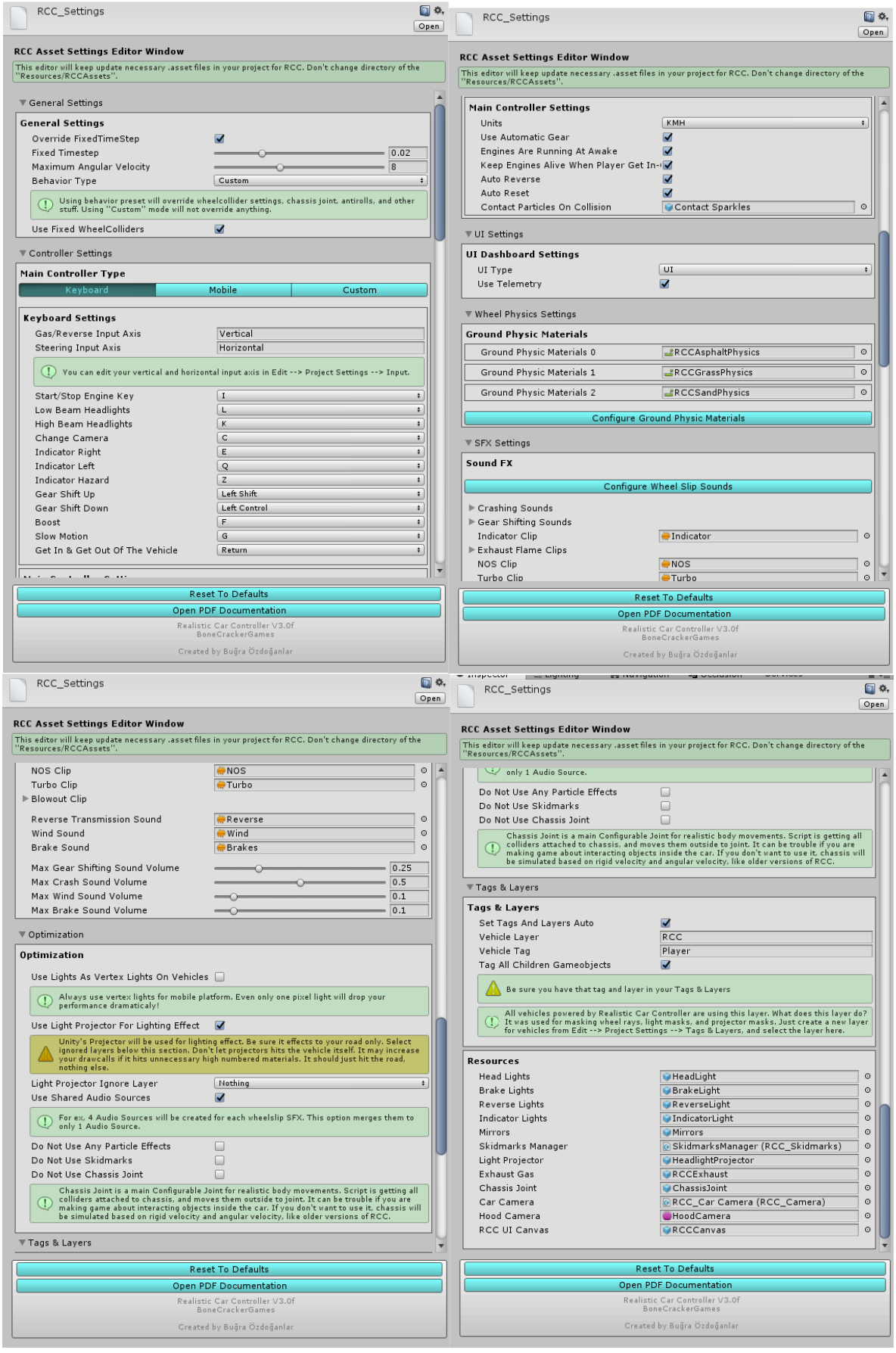


Making vehicles controlled by AI in Tools → BoneCracker Games → Realistic Car Controller → AI Controller.



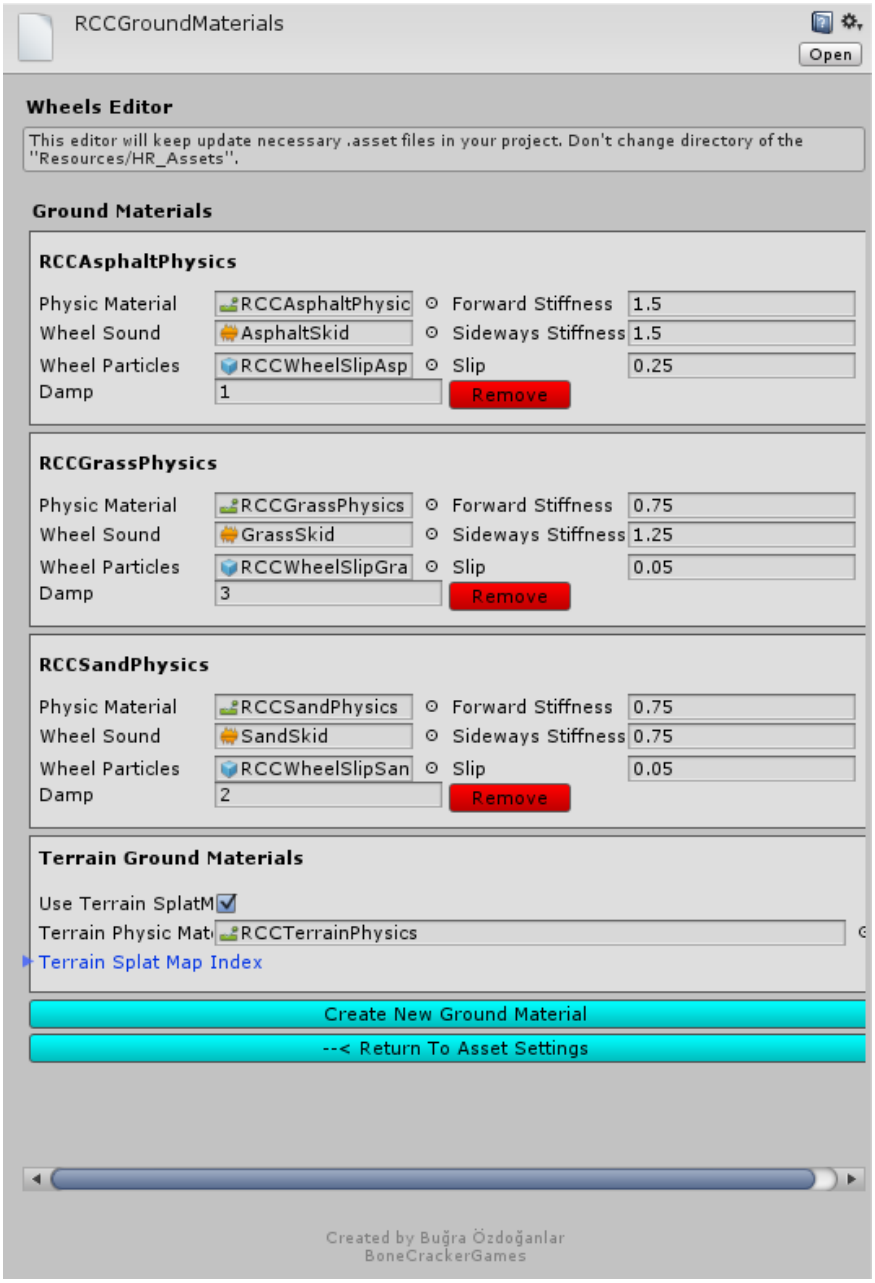
RCC Settings

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



Configurable Ground Materials

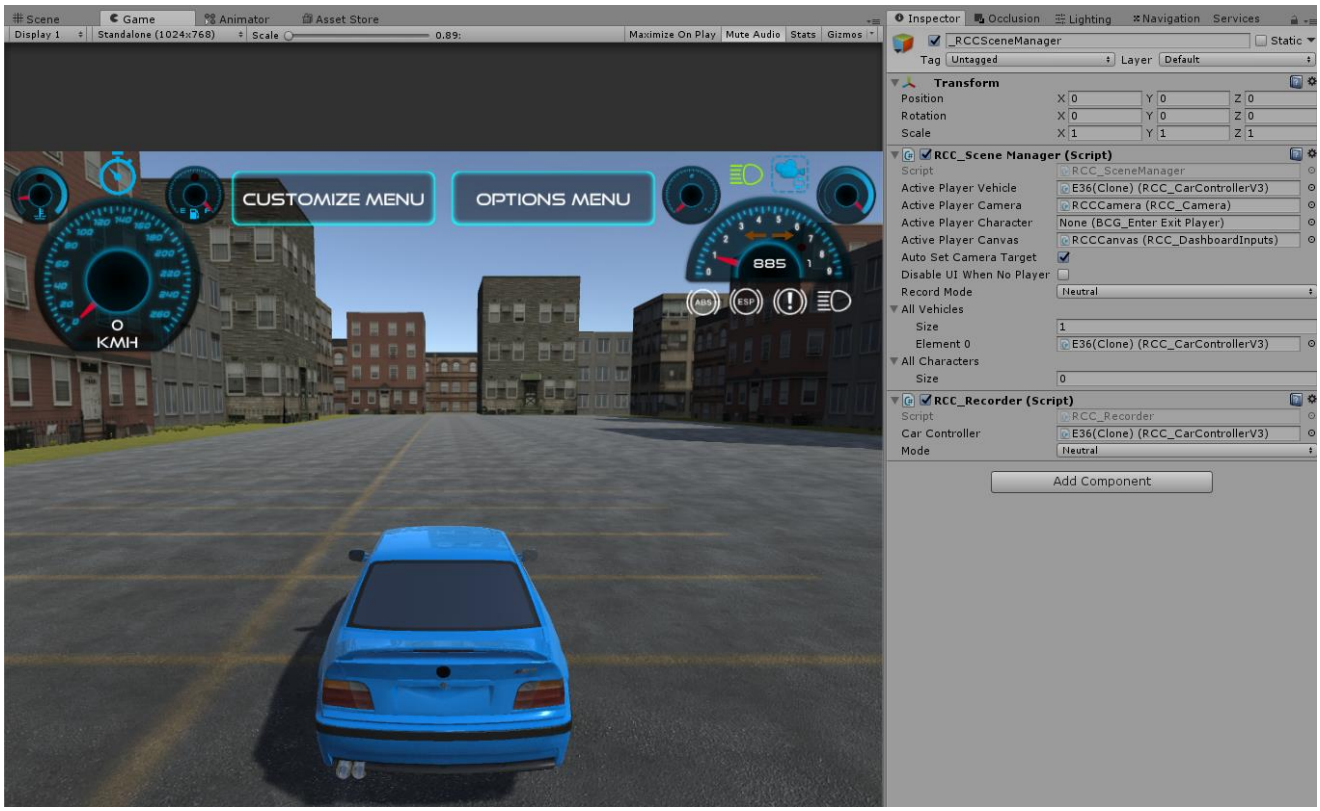
Changing or adding new ground materials physics, particles, damps, sounds, etc in Tools → BoneCracker Games → Realistic Car Controller → Configure Ground Materials.



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

RCC Scene Manager

Every scene will have this manager. RCC Scene Manager contains current player vehicle, current player camera, current player UI, current player character, recording / replay mechanim, and other vehicles as well. Instead of finding current car controller on scene, RCC Scene Manager will find it and manage it only. All other scripts depending on player vehicle will take reference of the RCC Scene Manager.



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 or wrong pivot positions or both of them. This is really painful if you don't know how to fix models directions. 2 ways for fixing this;

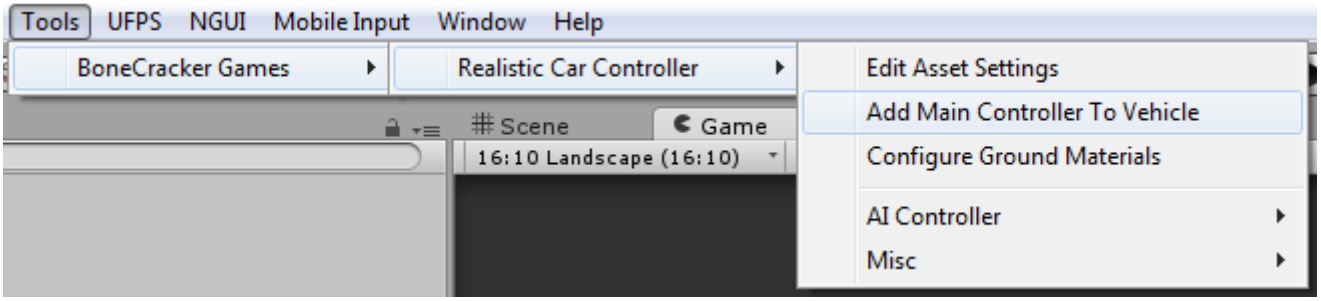
1 – Create a empty Gameobject at the center of the model, and rotate the new Gameobject to correct directions. Then parent your model in to this new Gameobject. It's simple.

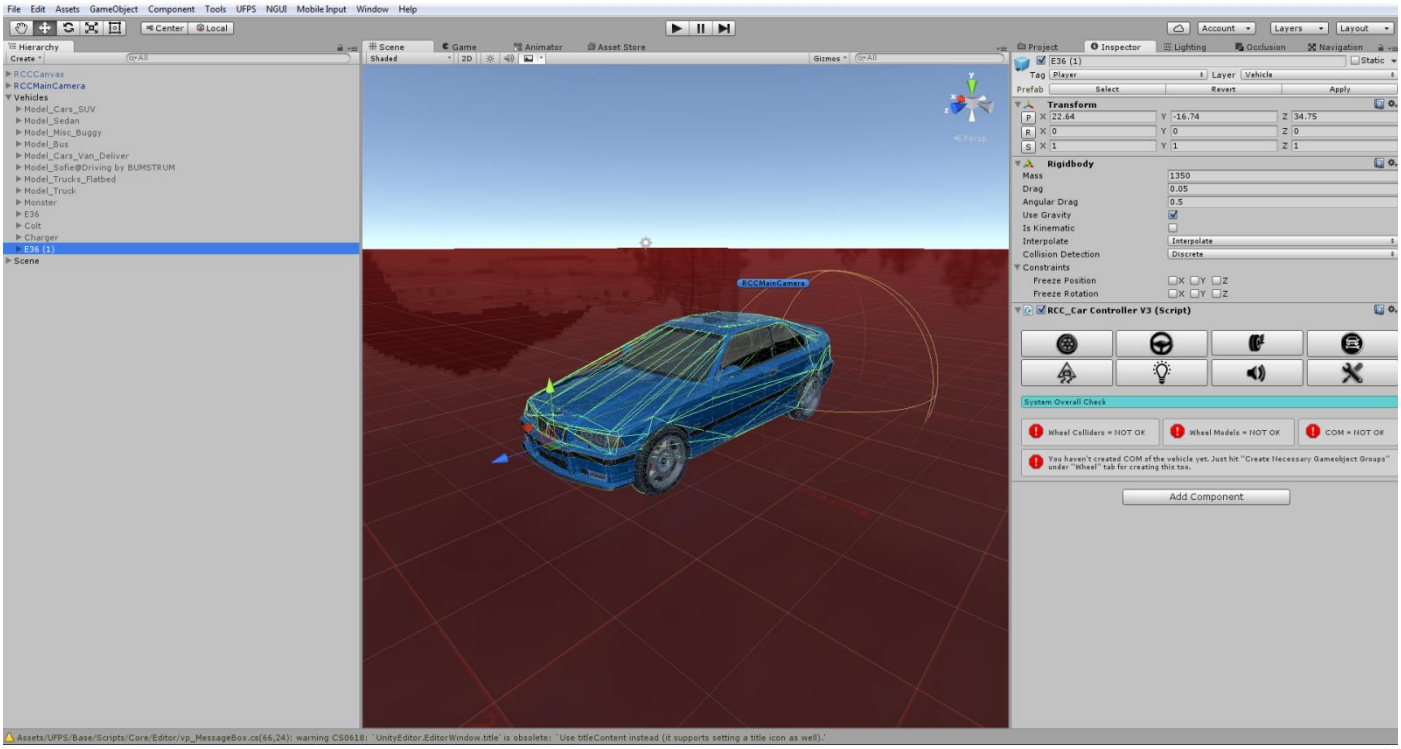
2 – Fix model's directions and pivot positions inside your Designing Software. You can check videos on Youtube for how to do this in 3ds Max.

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. PhysX is calculating size of the collider too. 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 current scene and let's get started;

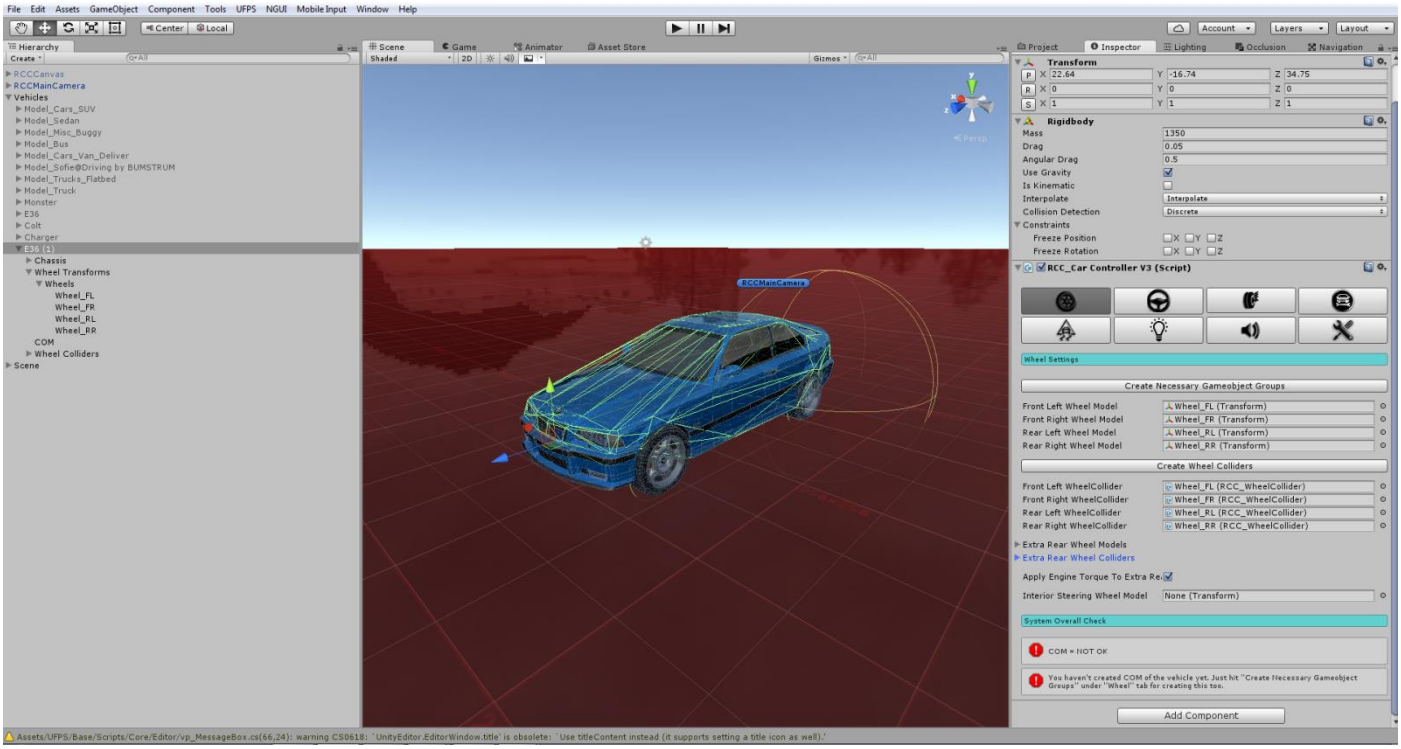
You have to add **Main Controller** to root of the vehicle. Just select your vehicle model on your scene, and click **Tools → BoneCracker Games → Realistic Car Controller → Add Main Controller To Vehicle**.

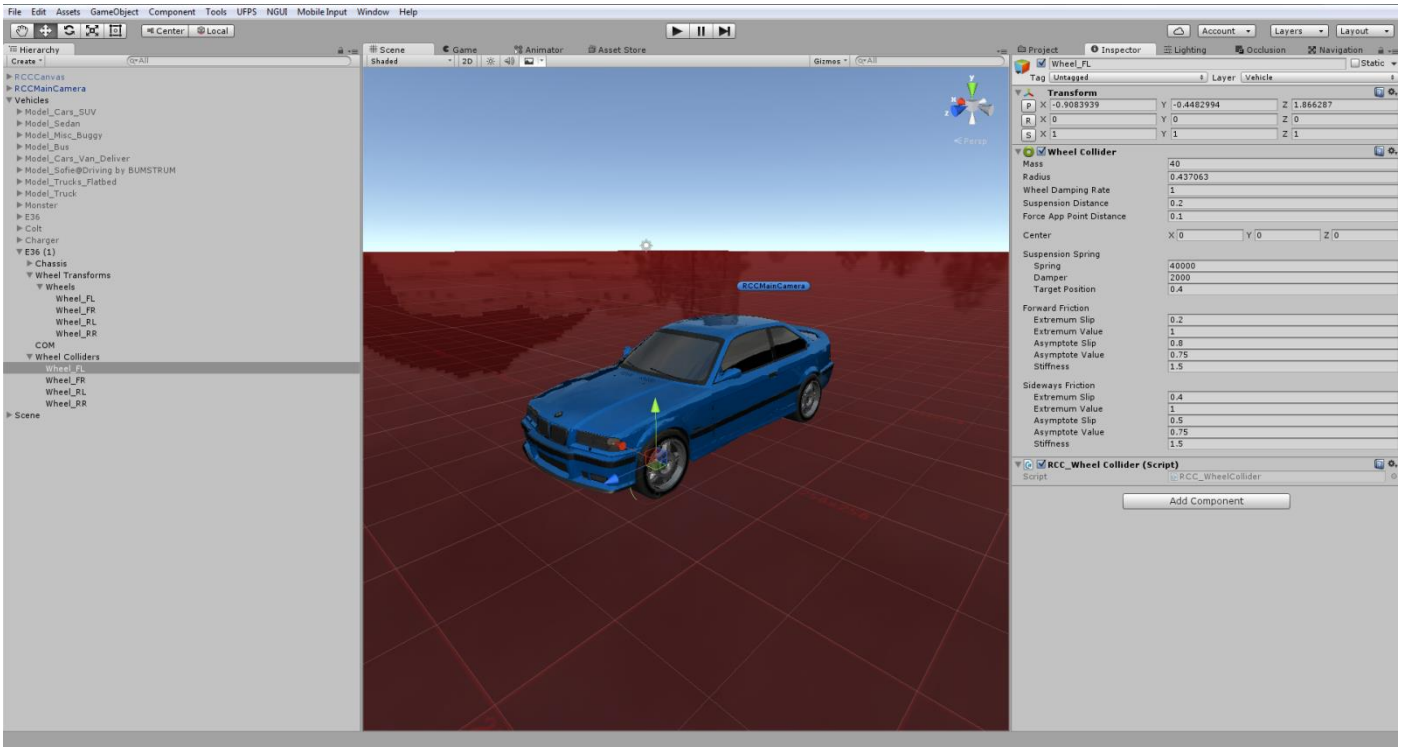




As soon as when you added **RCC_CarControllerV3.cs** script to your car, Rigidbody component will be added automatically. Set your mass to around **1250-1500** for this type of the vehicles. Interpolate Mode = Interpolate, Angular drag is around 0.1 – 0.25 for medium angular velocity. These rigidbody 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 1250-1500 mass vehicles. If you have heavy vehicle such as bus or truck, you must increase Wheel Collider's mass. This value is overrided 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.

Spawning New Vehicles With Code

You don't have to use ~~GameObject.Instantiate~~ for spawning new vehicles. You can spawn new vehicles by just one line of code. You can take a look at API documentation.

Spawning New Vehicles With Given Position, Rotation, Sets It's Controllable, And Engine State

You can spawn new vehicles by;

RCC.SpawnRCC(RCC_CarControllerV3 vehiclePrefab, Vector3 position, Quaternion rotation, bool registerAsPlayerVehicle, bool isControllable, bool isEngineRunning
);

As you can see, you can spawn your vehicle with given configuration by only one line of code.

Registering Vehicle As Player Vehicle

You can register the vehicle as player vehicle by;

RCC.RegisterPlayerVehicle(RCC_CarControllerV3 vehicle);

De-Registering Player Vehicle

You can de-register the player vehicle by;

RCC.DeRegisterPlayerVehicle ();

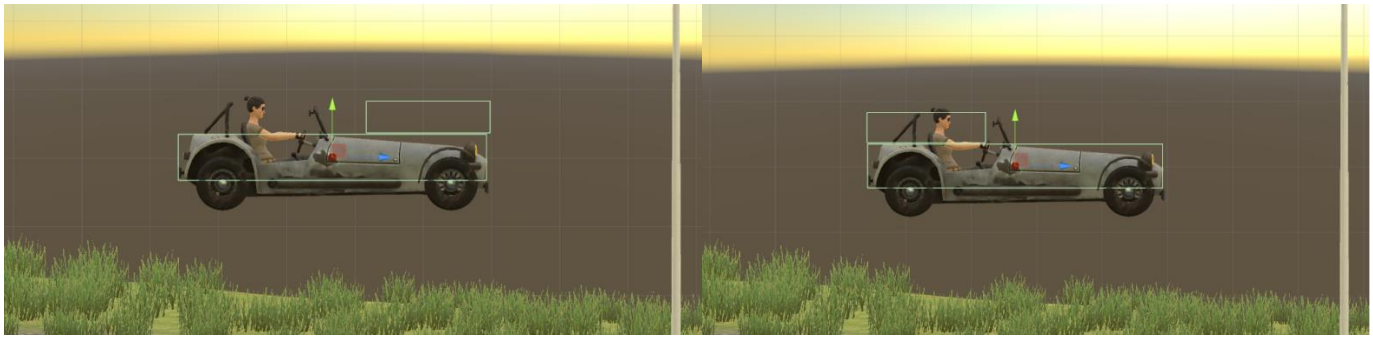
Setting Controllable State Of The Player Vehicle

You can set controllable state of the player vehicle by;

RCC.SetControl(RCC_CarControllerV3 vehicle, bool controlState)

Collider Shapes Of The Vehicle

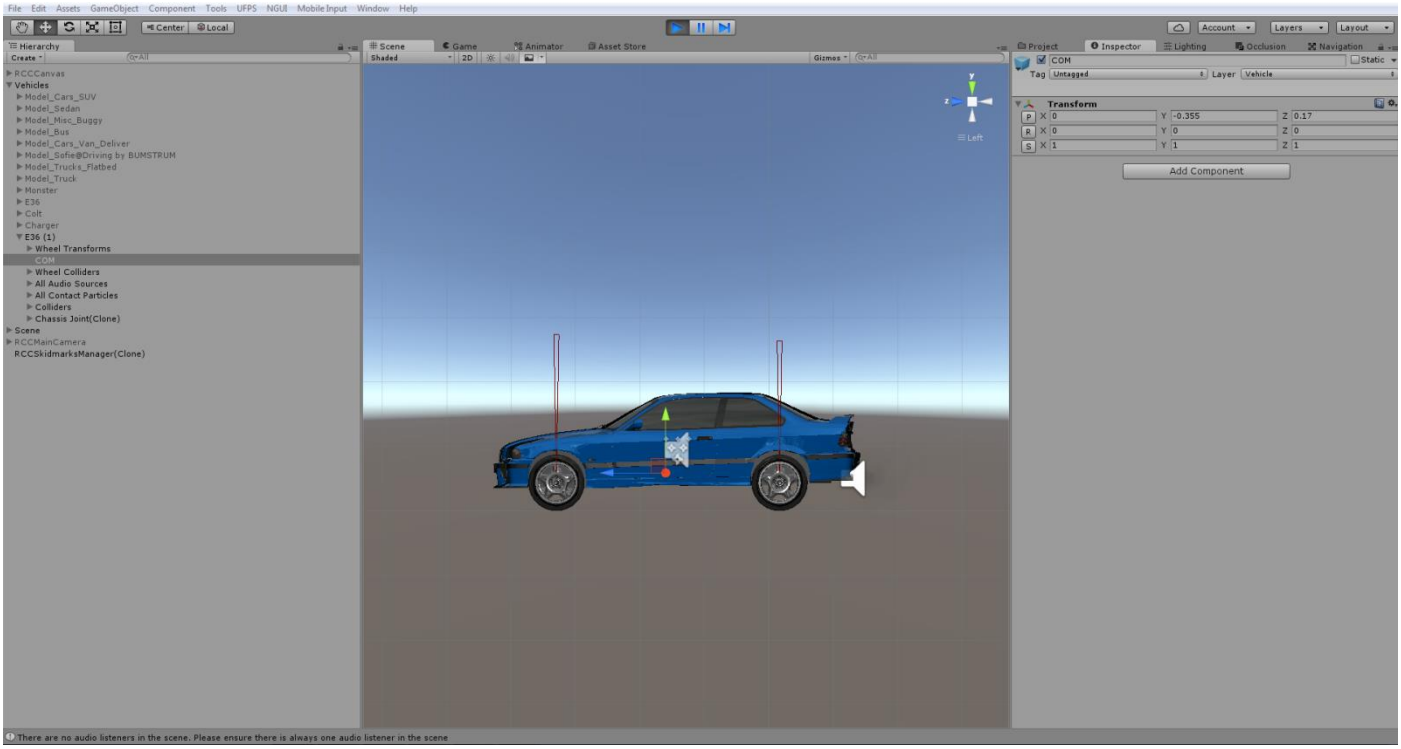
This is one of the most important thing for physics behavior. Most of devs are using mesh colliders for their vehicles. Remember that, shape of your vehicle collider will affect physics behavior directly. Let me explain this;



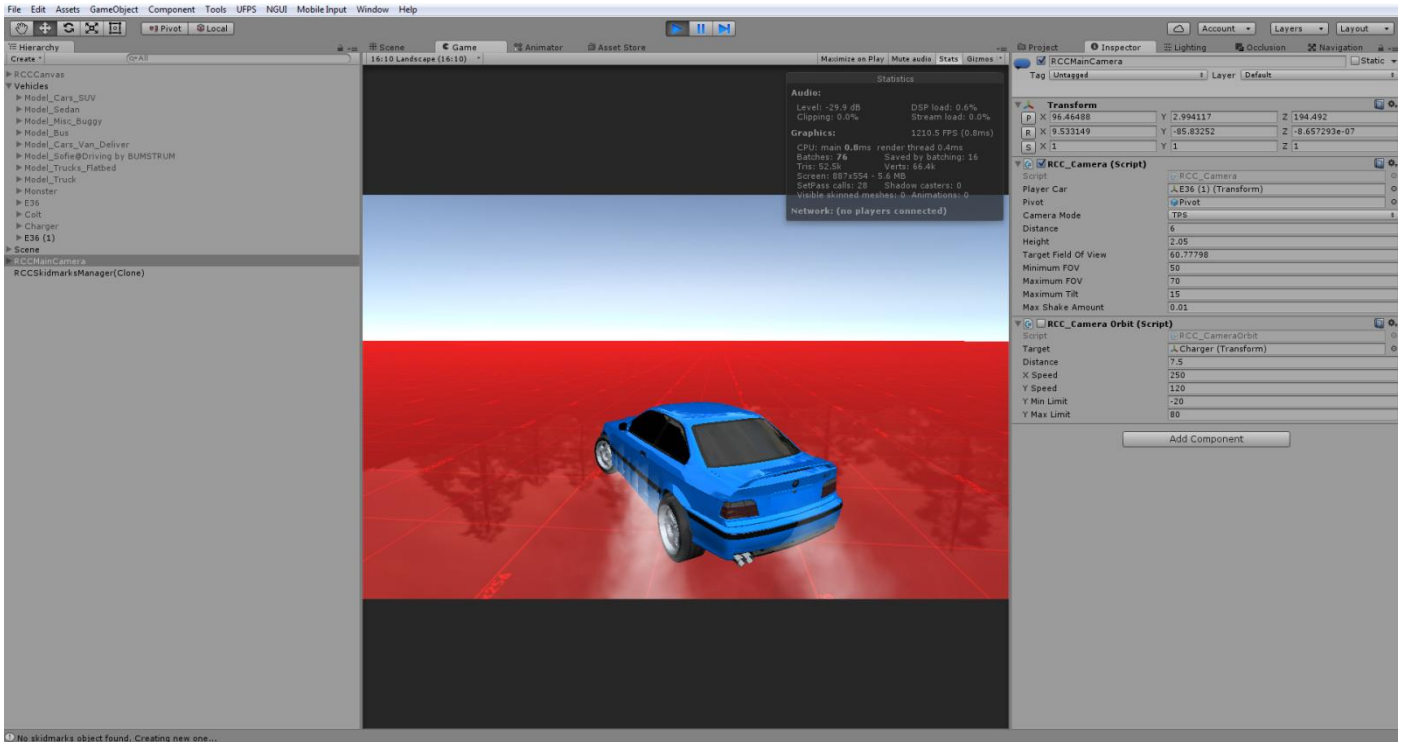
This vehicle has 2 Box Colliders. Second Box Collider is placed at front of the vehicle here on first screenshot. This will bring you more controllable and stable vehicle. Usually, second Box Collider should be at center Z of the vehicle for this model. But this is high speed vehicle, so it must be stable. 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, keep in that your mind.

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.



Configurate 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.)

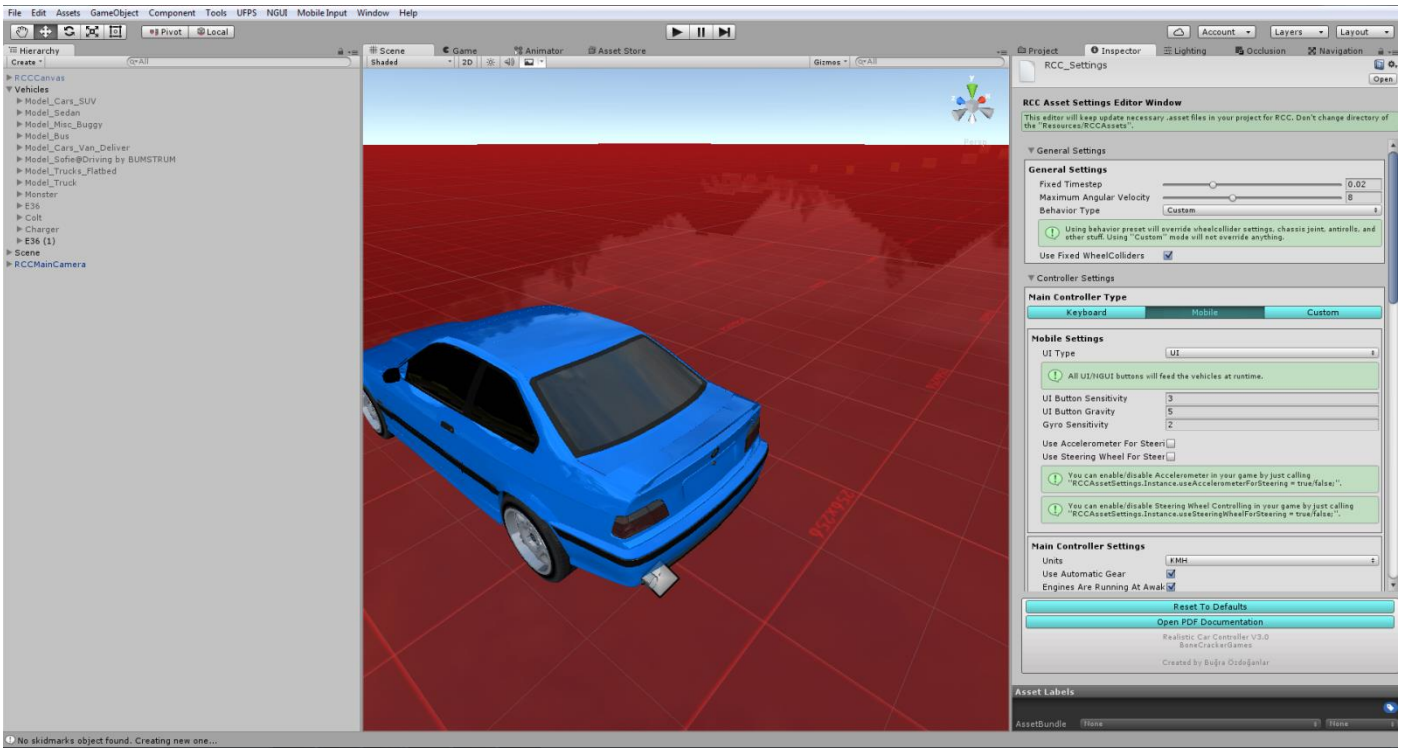
Mobile Controllers

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" prefab in RealisticCarControllerV3/For NGUI. Drag and drop the RCC_NGUIRoot prefab to your Hierarchy.

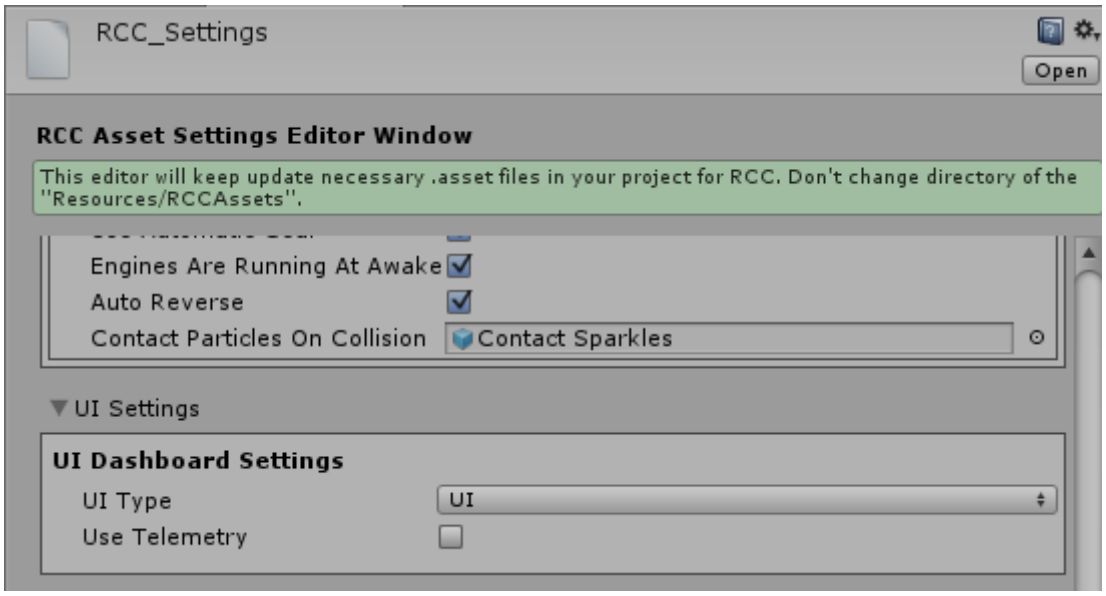
This prefab includes NGUI Controller, NGUI Dashboard, and other necessary scripts for ready to use. Canvas is designed for all aspect ratios. Controller elements won't disappear or change positions on other aspect ratios or screen resolutions.

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 mobile shaders in **RCC City Mobile** scene. If you build an APK without editing materials, you will get performance loss.

Dashboard Configuration

You can access dashboard from RCC Settings.



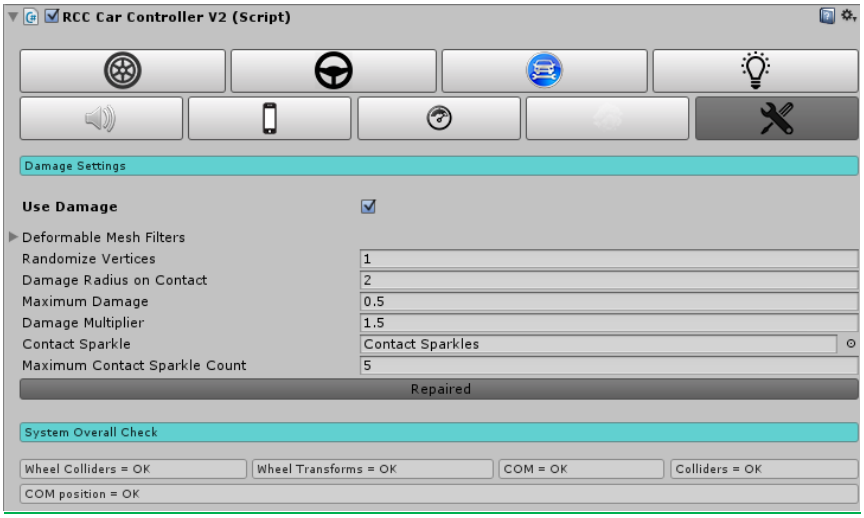
Default Dashboard Type is "UI". Script will access "RCC_DashboardInputs.cs" and "RCC_DashboardDisplay.cs" scripts for displaying dashboard. These scripts are attached on;

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 legacy 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 unrealisticly.

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 → Lights.

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

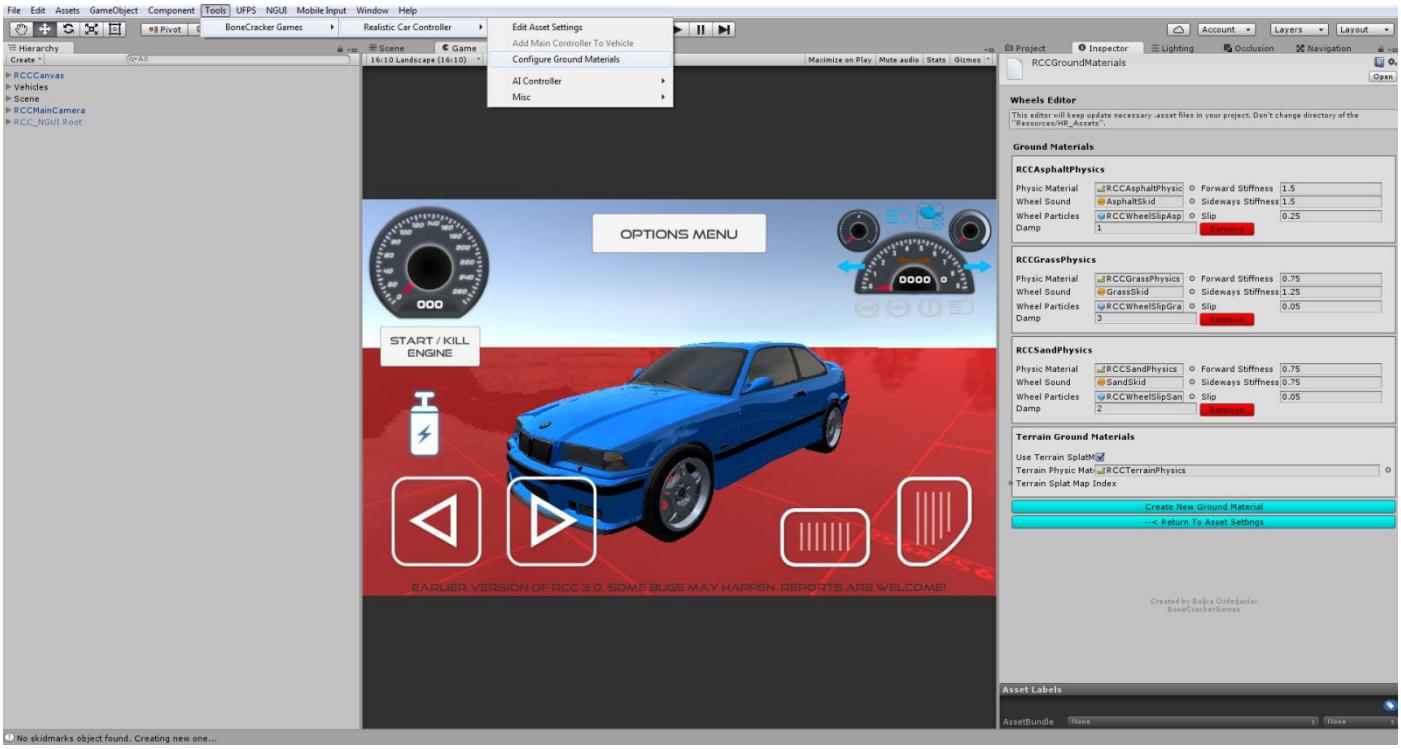
You will find "RCCWheelSlipAsphalt", "RCCWheelSlipGrass" and "RCCWheelSlipSand" prefabs under Prefabs folder. You can use your own smoke prefab as you wish.

If you want to use exhaust effects, you can create it from Tools → BoneCracker Games → Realistic Car Controller → Create → Misc. 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

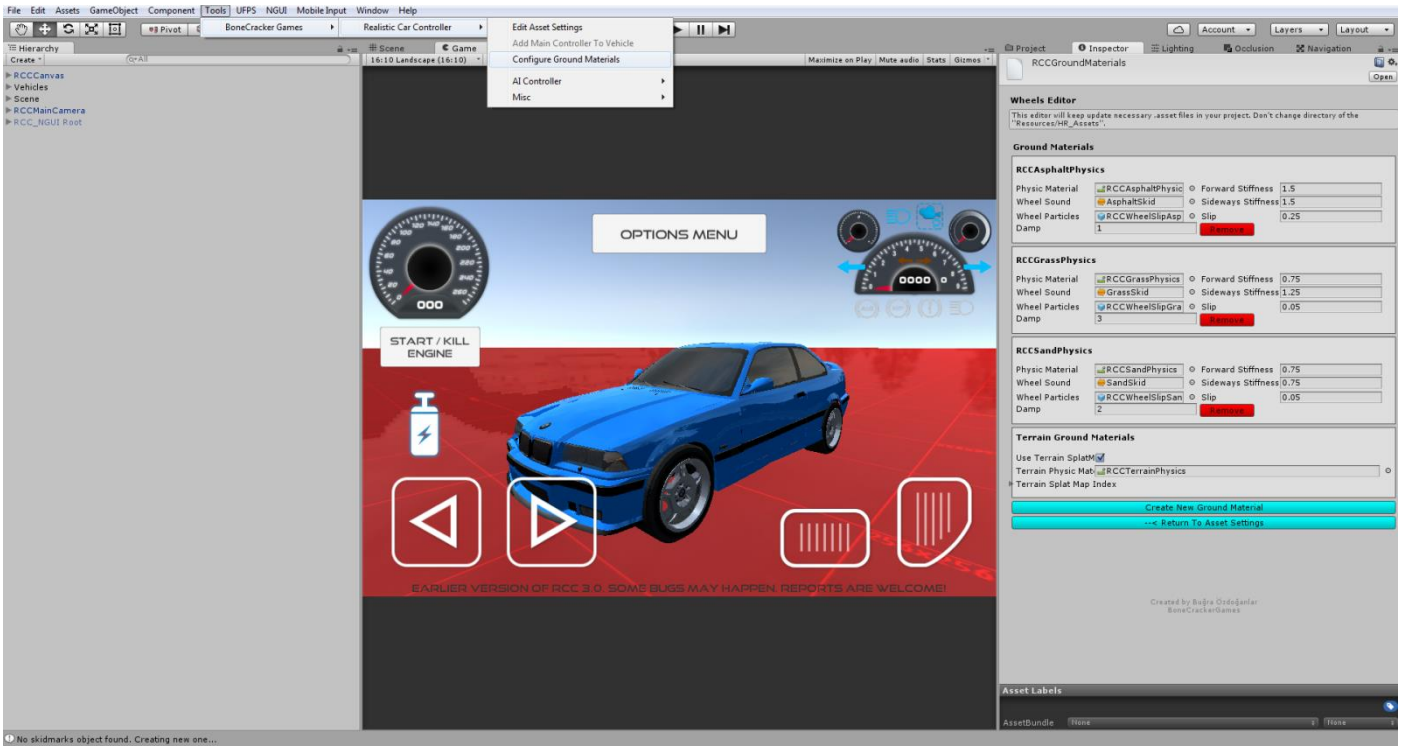
If you want to get variable tire grips on varied surfaces, you can use this feature. Open up Tools → BoneCracker Games → Realistic Car Controller → Configure Ground Materials.



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

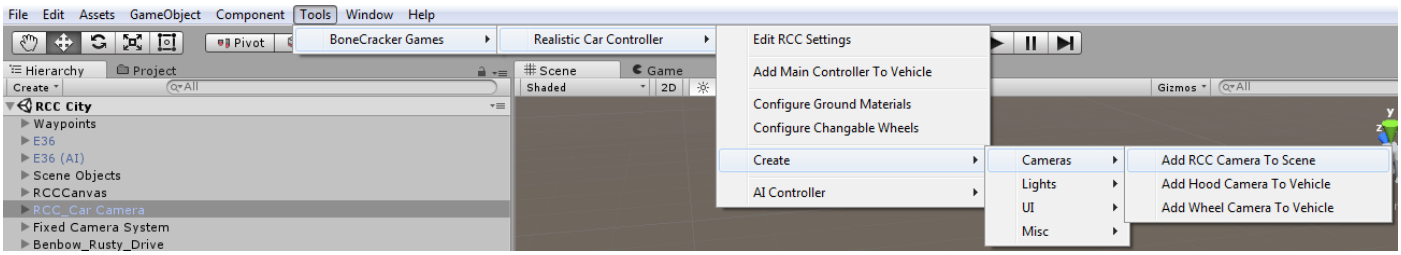
You can adjust each materials Ground Particles, Wheel Sounds, Damp, Forward and Sideway Stiffness, Slip here.

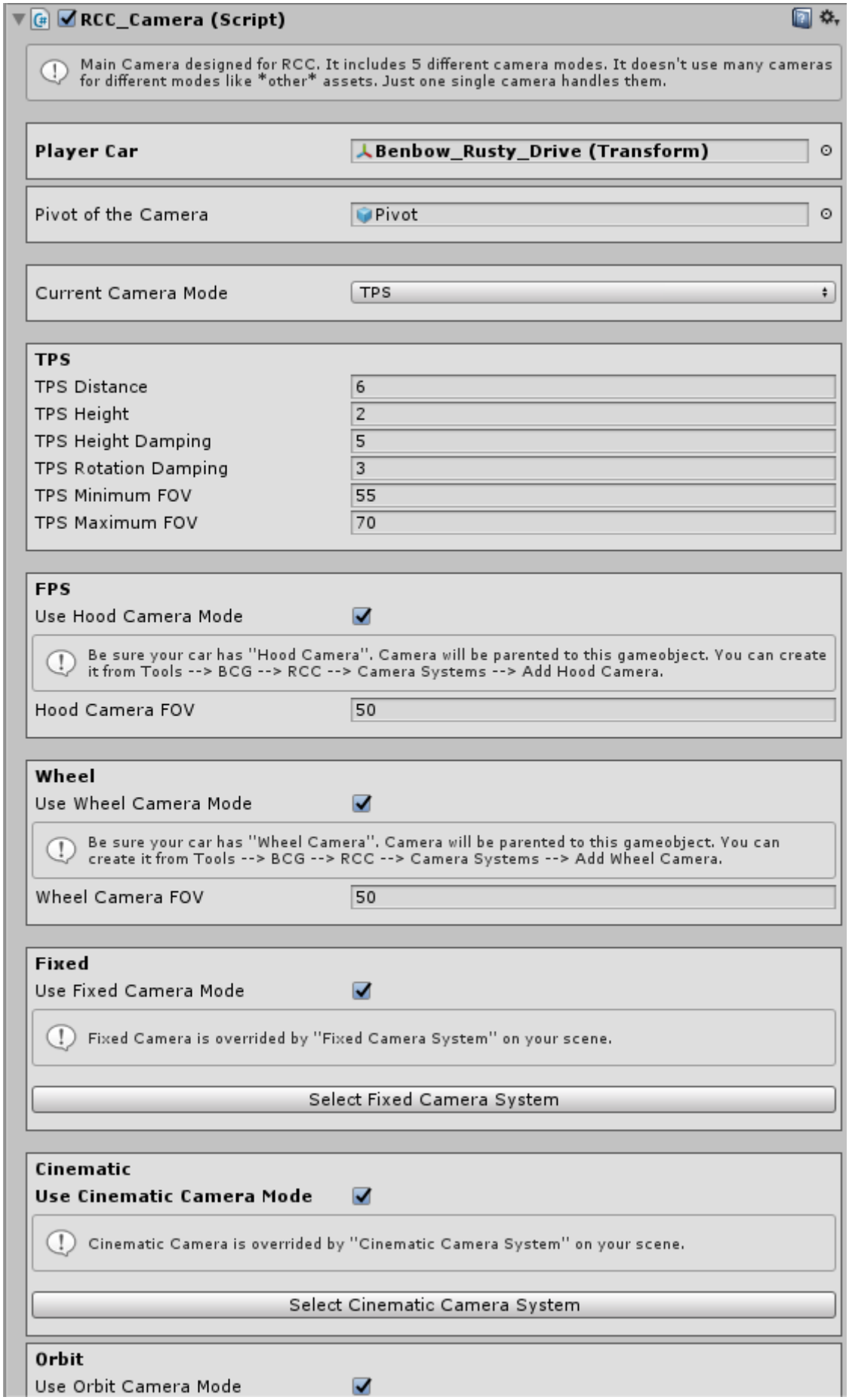
As i said, these are optional effects. If you don't want to use them, just leave.

RCC Camera System

Main Camera system desiged for using with RCC. Includes 6 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
Benbow_Rusty_Drive (Transform)
Pivot of the Camera
Pivot
Current Camera Mode
TPS
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
50
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
50
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

Record / Replay

Complete physics based record / replay system. Only player vehicle can record / replay. All you have to do is press "**R**" for start recording, and "**P**" for start replay. These keys can be changed in RCC Settings. There is a UI button for mobile.

RCC_Recorder can be found in _RCCSceneManager on your scene. Script will be added at awake, or you can add it by manually. It has 2 public methods for starting record / replay. You can use them if you want to start record / replay at any time you want. For ex;

```
RCC_SceneManager.Instance.gameObject.GetComponent<RCC_Recorder> ().Record ();
RCC_SceneManager.Instance.gameObject.GetComponent<RCC_Recorder> ().Play ();
```

Customization

You can customize your vehicles by just calling a single method. Please take a look at "Realistic Car Controller V3 Scripts" documentation. All methods in RCC_Customization 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;

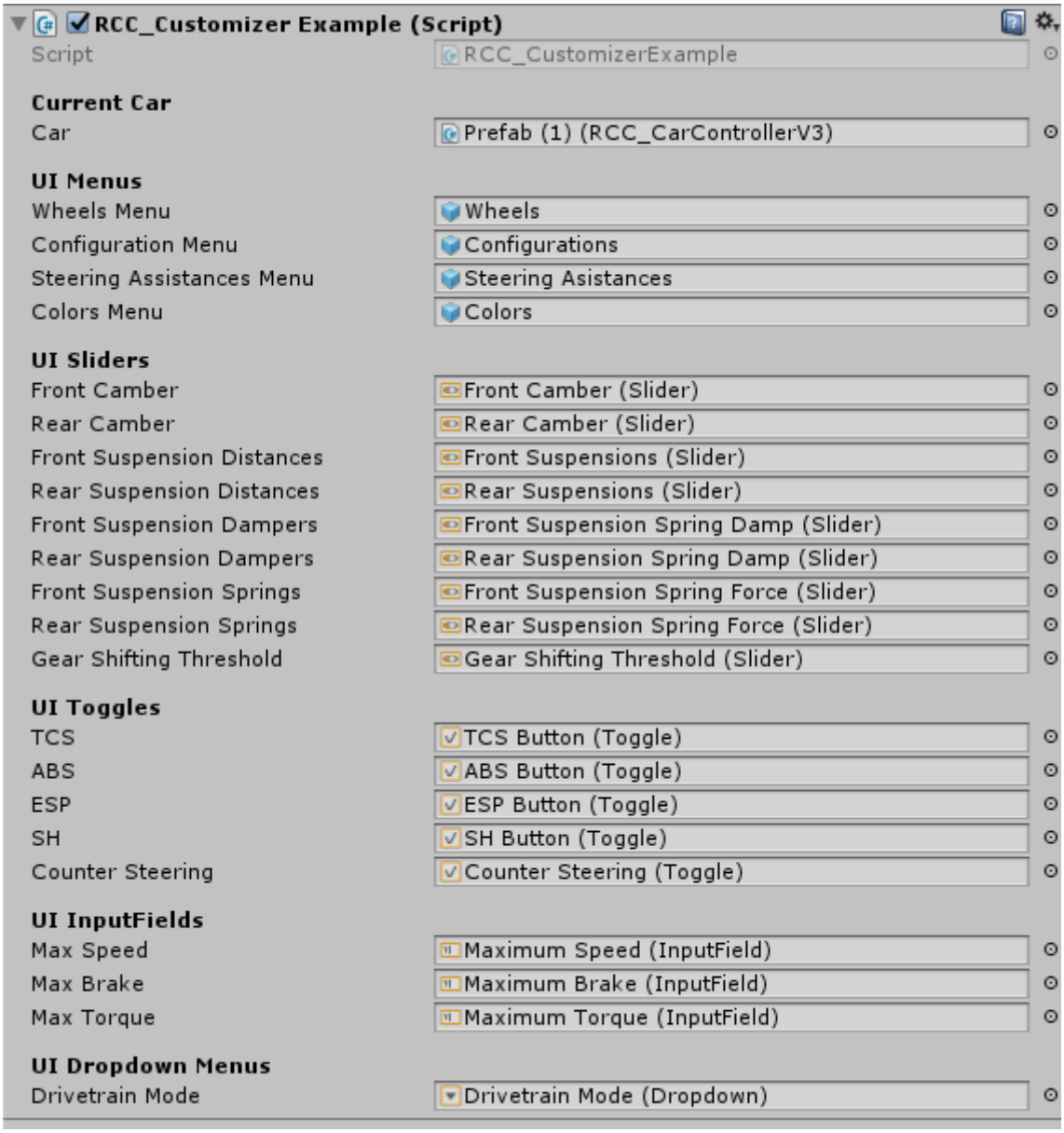
```
RCC_Customization. RepairCar (targetRCC);
```

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.

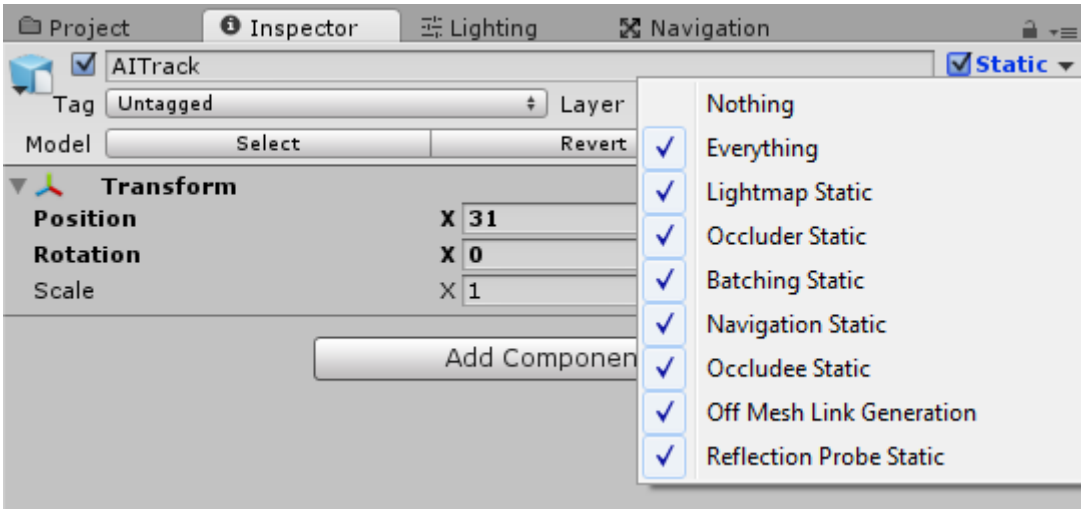


This example script handles all UI menus, buttons, sliders, toggles, inputfields, and dropdown menus. It just receives inputs from UI, and fires necessary action.

AI Configuration

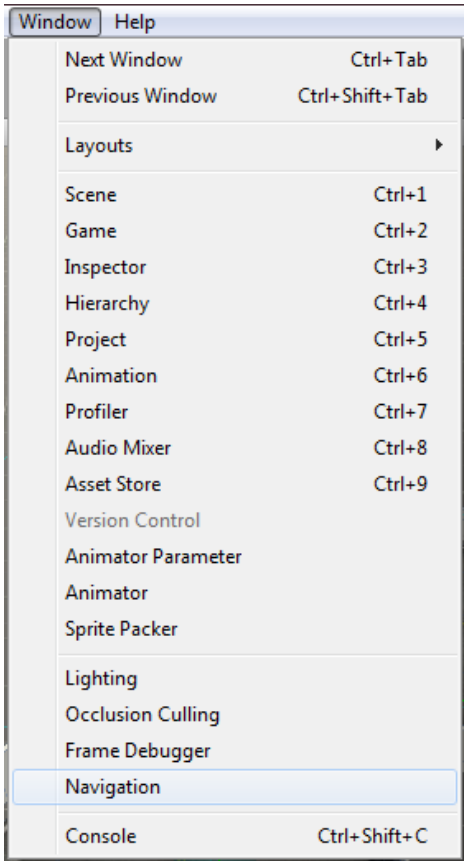
Creating NavMesh For Scene

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**".

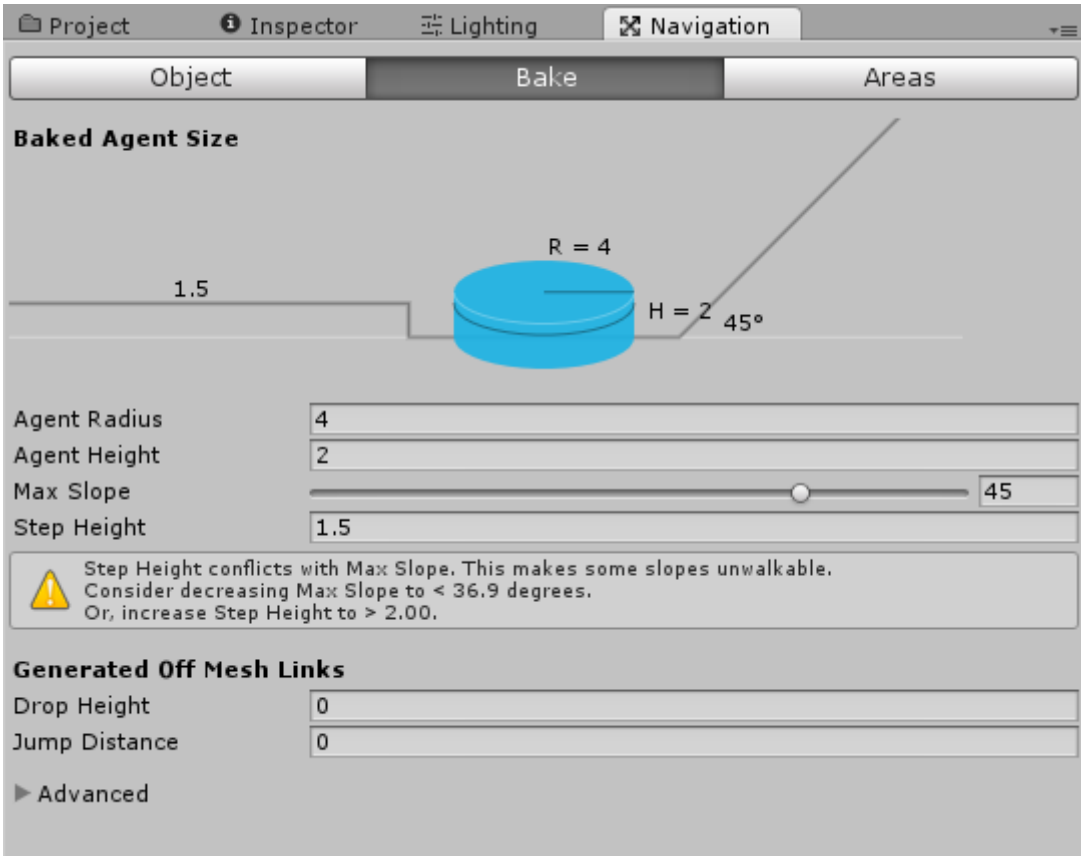


When all your static objects are marked as "Static", then you can bake your navigation mesh.

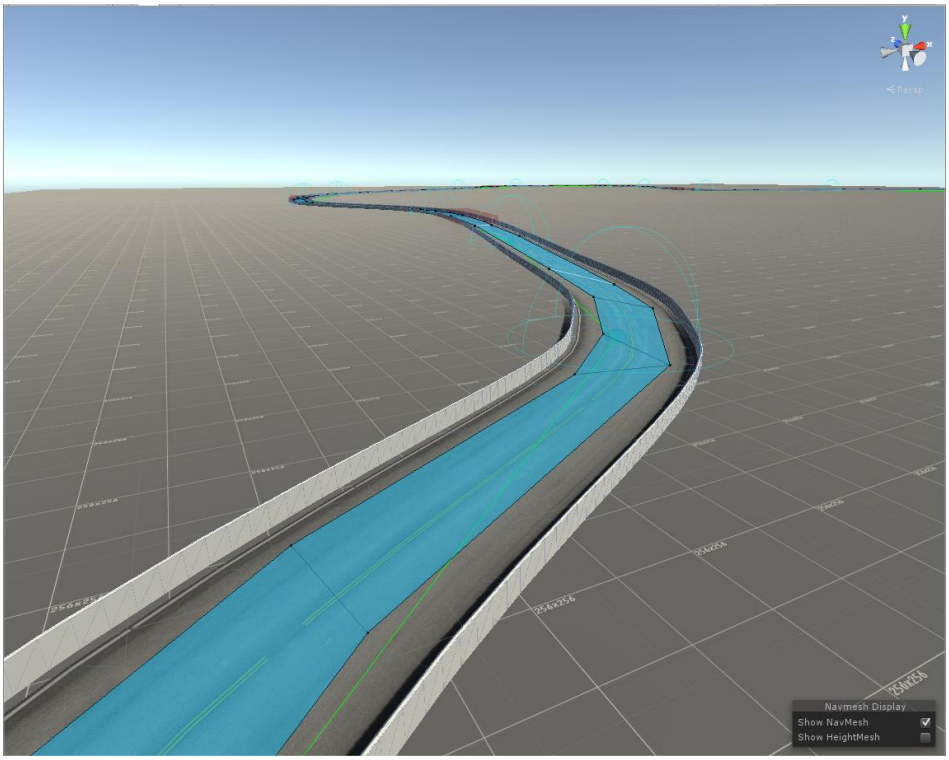
Open "**Navigation**" window from **Window → Navigation**.



Default settings should be like this;

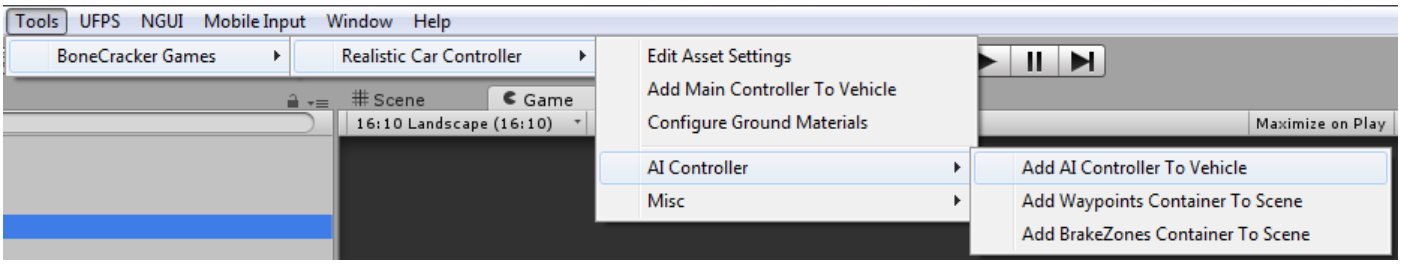


And then, click the bake button and bake your scene. Check your blue navigation mesh. Should be like this;

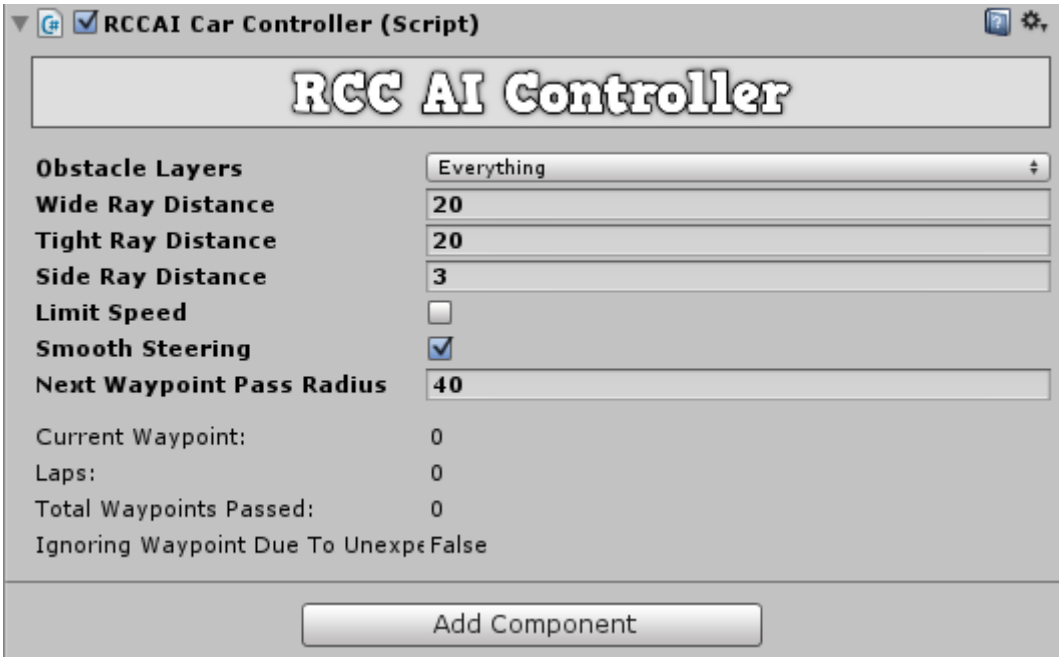


Adding AI Controller To Vehicle

First, build and configurate 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"

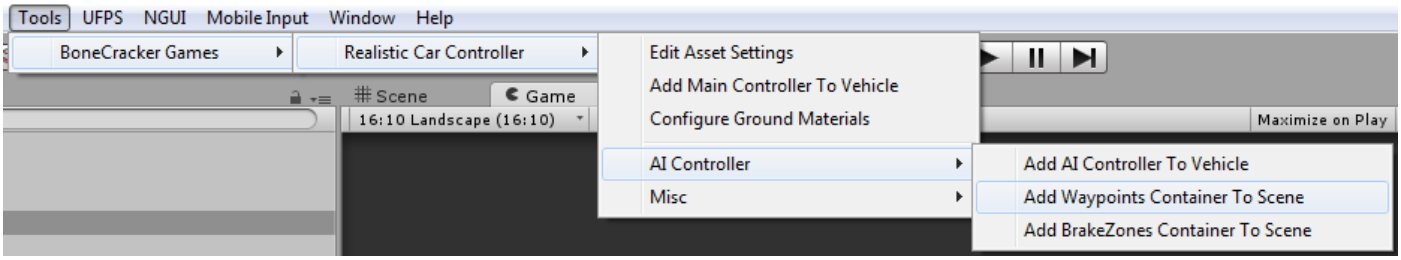


This will add "RCC_AIController" to root of your vehicle.

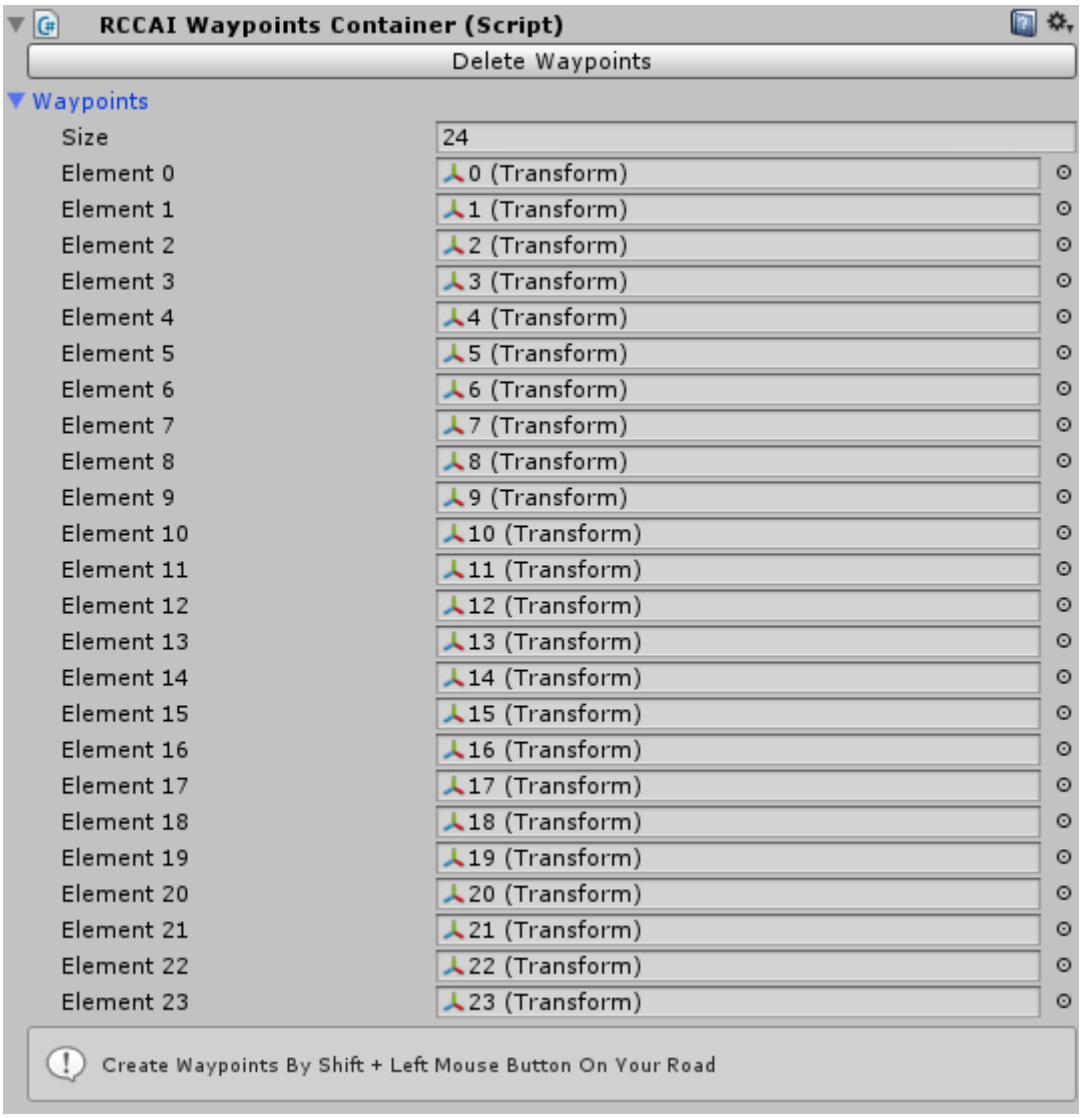


Vehicle will use "Nav Mesh Agent" for road path based on your waypoints, and will use raycasts for dynamic objects. If you have specified gameobjects for ignoring raycasts for this gameobject, you can select your gameobject layer from obstale layers.

Adding Waypoints Container To Scene

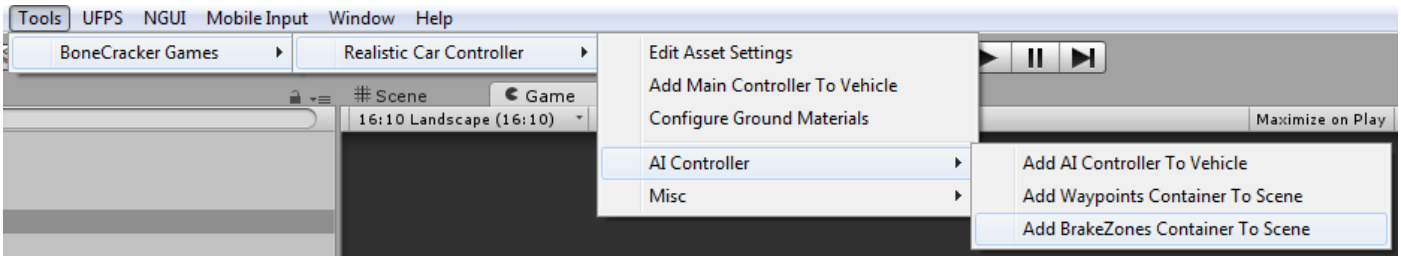


This will add "RCC AI Waypoints Container" to your scene. Simply hold Shift and left click on your road to create a new waypoint. Create your path with them.

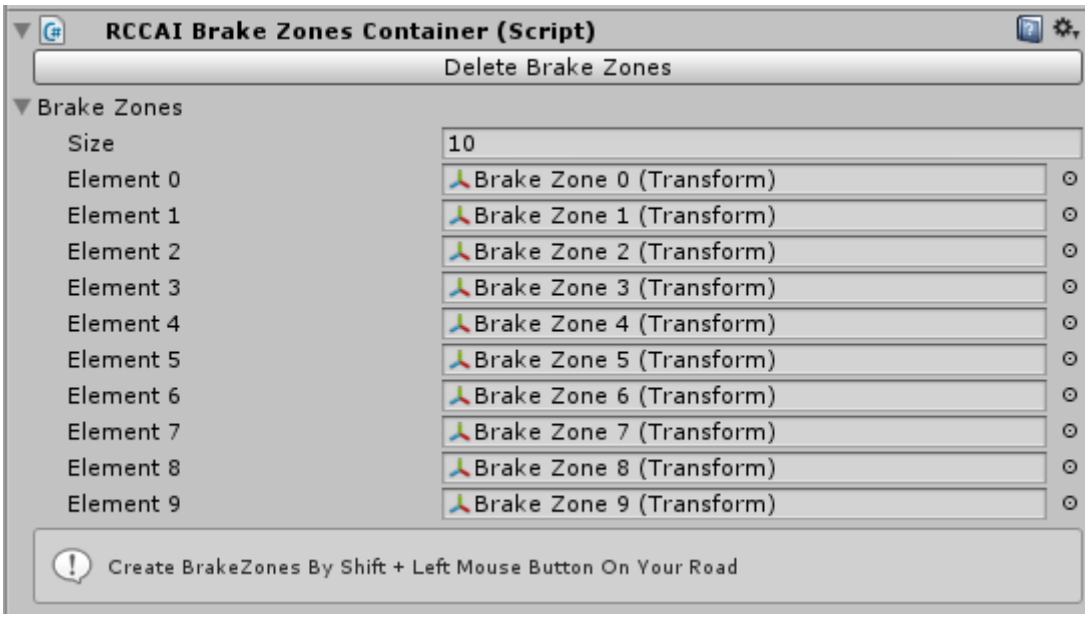


Info: Do not use CTRL+D for duplicate any waypoint.

Adding BrakeZones Container To Scene



This will add "RCC AI BrakeZones Container" to your scene. Simply hold Shift and left 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 is moving backwards!

A. Your vehicle model has wrong axes. Z must be looking at forward.

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 in 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 (expalined below), and then import "RCC_PhotonNecessaryScripts" in Scripts/Photon folder. Now you can test the Photon demo scene.

There is a scene named "**RCC City Photon**" in Demo Scenes folder. Same scene with regular city scene. Only difference is this scene has **Photon Network Manager**.

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.

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 V3.2");”

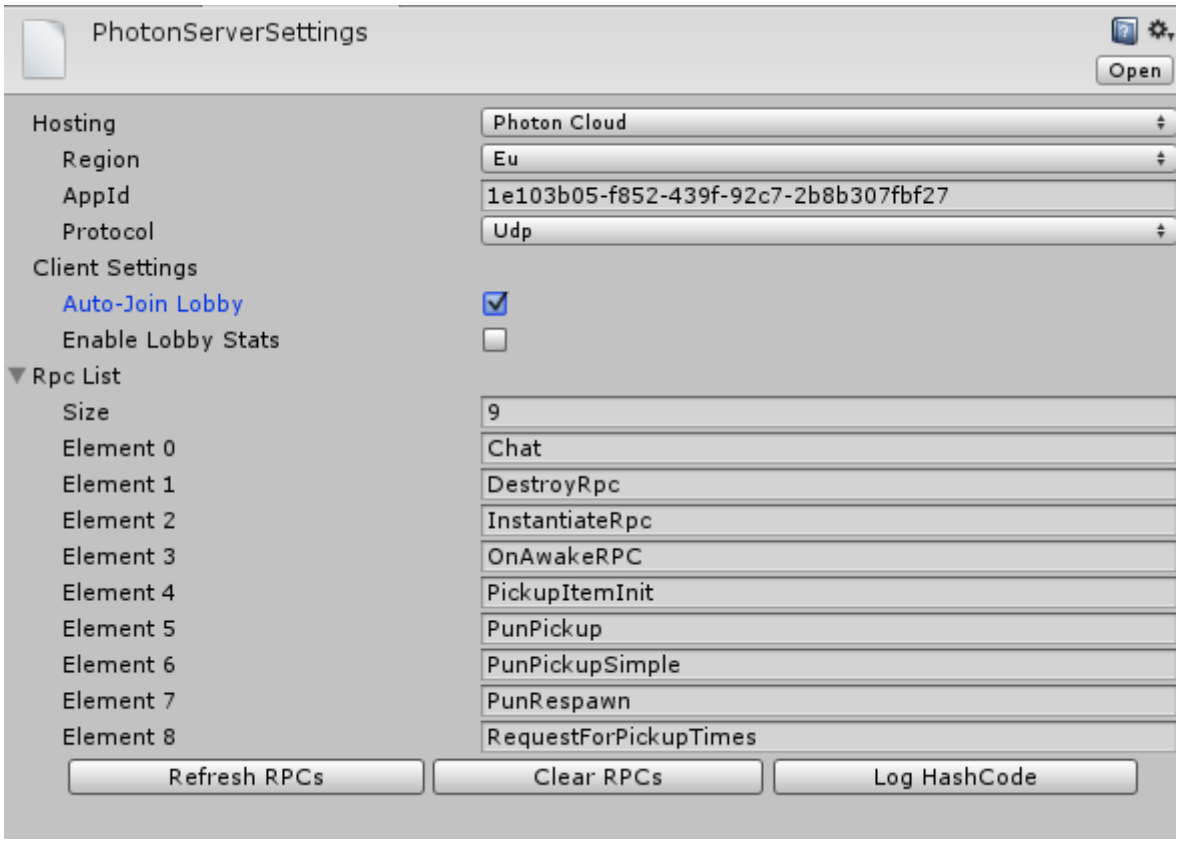
We are connecting to the server first. We can listen which connection status we are on in OnGUI() method. Like this;

“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 active 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**.

Multiplayer with UNet

There is a scene named "RCC City UNet" in Demo Scenes folder. Same scene with regular city scene. Only difference is, this scene has Unity's Network Manager and Network Spawn Positions.

When you joined a game or started host, Network Manager will spawn your vehicle prefab on designated spawn point. Your spawned vehicle will have a script named "RCC_UnetNetwork.cs" in action. This script will send/receive transform, inputs, configurations, and lights of the vehicle on realtime. Haven't used any single SyncVar. All of them organized, and optimized. All of your actions on vehicle will act on other side immediately.

Also haven't used Unity's Network Transform component for transform syncronization. "RCC_UnetNetwork.cs" has its own methods for calculating positions and rotations depends on vehicle velocity. This brings more smooth, lag free, and accurate data of the transform. And also collisions are definitely more accurate against it.

It's simple. If vehicle is in your hands, "RCC_UnetNetwork.cs" will send all of your data to the server. If it's not, script will receive all data from the server and feed the RCC.

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. You can't redistribute / resell them.

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 by 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