



Realistic Car Controller 2.9

Package Includes

8 Configured Vehicle Prefabs for READY TO USE,

1 Driver Animated Vehicle by BÜMSTRÜM,

Configured UI and NGUI Controller for Mobile, and Dashboard Prefabs,

5 Demo Scenes for Presenting UI Mobile Controller, NGUI Controller, AI Controller, FPS Enter-Exit,
and Desktop Controller,

User Friendly Editor Scripts,

AI Controller,

All Necessary Scripts,

Extremely Easy To Use,

Engine Sounds, Gear Sounds, Crash Sounds,

And more...

You can find more updated details on

<http://www.bonecrackergames.com/#!Realistic%20Car%20Controller%20Documentation/c1z2r>

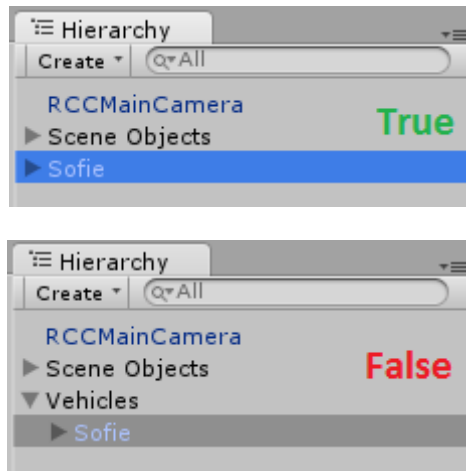
(You can zoom in with CTRL + ScrollUp for enlarge PDF pages)

Contents

Package Includes	1
First to Do!	3
About New Wheel Colliders	4
Creating New Vehicles	5
Warning	5
Important.....	5
Collider Shapes Of The Vehicle.....	10
Driving Assistances	12
Mobile Controllers	12
Mobile Controller (With Unity UI)	13
Mobile Controller (With NGUI)	14
About Mobile Usement on City Scene	15
Dashboard Configuration	16
Damage (Based on Mesh Deformation)	17
Creating Lights, Sounds, Skidmarks, Smoke Effects	18
Variable Ground Tire Grip.....	18
Adjusting Stiffness On Different Grounds	19
AI Configuration	20
Creating NavMesh For Scene	20
Adding AI Controller To Vehicle	22
Adding Waypoints Container To Scene.....	23
Adding BrakeZones Container To Scene	24
Enter-Exit System	24
Credits	25
Extreme Vehicle Pack by Vertigo Games.....	25
Sofie With Animations by BÜMSTRÜM	25
Sound Effects.....	25
License.....	26

First to Do!

Just don't parent your cars to another gameobject. Your car must be at root of your hierarchy;



Each vehicle has own [RCCCarControllerV2](#) script. Each vehicle is responsible for own [RCCCarControllerV2](#).

10 Main Categories for Easily and Understandable Creating–Configuring Vehicles.

[Wheels](#), [Steering](#), [Suspensions](#), [Mechanic Configuration](#), [Lights](#), [Sounds](#), [Mobile Controllers](#), [Dashboards](#), [Smokes](#), and [Damage](#).

About New Wheel Colliders

As you well know, Unity 5 uses Physx 3.3. They have improved wheelcolliders too. I'll explain new features about this;

Wheel Colliders can actually spin if you apply high torque to them. You can set your force point on Wheel Colliders. Realistic handling. Almost perfect everything. But... There are some bugs about current Wheel Colliders (5.1.1f1).

Wheel Colliders are based on your vehicle rigidbody. E.g. your Wheel Collider's spring force is 35000, and your vehicle mass is 1500. So when you lower your vehicle mass, suspension will loose force and vehicle will hit the ground unrealistically. This is known bug about new suspension calculation.

Enabling/disabling "isKinematic" bool in Rigidbody, requires re-enable all Wheel Colliders. Otherwise Wheel Colliders won't work after enabling/disabling "isKinematic" bool.

Sometimes Wheel Colliders are lagging and shaking when your Wheel Collider's settings are unproportional, or your Wheel Collider's mass is too heavy/light.

Root of the vehicle scale must be 1, 1, 1. Otherwise Wheel Colliders positions will be buggy-changed. Just try it, and you will see your Wheel Colliders positions are changing offsets while scaling root of the vehicle.

I'll update the package with each Unity update.

Creating New Vehicles

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

Warning

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

Important

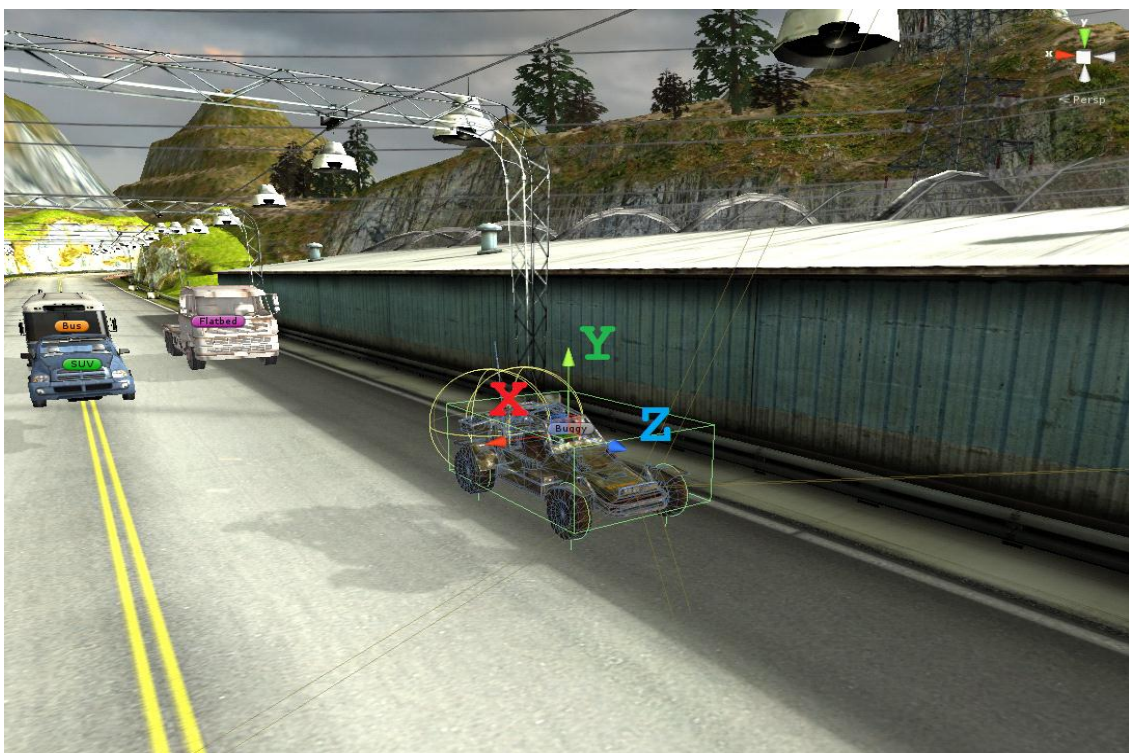
Be sure you are in **PIVOT** and **LOCAL** mode while checking directions.



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. (I don't prefer this way. If you want to do clean work, you should do this on your Designing Software).

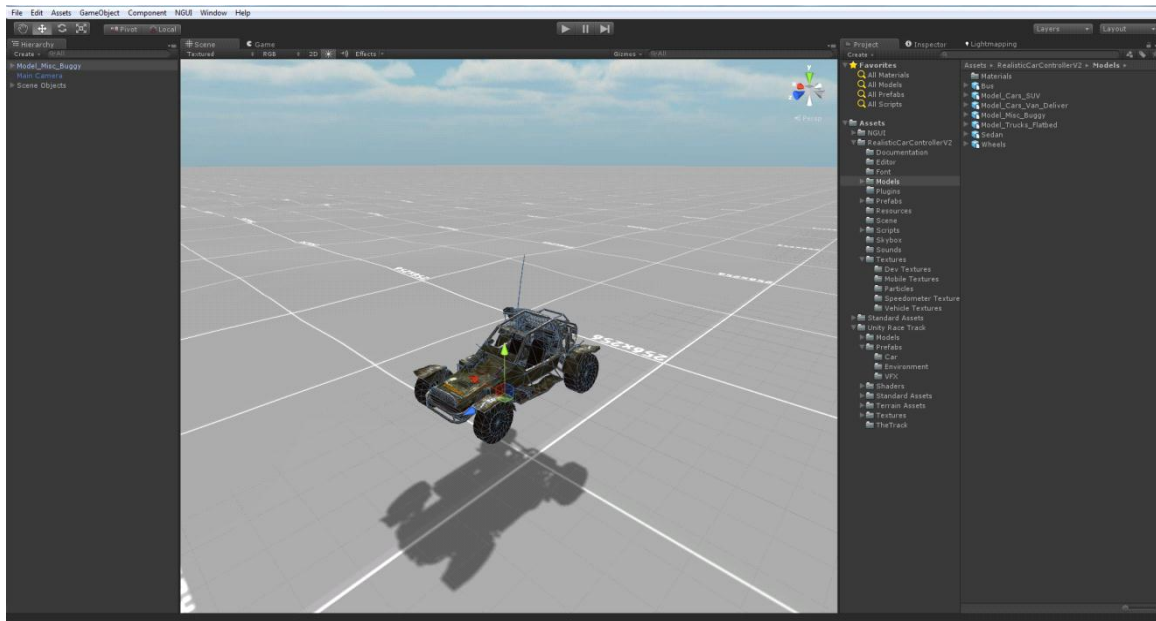
2 – Fix model's directions and pivot positions inside your Designing Software. Check my Youtube Channel for how to do this in 3ds Max.



So, you have checked inputs, your vehicle and wheel models pivot positions, and their X, Y, Z directions. Everything is OK right? Then...

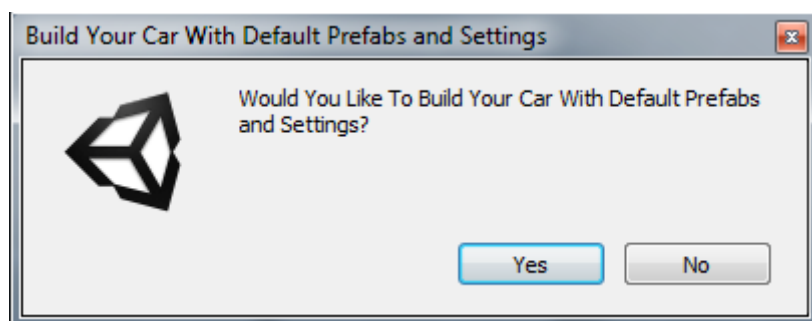
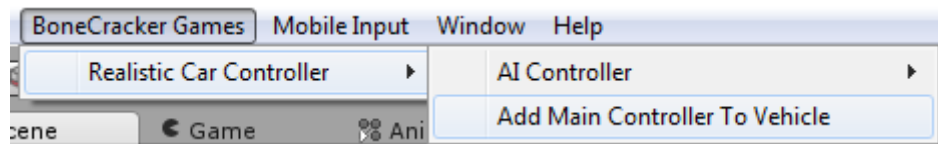
Drag and drop your Vehicle Model to your existing Scene and let's get started;

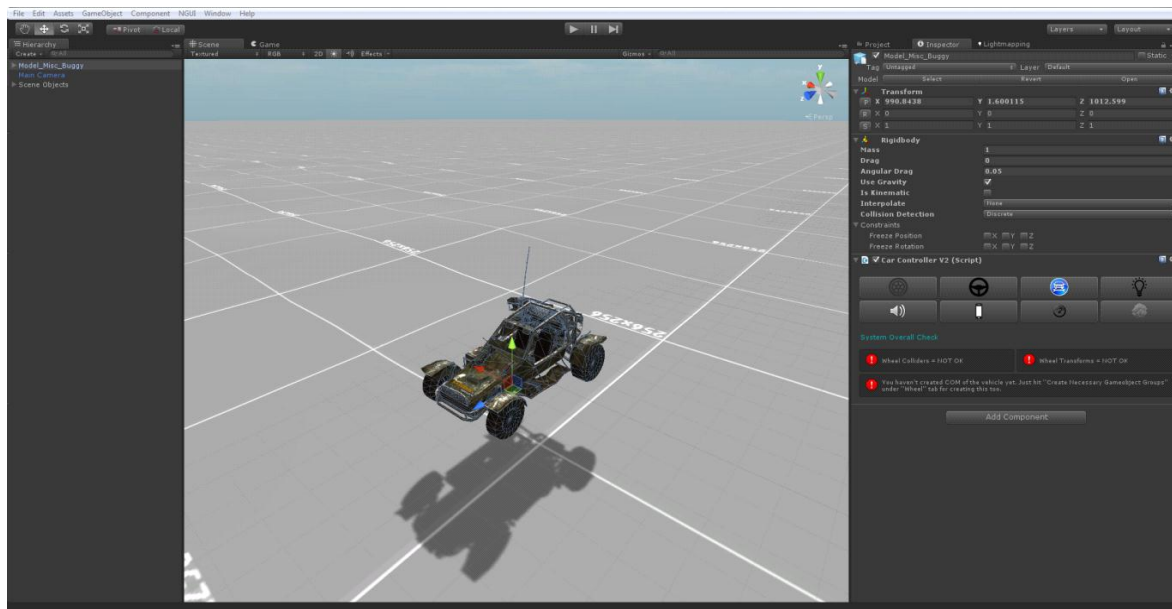
(Some images were taken in Unity 4, but they are nearly same with in Unity 5)



I'm using "[Extreme Vehicle Models](#)" for this. This is really great and mobile optimized asset for low price. I highly recommend you to use this amazing asset for your Project.

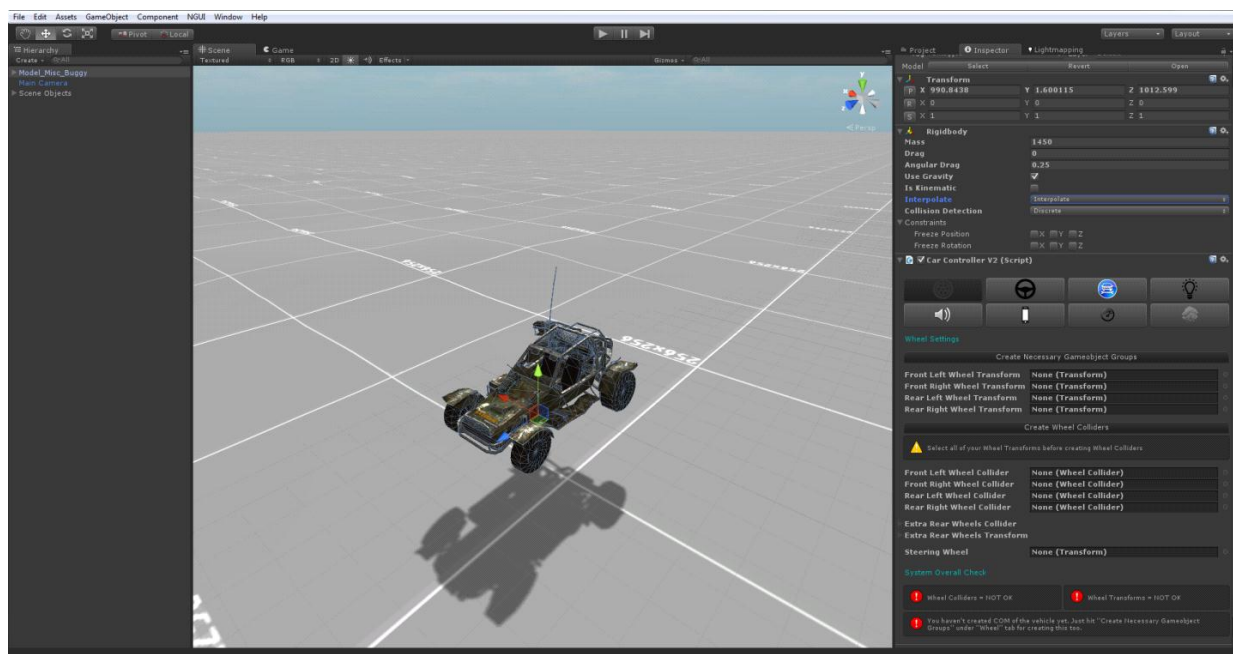
Now, you have to add [RCCCarControllerV2](#) to root of the vehicle. Just click Realistic Car Controller Menu and Add Main Controller To Your Vehicle;





As soon as when you added **RCCCarControllerV2** 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 0.05 – 0.15 for medium angular velocity.

After rigidbody settings, click “Wheel” tab in the editor script;



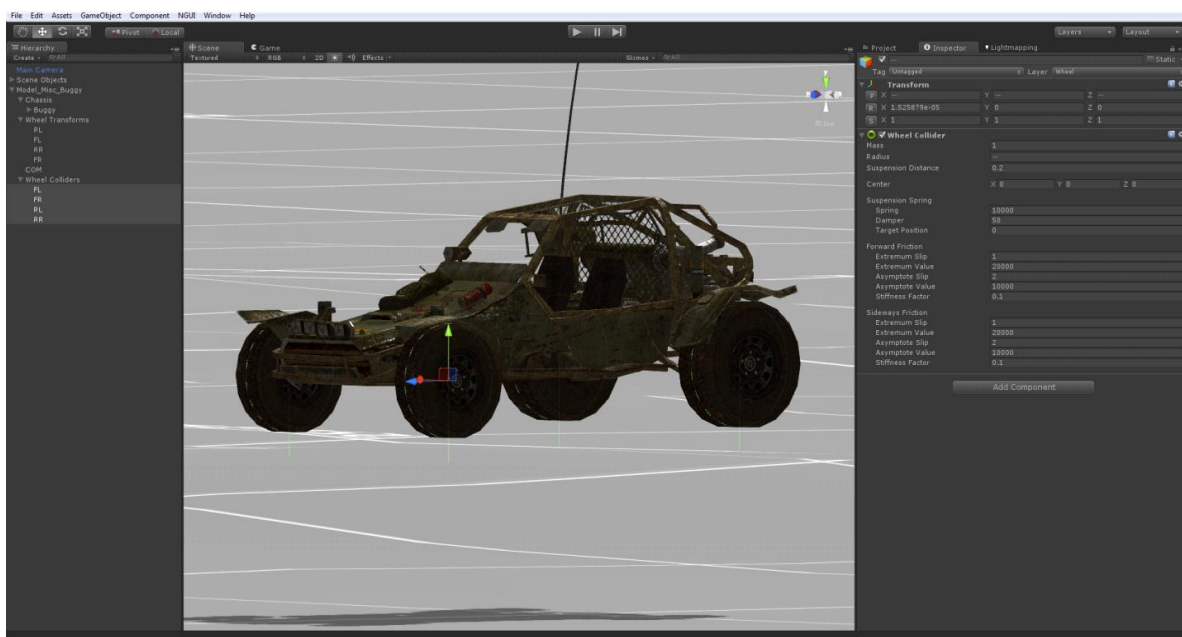
Click “Create Necessary Gameobject Groups”. This will create **Chassis**, **Wheel Transforms**, and **COM** of the vehicle (You don’t need to do this if you add controller from tab menu).



After creating these necessary gameobjects, simply drag and drop your model's parts to corresponding property. Your wheel gameobjects to "Wheel Transforms", your chassis to "Chassis". Then select your wheel transforms inside editor script.

After selecting your wheel transforms, hit the "**Create Wheel Colliders**" for creating Wheel Colliders with proper radius, suspension, damper, and friction curves AUTOMATICALLY*.

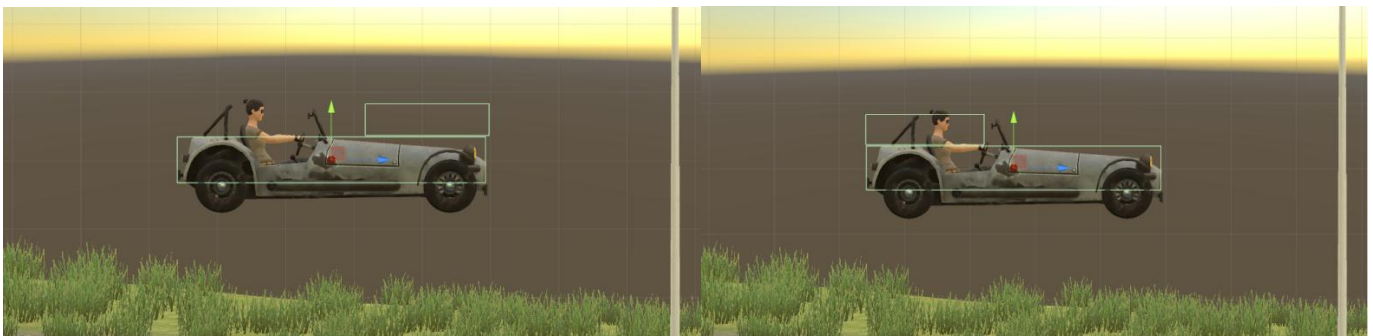
*(If your vehicle model direction is wrong, script will create Wheel Colliders at wrong angles)



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

Collider Shapes Of The Vehicle

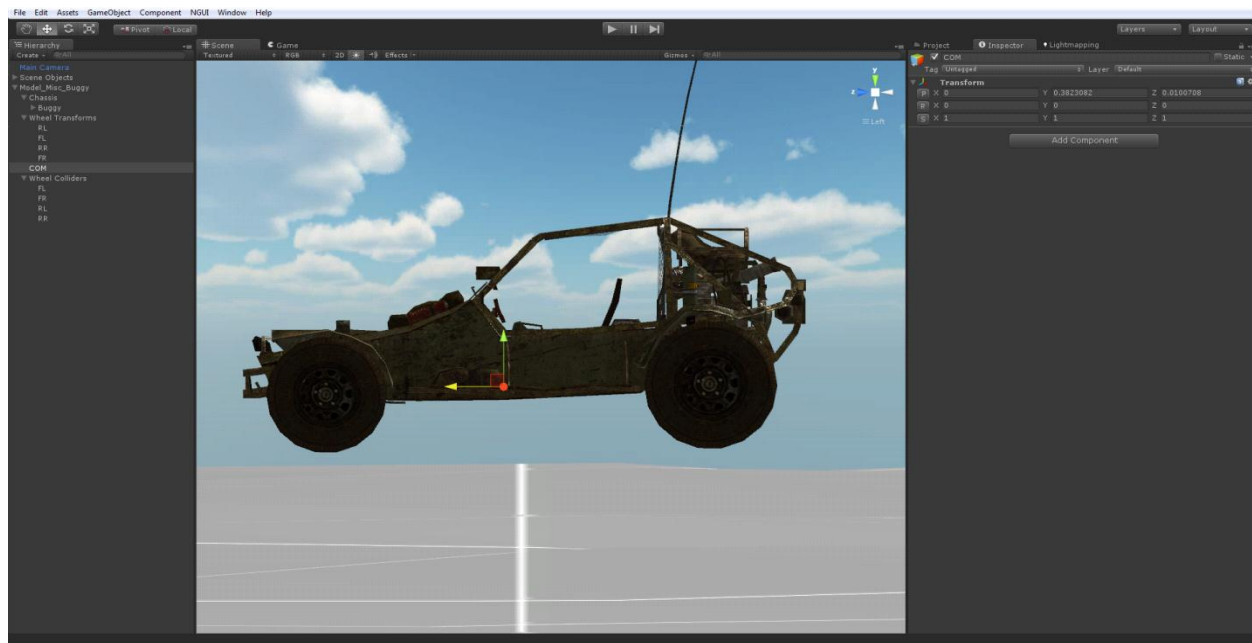
This is one of the most important thing for physics behavior. Most of devs are using mesh colliders for their vehicles. Remember that, shape of your vehicle collider will affect physics behavior directly. Let me explain this;



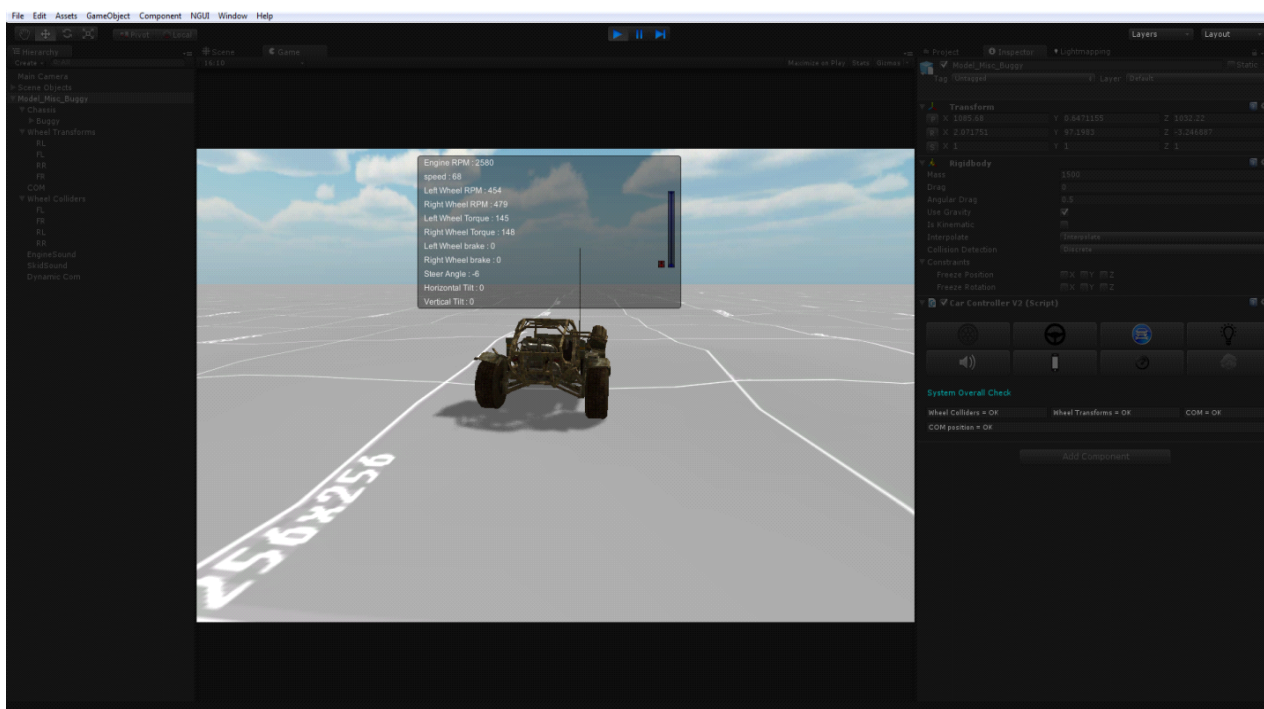
This vehicle has 2 Box Colliders. Second Box Collider is placed at front of the vehicle here on first screenshot. This will bring you more controllable and stable vehicle. Usually, second Box Collider should be at center Z of the vehicle. But this is high speed vehicle, so it must be stable. That's why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

Second Box Collider is placed at rear of the vehicle on second screenshot. This will bring you more unstable and slippery vehicle. Differences between first colliders and second colliders are huge. Just remember that collider shapes are effecting vehicle behavior directly.

After end up with Wheel Transforms 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. At this model, engine and other stuff is placed to back of the vehicle. But i'm not gonna place the COM to back of the car. Why? Because this vehicle is RWD, and can flip back easily when applying high torque to rear wheels. So, i'll just set it to just like this;



Runs perfectly after just few clicks.



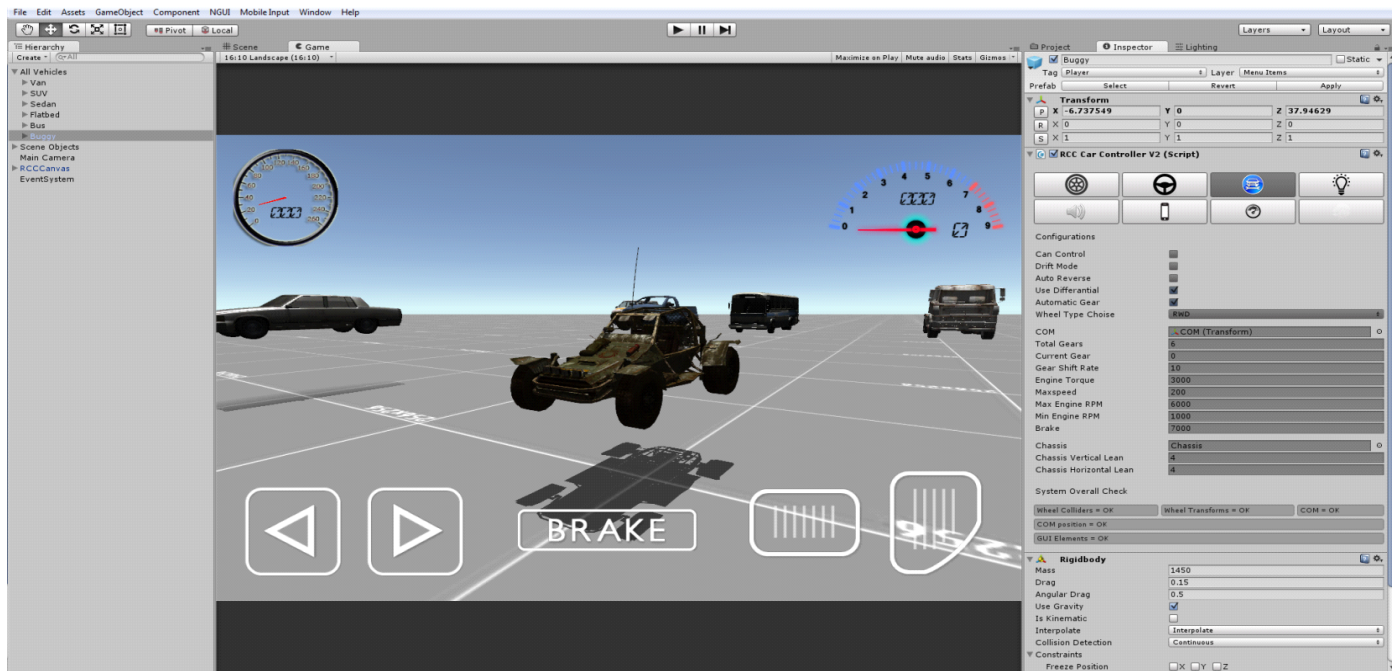
Default Main Settings are;

Engine Torque = 2500;

Brake Torque = 4000;

Maximum Speed = 220;

Wheel Type = RWD (I used RWD for this vehicle);



Configure your vehicle as you wish. If you want to use manual gear, you need to create 2 Inputs. “**RCCShiftUp**” and “**RCCShiftDown**”. And set which key you want to. You can create new Inputs by Edit → Project Settings → Input.

Driving Assistances

Script has **ABS**, **TCS**, **ESP** assistances. Threshold means, if wheel slip is equals or higher then this threshold value, corresponding assistance will be engaged immediately. Steering helper 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.

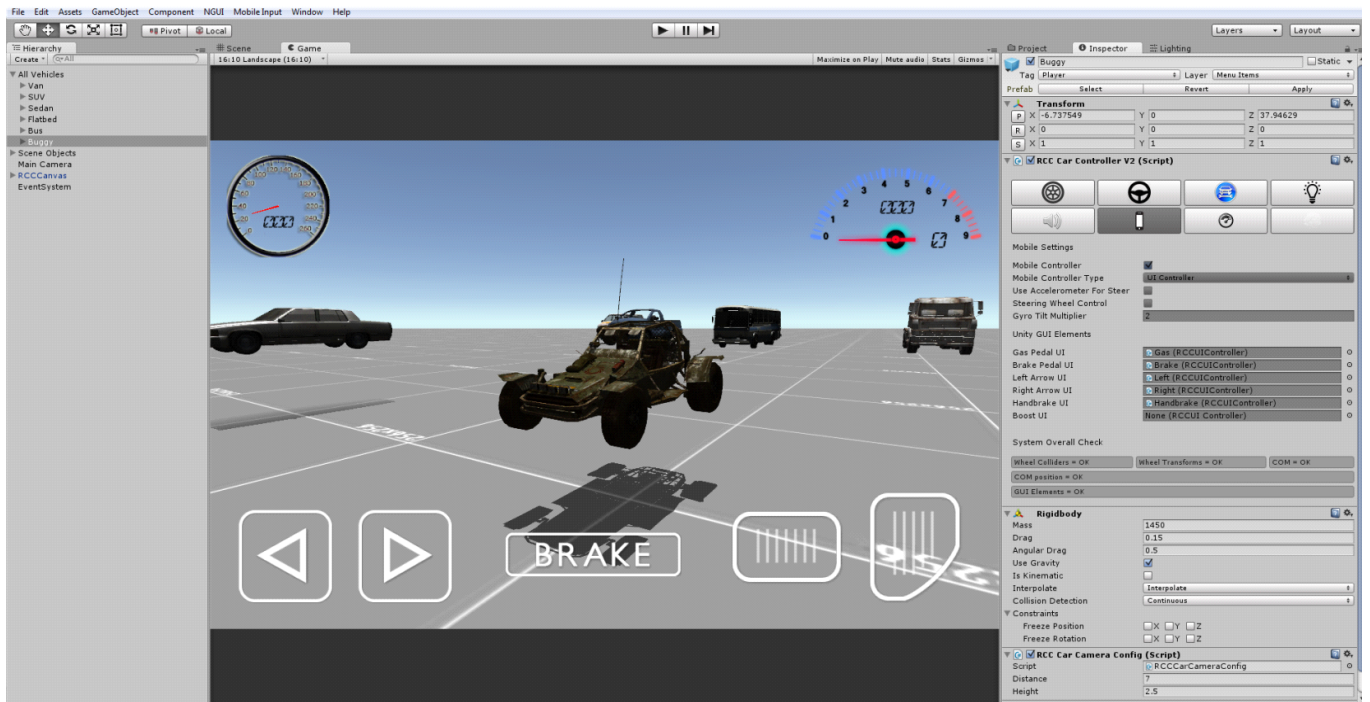
Mobile Controllers

Now let’s make our vehicle can be controller by Mobile Controller. Just click “**Mobile**” tab in the editor script and enable “**Mobile Controller**”.

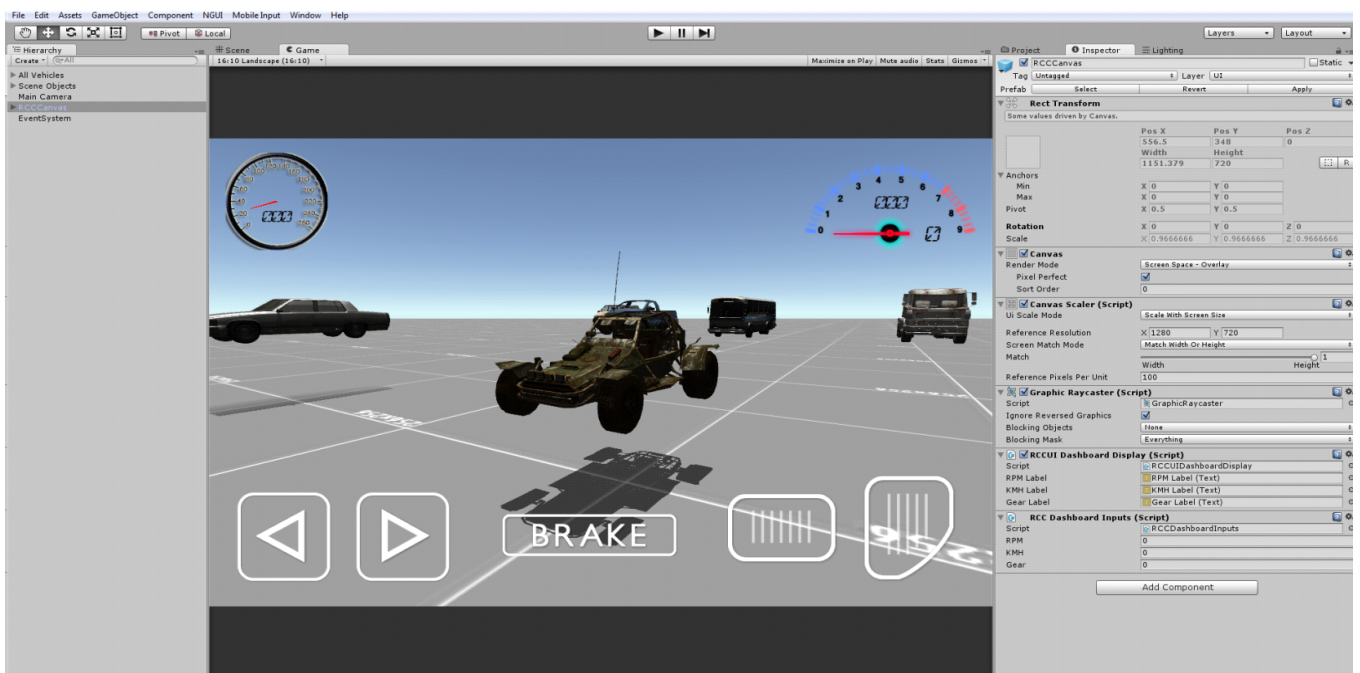
Package supports two type of controller and dashboard display;

- **Unity UI (+4.6),**
- **NGUI.**

Mobile Controller (With Unity UI)



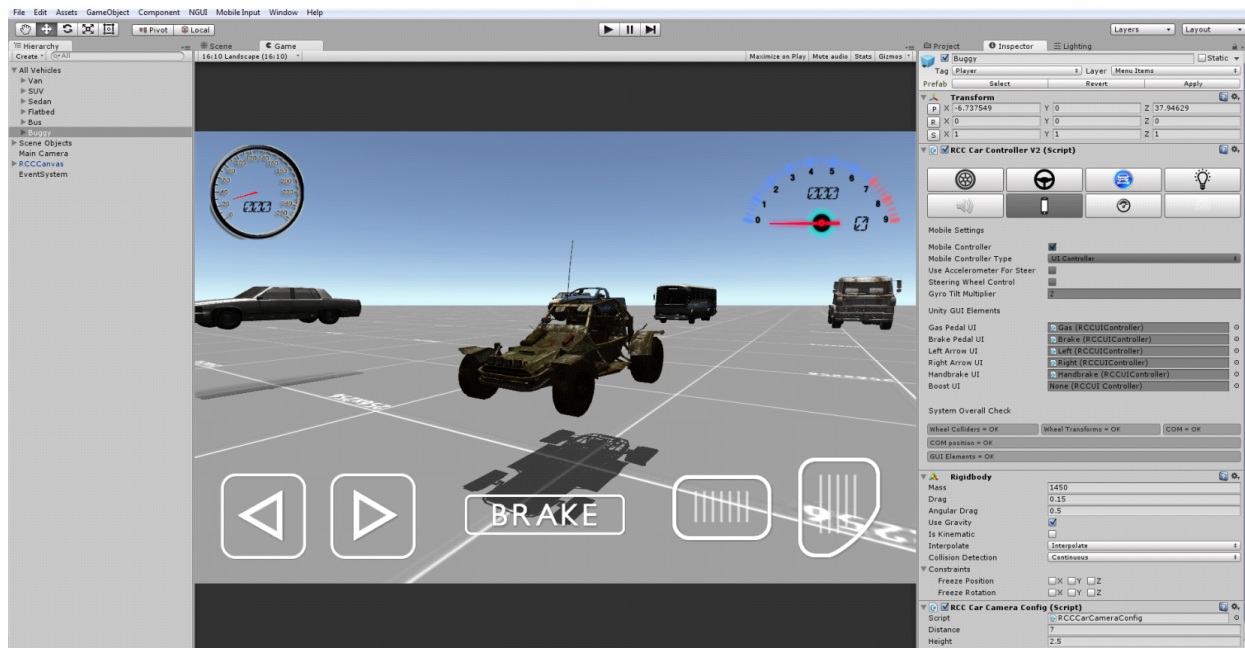
Default Mobile Controller type is “**UIController**”. You will find “**RCCCanvas**” prefab under RealisticCarControllerV2/Prefabs. Drag and drop to your Hierarchy.



Each UI controller gameobject has “**RCCUIController**” script for inputs.

~~(Obsolete) Select all of your UI controller elements inside the editor script, and give it a try.~~

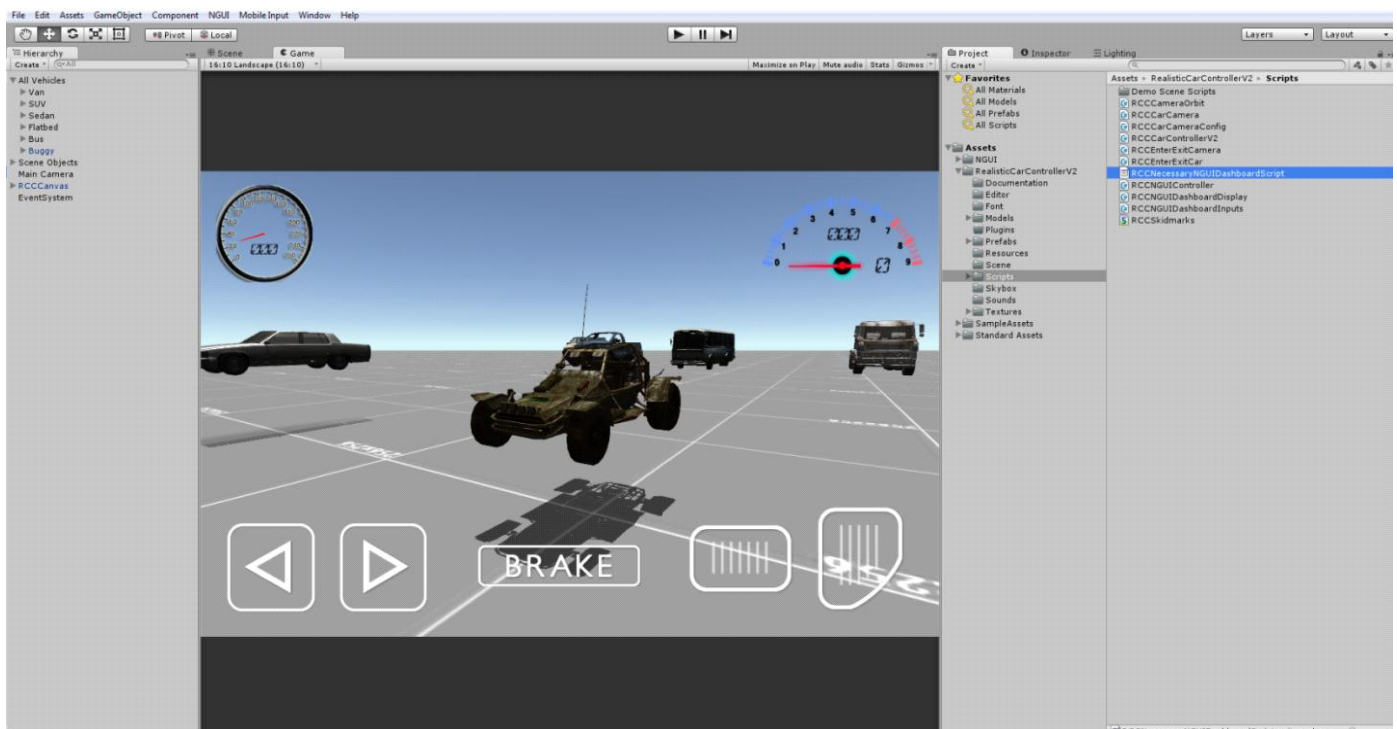
You can enable “Auto Find Mobile Buttons” bool, and let script find them automatically.



Runs perfectly with Unity Remote (Emulating to my android device). I forgot to add boost button, anyway, boost button is optional. Use it or not. Boost button will multiply your engine torque by 2.5x.

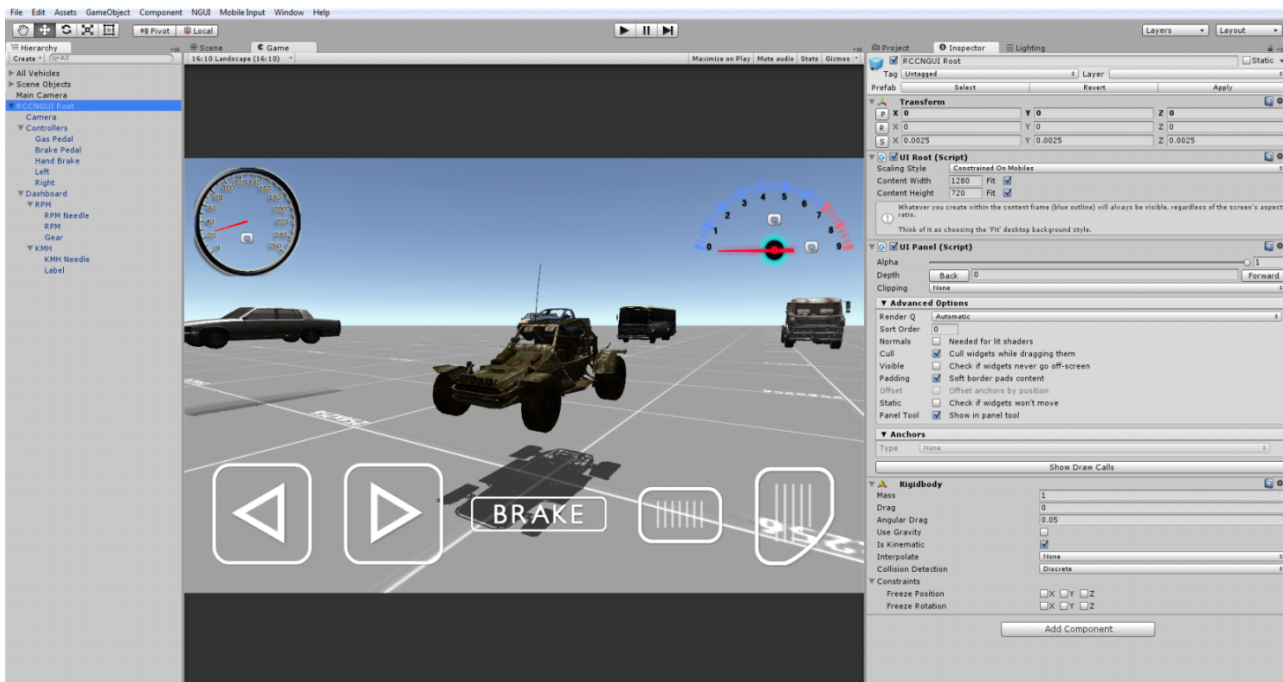
Mobile Controller (With NGUI)

First, import latest NGUI Package to your Project. Then, import “[Necessary NGUI_Dashboard Scripts](#)”(under RealisticCarControllerV2 /Scripts) to your Project;



You will find “**RCCNGUIRoot**” under RealisticCarControllerV2 /Prefabs. Drag and drop the **RCCNGUIRoot** 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.



You can enable “**Auto Find Mobile Buttons**” bool, and let script find them automatically.

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 our 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 would get performance loss.

Dashboard Configuration

Just click the “Dashboard” tab, and enable “Dashboard”;

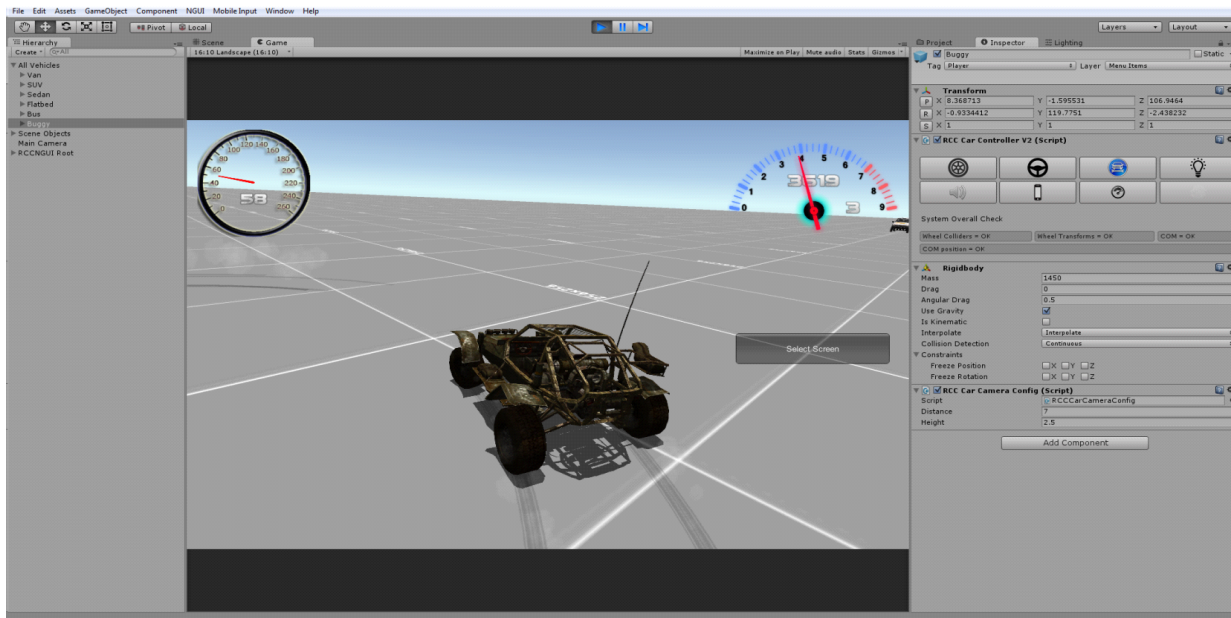


Default Dashboard Type is “**UI Dashboard**”. Script will access “**RCCDashboardInputs**” and “**RCCDashboardDisplay**” scripts for displaying dashboard. These scripts are attached on;

Unity UI – RCCCanvas root.

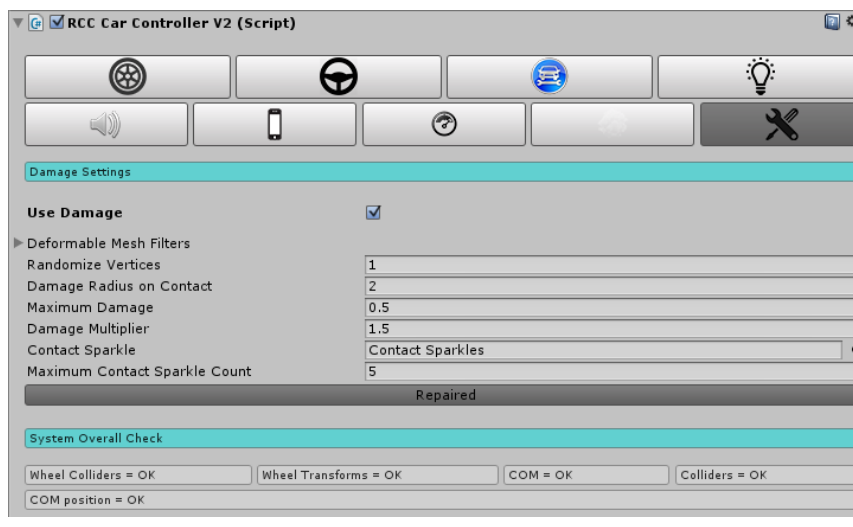
NGUI – RCCNGUI Root.

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.



Runs perfectly.

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. Click “Light” in editor script, and select all of your corresponding light.

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 “[WheelSlipAsphalt](#)” and “[WheelSlipSand](#)” prefabs under Prefabs folder. You can use your own smoke prefab as you wish.

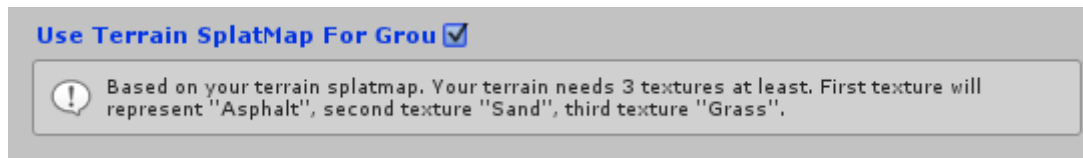
If you want to use exhaust effects, you will find “[RCCExhaust](#)” under Prefabs folder. You need to place it to your model correctly. Then select it inside editor script.

You will find “[RCCWheelSkidmarks](#)” under Scripts folder. Attach it to your all of your Wheel Colliders. Select your vehicle inside script. And scene must have “[RCCSkidmarksManager](#)”. You will find the “[RCCSkidmarksManager](#)” prefab under Prefabs folder. Drag and drop to your Hierarchy.

Variable Ground Tire Grip

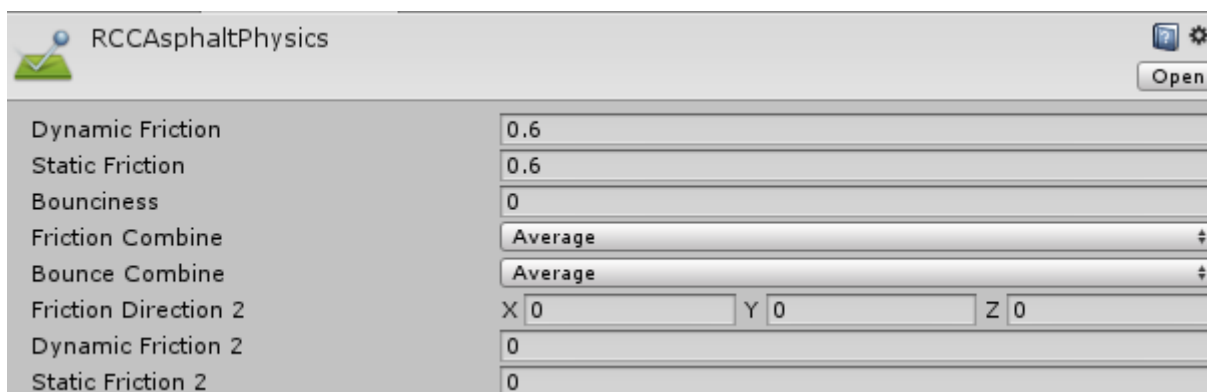
If you want to get variable tire grips on varied surfaces, use this option. Currently 3 surfaces available such as Asphalt(Default), Sand, and Grass. You will find “[RCCAsphaltPhysics](#)”, “[RCCGrassPhysics](#)” and “[RCCSandPhysics](#)” Physic Materials under “[Resources](#)” folder. If your scene 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. Your terrain needs 3 textures at least. Select each index of your terrain texture slot for corresponding physics material. And don't forget to enable "[Use Terrain SplatMap For Ground Physics](#)" bool for your vehicle.



[Adjusting Stiffness On Different Grounds](#)

You can adjust each materials stiffness by changing friction of the physics material. Just like this;



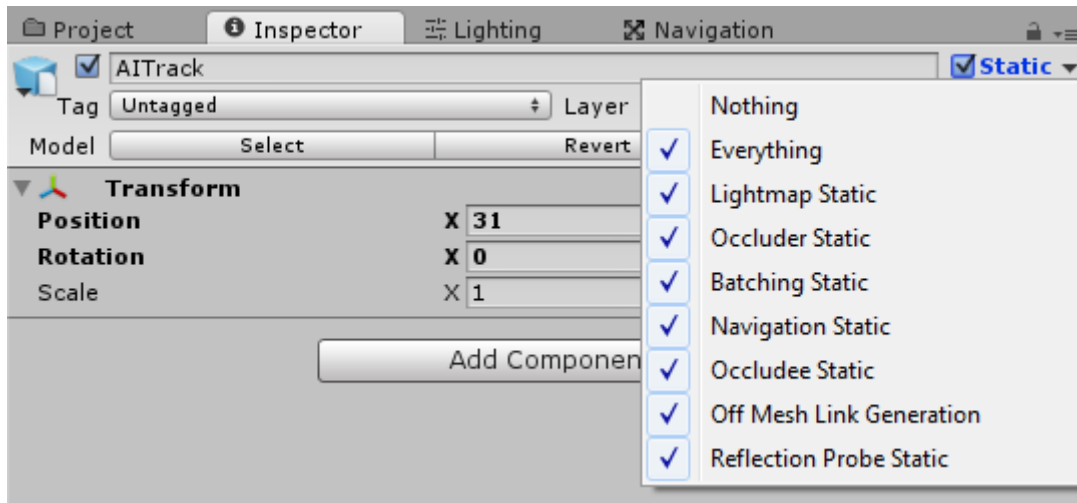
Script is accessing materials static friction value for stiffness. Default friction value is 0.6. Script is multiplying this value by 1.666 for getting default 1 value. If wheel hits on a physic material with default static friction value 0.6 will return by 1 stiffness value.

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

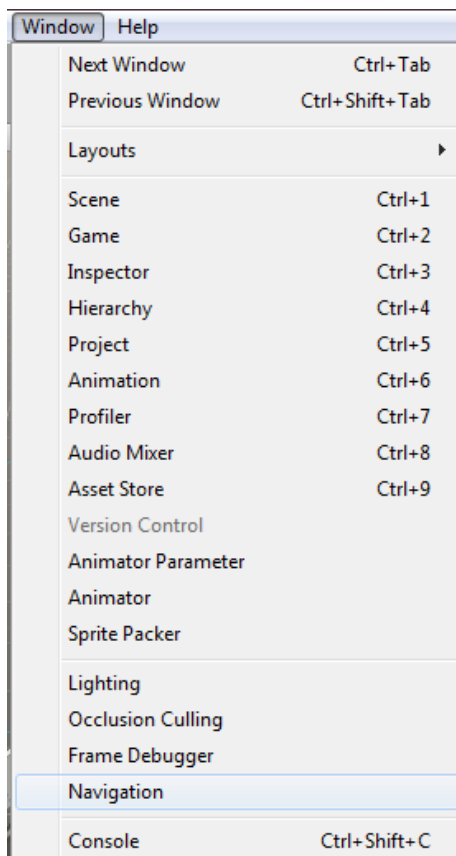
AI Configuration

Creating NavMesh For Scene

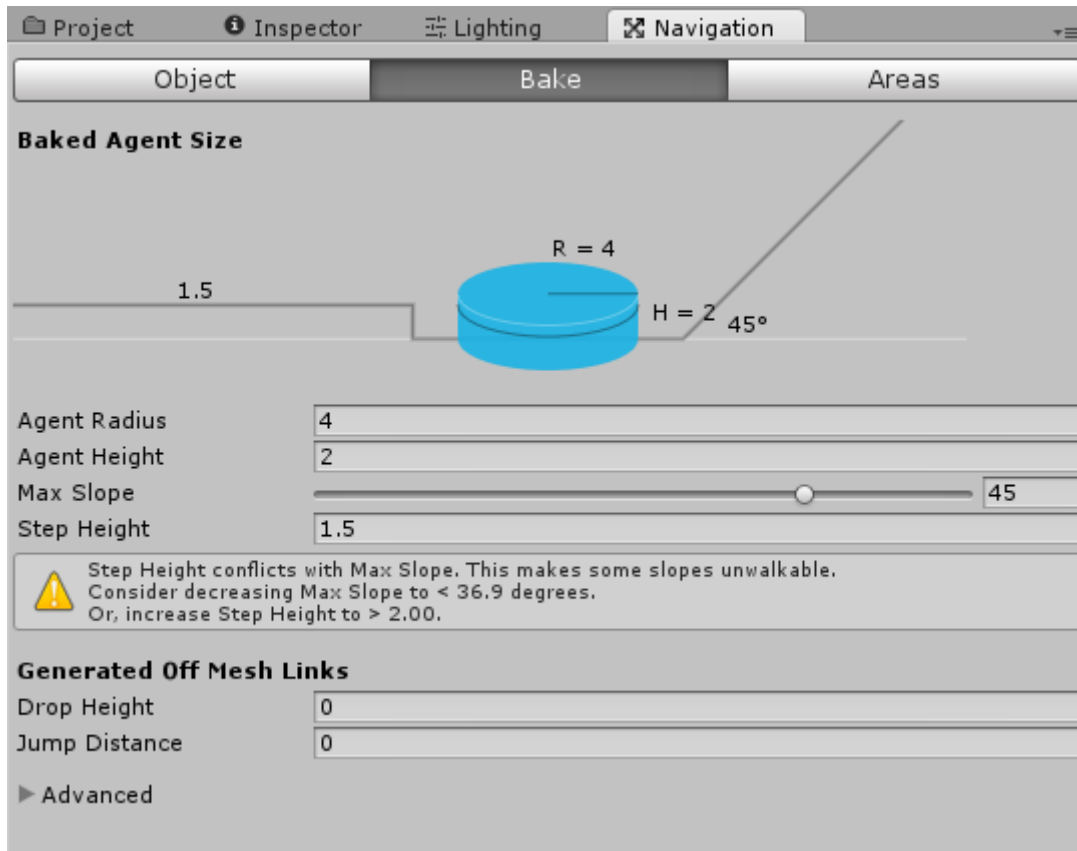
AI is based on Unity's Nav Mesh. Therefore, you must bake and create navigation mesh for your scene. Select your all static objects (including road too). And set them as "Static". Just like in AITrack Demo Scene;



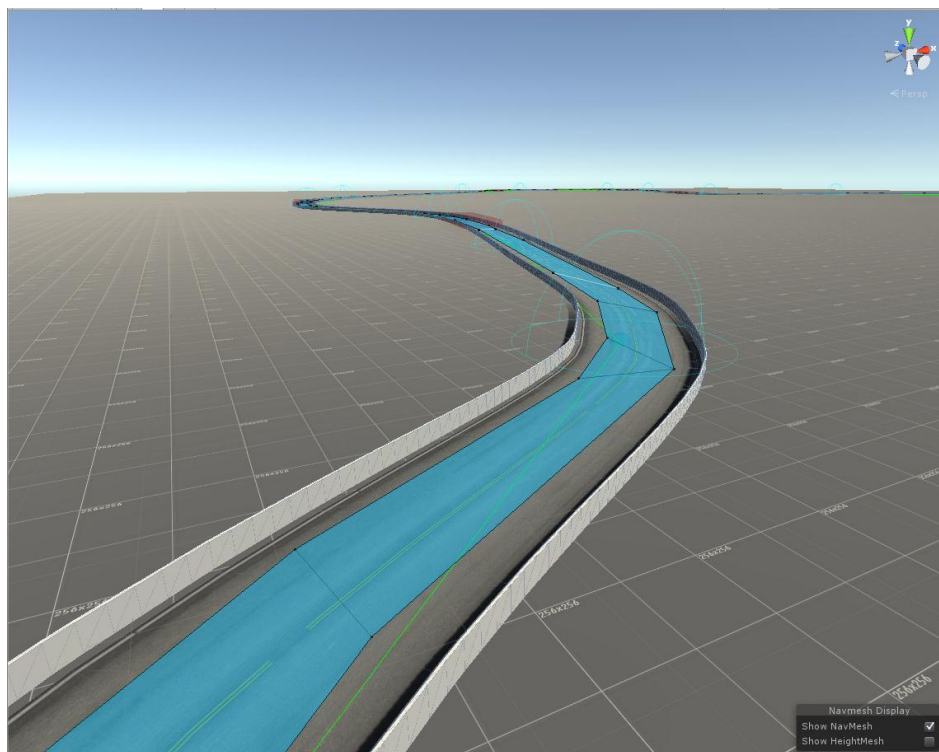
When all your static objects are marked as "Static", then you can bake your navigation mesh. Open "Navigation" window from Window → Navigation.



Default settings should be like this;

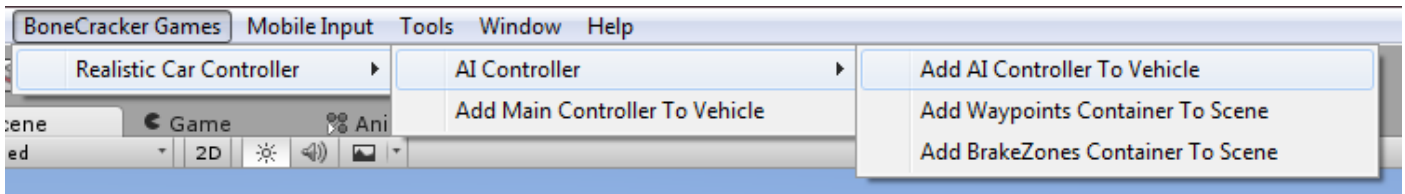


And then, click the bake button and bake your scene. Check your blue navigation mesh. Should be like this;

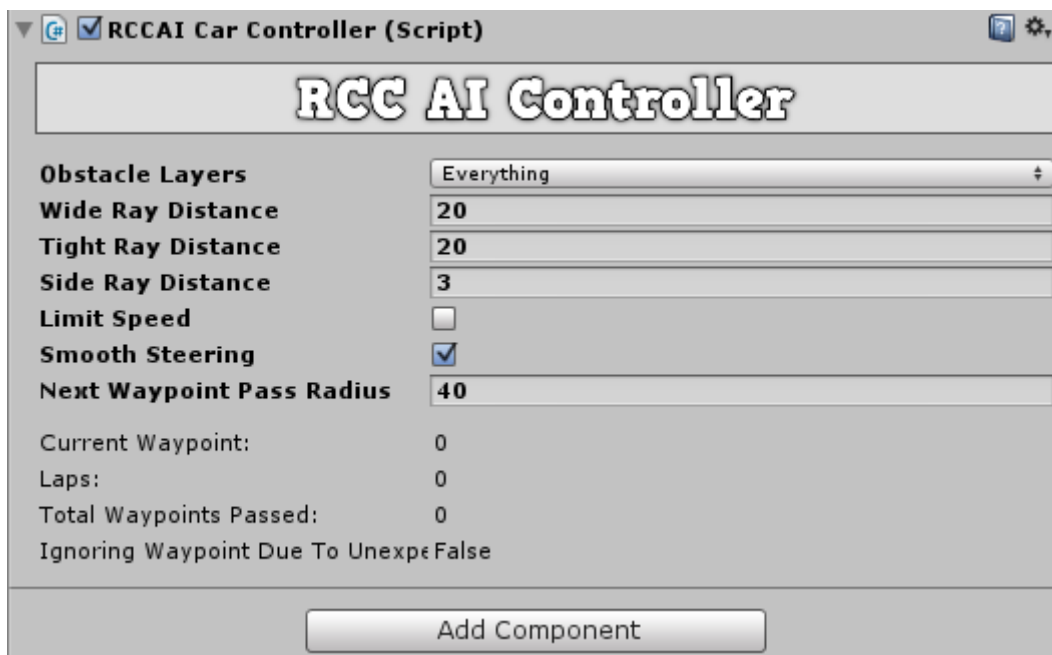


Adding AI Controller To Vehicle

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 “BoneCracker Games/AI Controller/Add AI Controller To Vehicle”

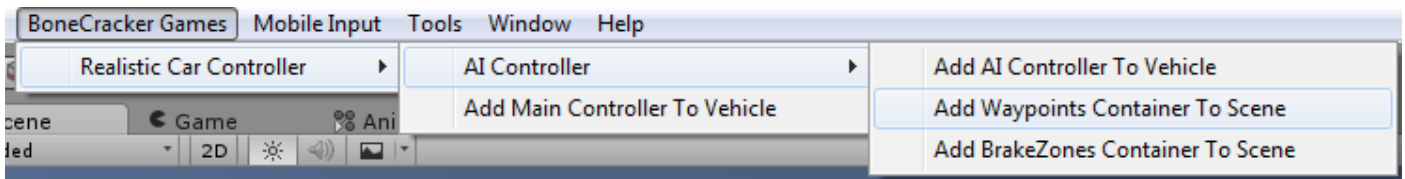


This will add “RCCAIController”to root of your vehicle.

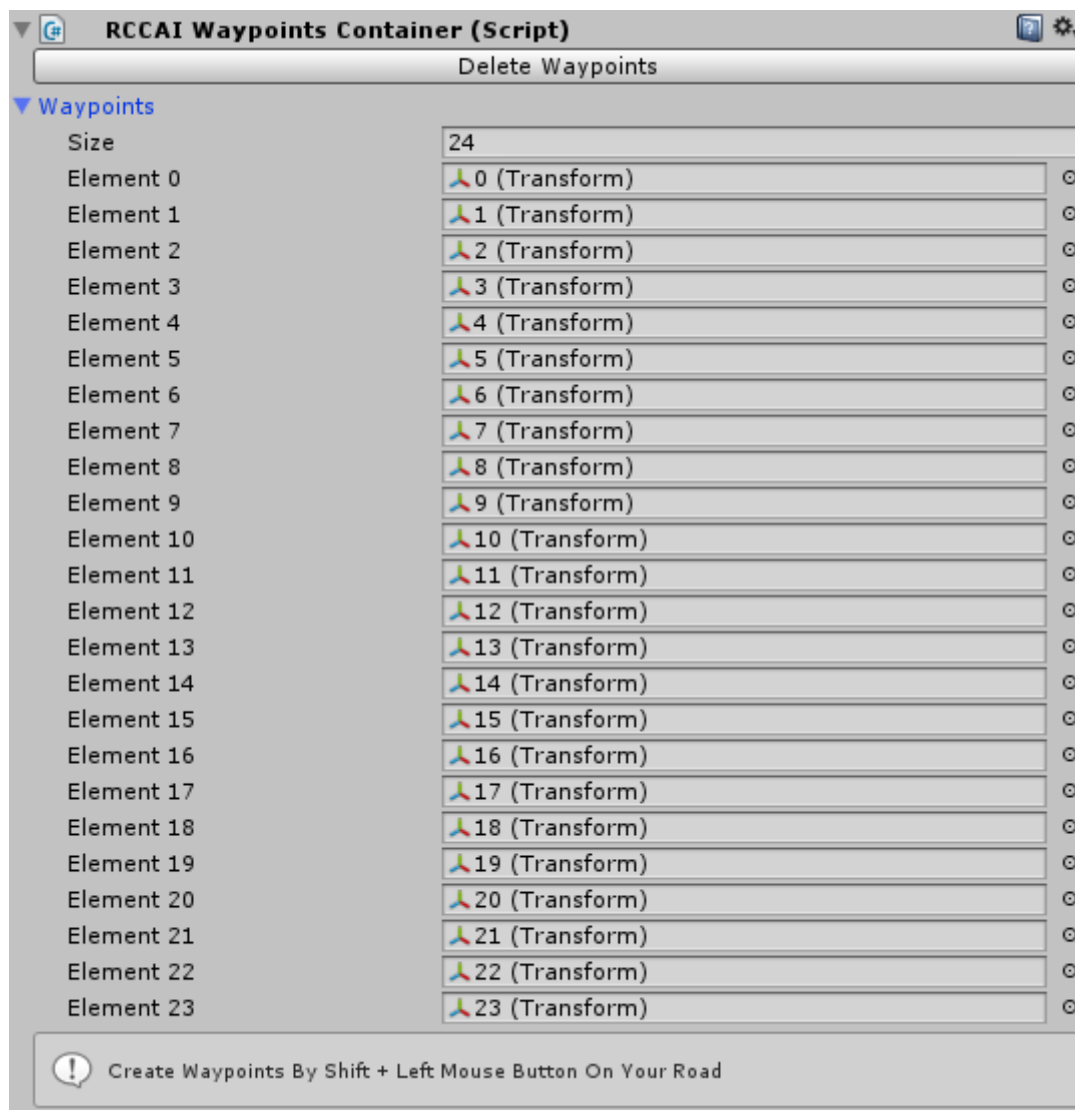


Vehicle will use “Nav Mesh Agent” for road path based on your waypoints, and will use raycasts for dynamic objects. If you have specified gameobjects for ignoring raycasts for this gameobject, you can select your gameobject layer from obstale layers.

Adding Waypoints Container To Scene

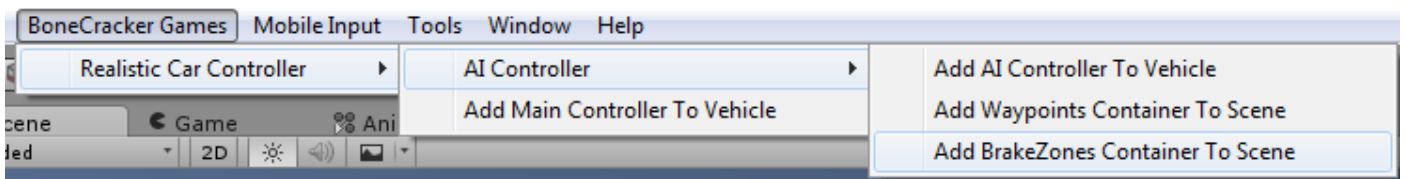


This will add “RCC AI Waypoints Container” to your scene. Simply hold Shift and click on your road to create a new waypoint. Create your path with them. Just like in the “AITrack Demo Scene”;

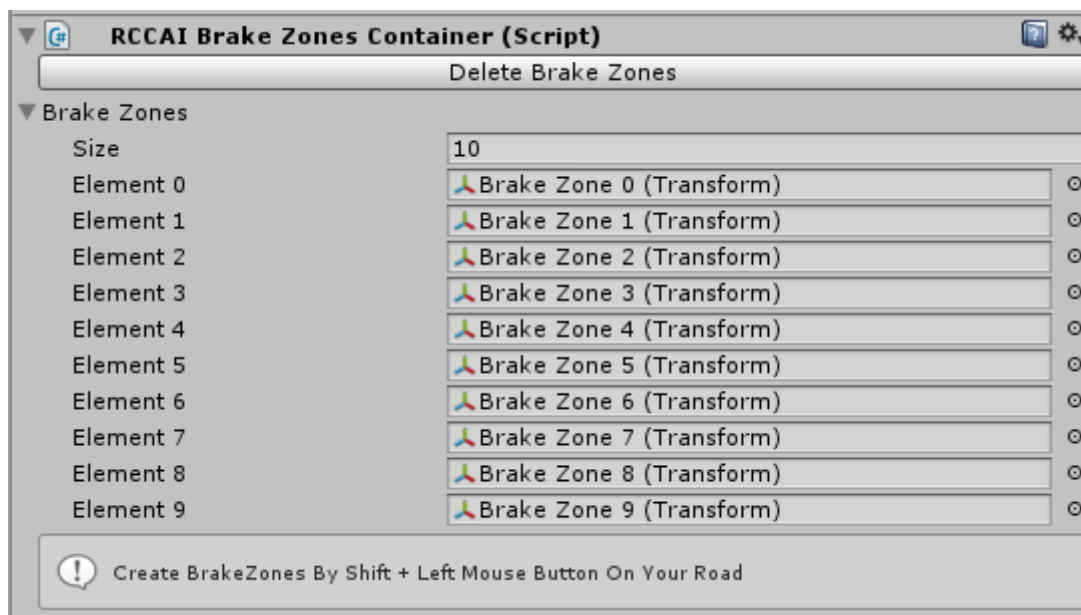


Info: Do not use CTRL+D for duplicate any waypoint.

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. Just like in the “AITrack Demo Scene”;

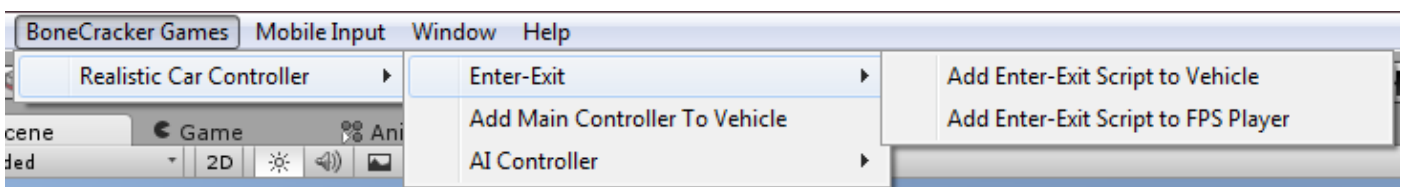


Info: Do not use CTRL+D for duplicate any brake zone.

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 RCCEnterExitCar.

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 BÜMSTRÜM

Driver Sofie, her animations, and her car model made by BÜMSTRÜM. You can access BÜMSTRÜM's 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 the package on any store. 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 Buğra Özdoğanlar

BoneCrackerGames@gmail.com