Keeree Car Constructor.

1. Scripts.

1. **Car constructor core.** The main script.

2. **Car part.** Contains information and parameters of every car part. This script should be assigned and configured to all car parts.

3. **ParametersKeeper.** Contains information and parameters of every Car. This script should me assigned to every car root gameobject.

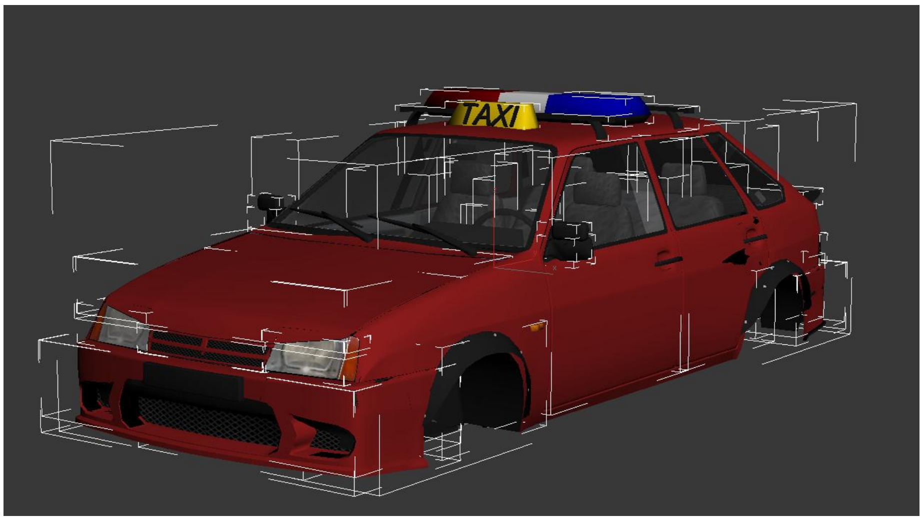
4. **CamControl.** Camera controller.

5. **AnimatedCarParts.** This script animates EVERYTHING moveable in car. Dynamic suspension, steering wheel rotation, engine parts spinning, wheels rotation are moved by this script.

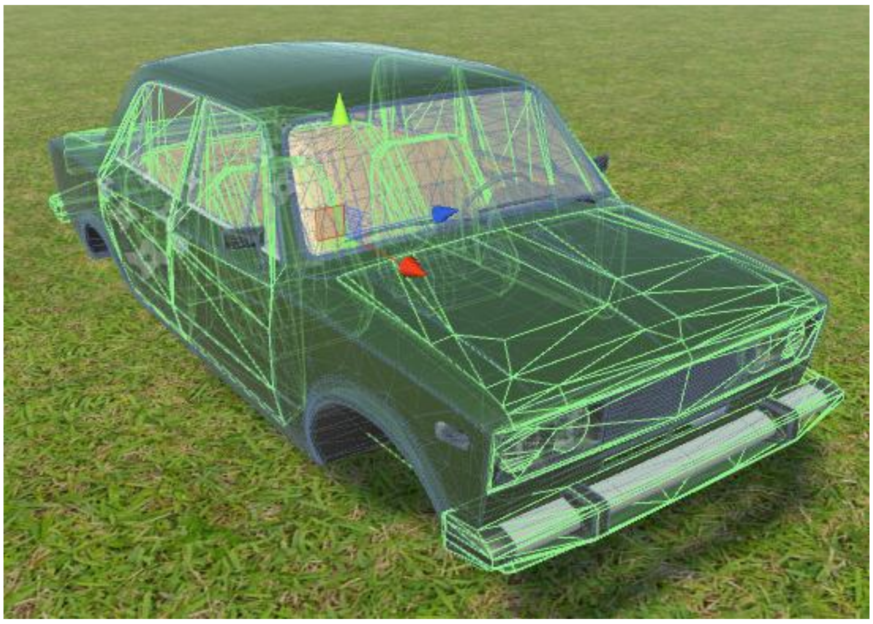
6. **CarController.** Simple car controller based on standard wheel colliders.

Tutorial – Preparing your car model

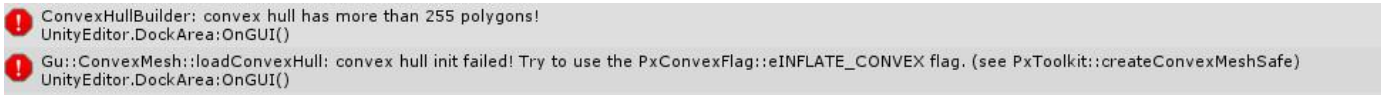
Every supposed to unmount car part should be a an individual object. It also has to have clean and correct mesh, as the unmount system uses raycasting and convex mesh colliders on each car part.



To make sure that the mesh is correct, export the model to .fbx file, import it into Unity scene and add a mesh collider component with checked "Convex" box to ALL car parts supposed to be unmountable (except car body). If the mesh colliders form correctly, the mesh is good.

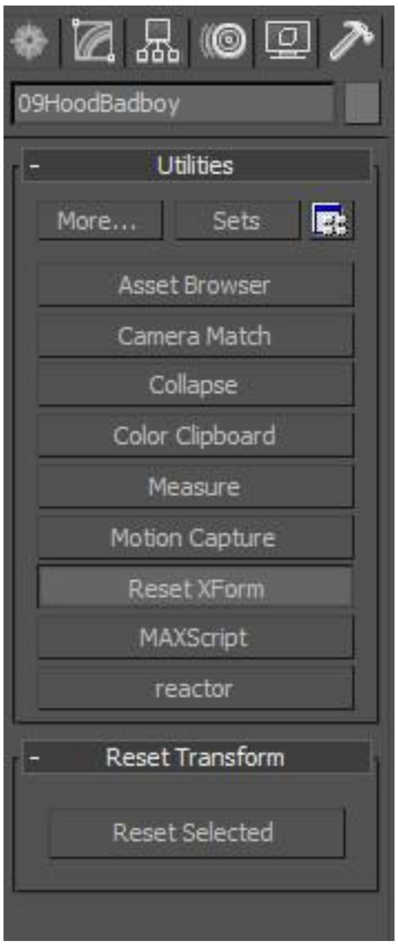


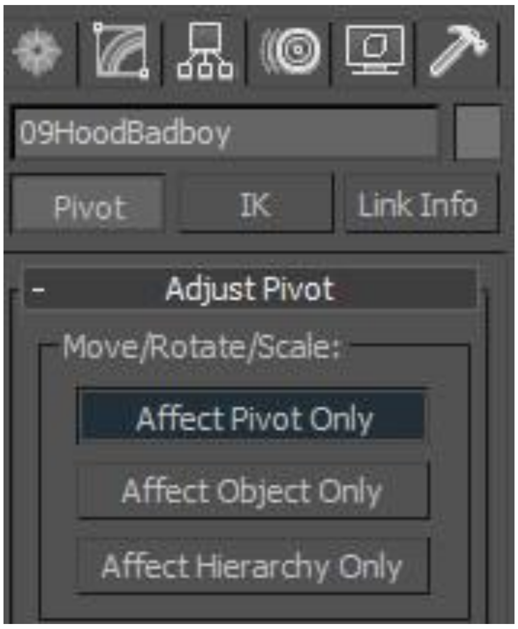
If there appears an error message, go back to Max and modify the meshes of incorrect parts.



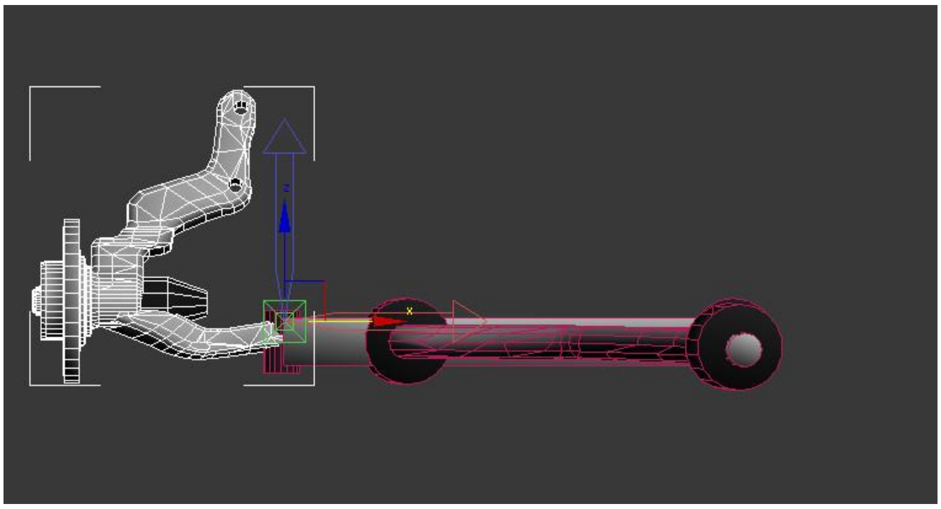
```
ConvexHullBuilder: convex hull has more than 255 polygons!
UnityEditor.DockArea:OnGUI()
Gu::ConvexMesh::loadConvexHull: convex hull init failed! Try to use the PxConvexFlag::eINFLATE_CONVEX flag. (see PxToolkit::createConvexMeshSafe)
UnityEditor.DockArea:OnGUI()
```

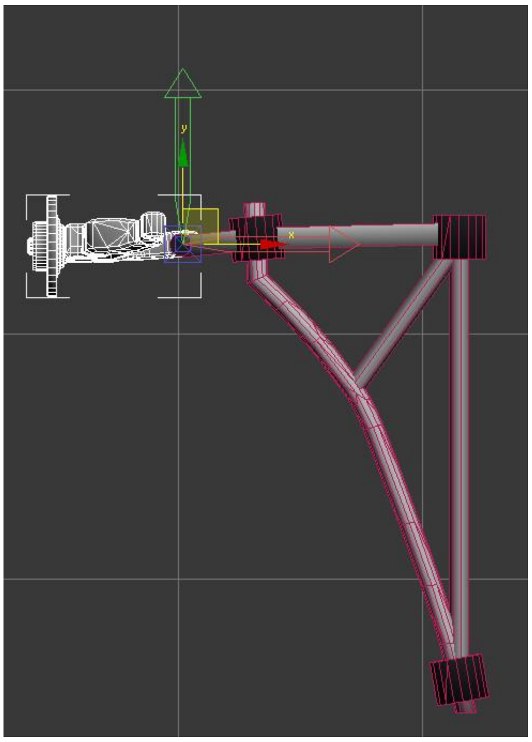
If you're working in 3dsmax, perform reset Xform for all object.



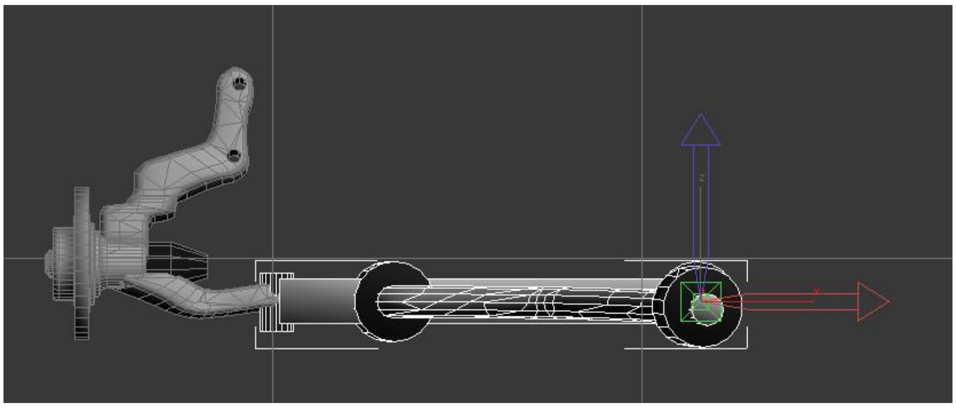
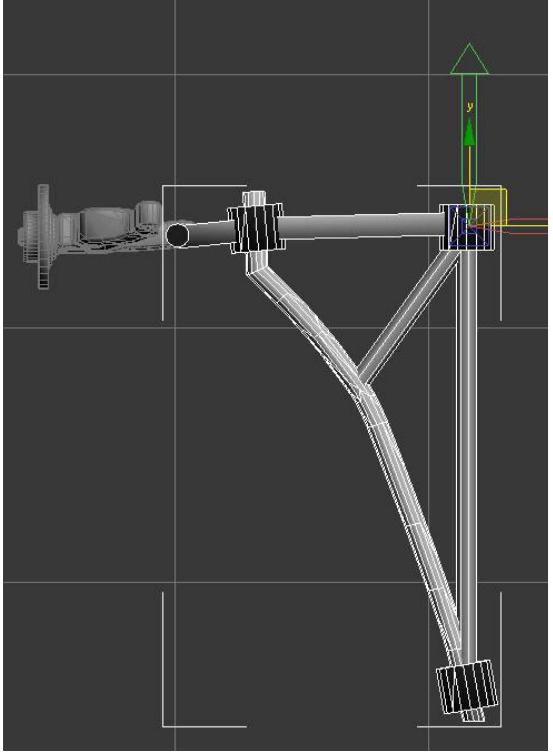


Pivot points. This is an extremely necessary step. Adjust pivot point of every car part. Notice, that doors, hood and trunk will rotate around the points you adjust. Also, make sure to put pivot points of suspension parts in right places. For HUB the best place for pivot is at the joint between hub and trailing arm. Like this:

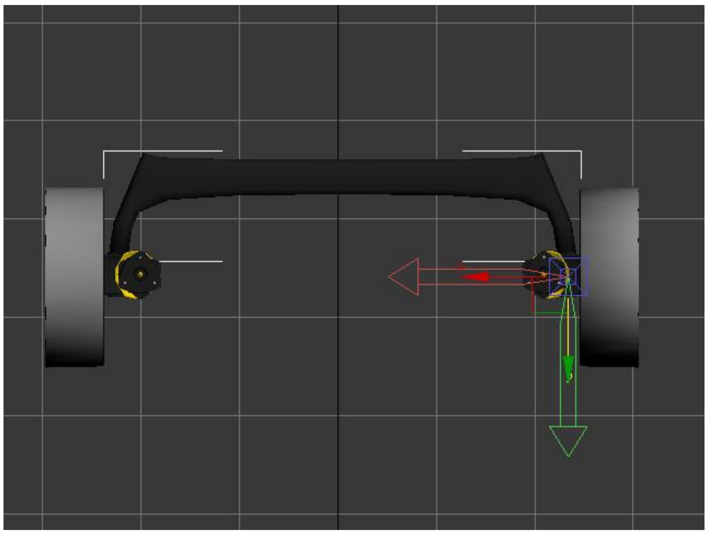




And for trailing arm:

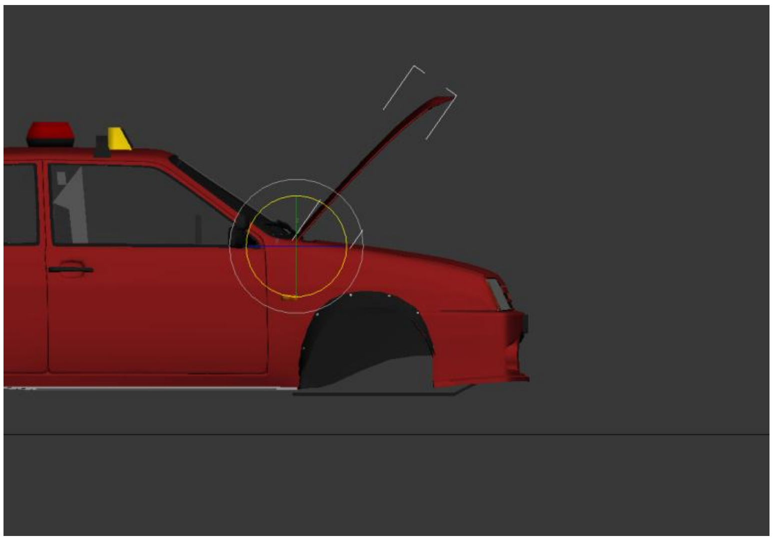



Pivot point of beam axle must be same to rear right hub.

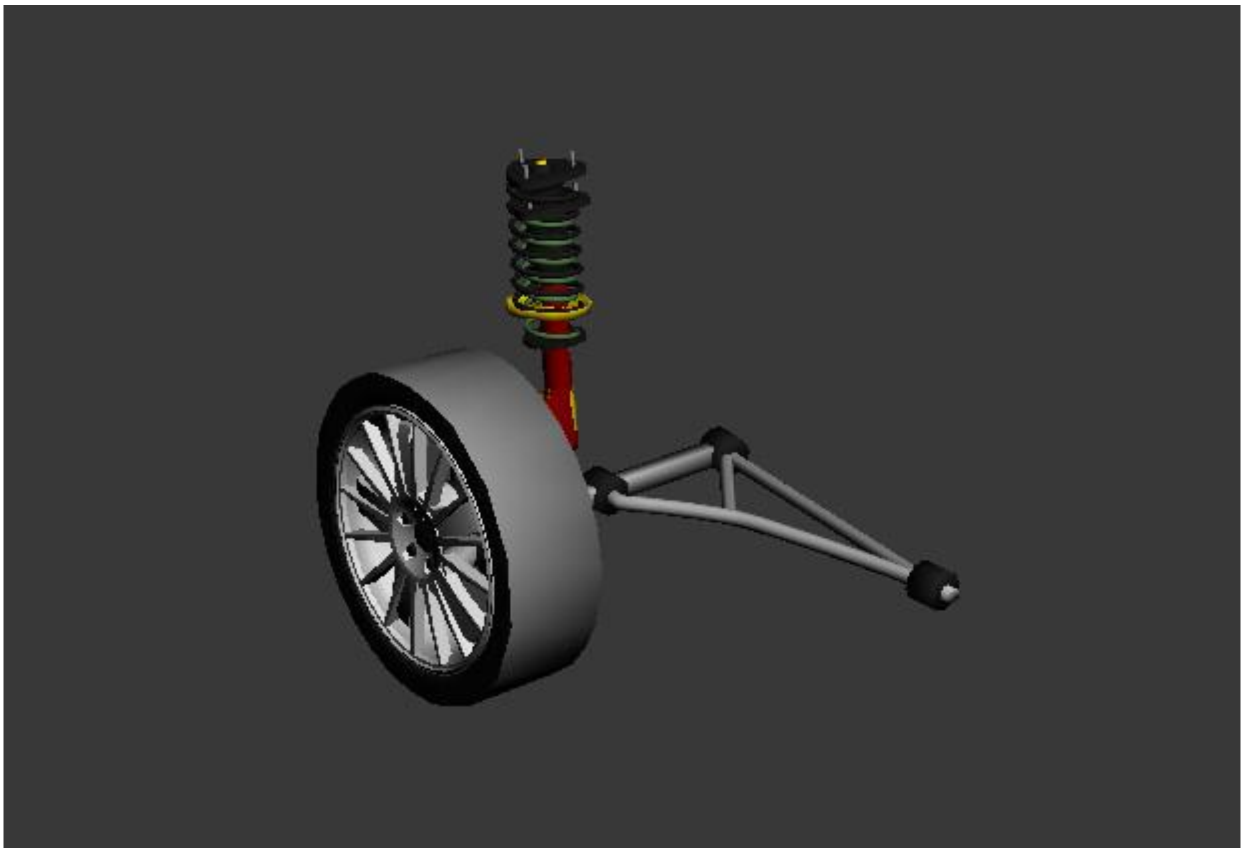


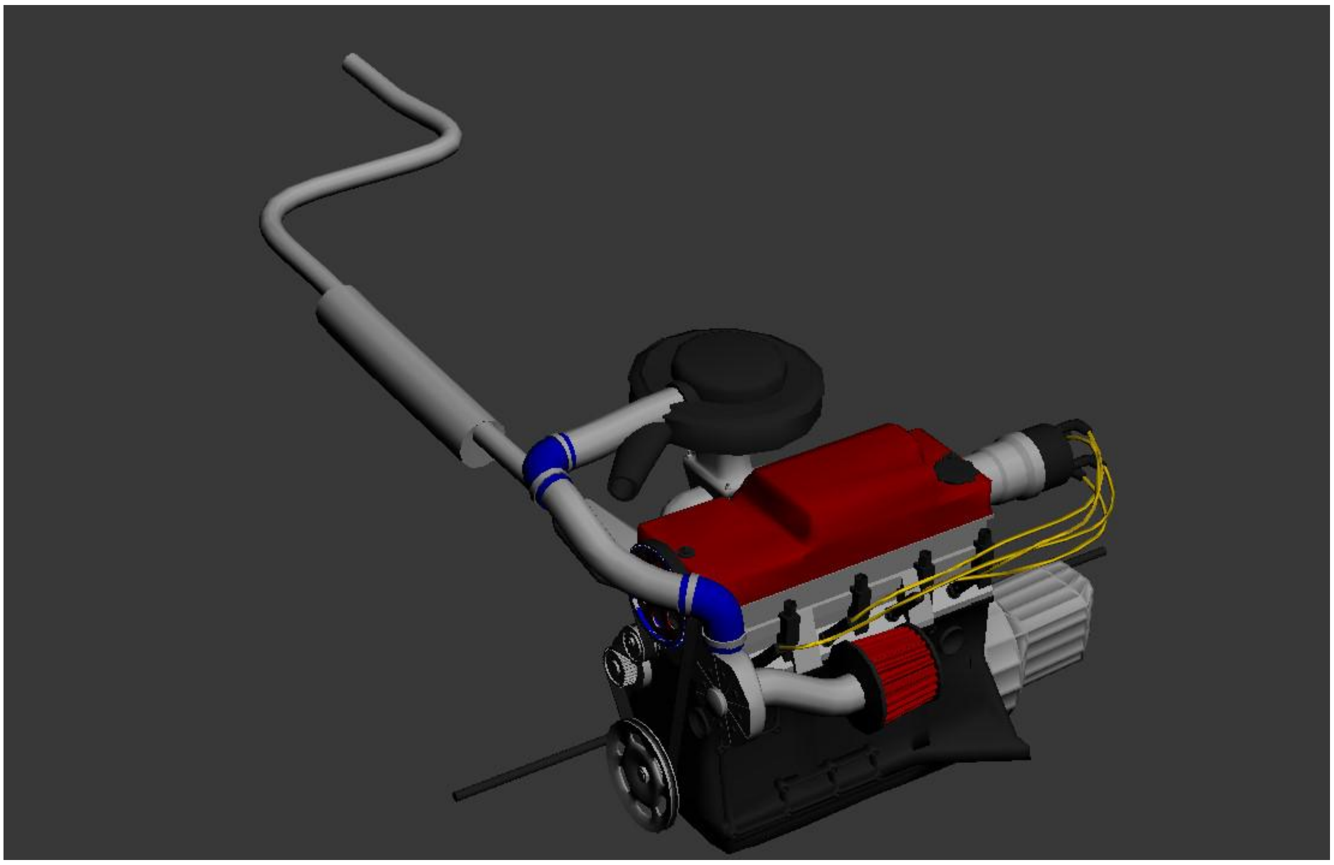
For all the other car parts (especially suspension) put the pivot point at the center of the object.

An example for hood pivot point:



Also, different systems of cars should be exported separately. It means, export Body, Suspension and Engine to different files. It should look like “Body.fbx”, “SuspensionFL.fbx”, “Engine.fbx”, etc.





Again, check all meshes for compatibility with mesh colliders in Unity, and then export body, suspension and engine to clean copies.

Tutorial – Setting up the scene

1.Setting up the scene.

1. Create new camera, name it PreviewCamera. Delete its' GUI layer, Flare layer, and Audio listener. Set Clear flags to Solid color. Then create a new layer, name it previewPart; and set Culling mask to nothing, and then to previewPart. Place the camera somewhere under the ground. Deactivate the preview camera.

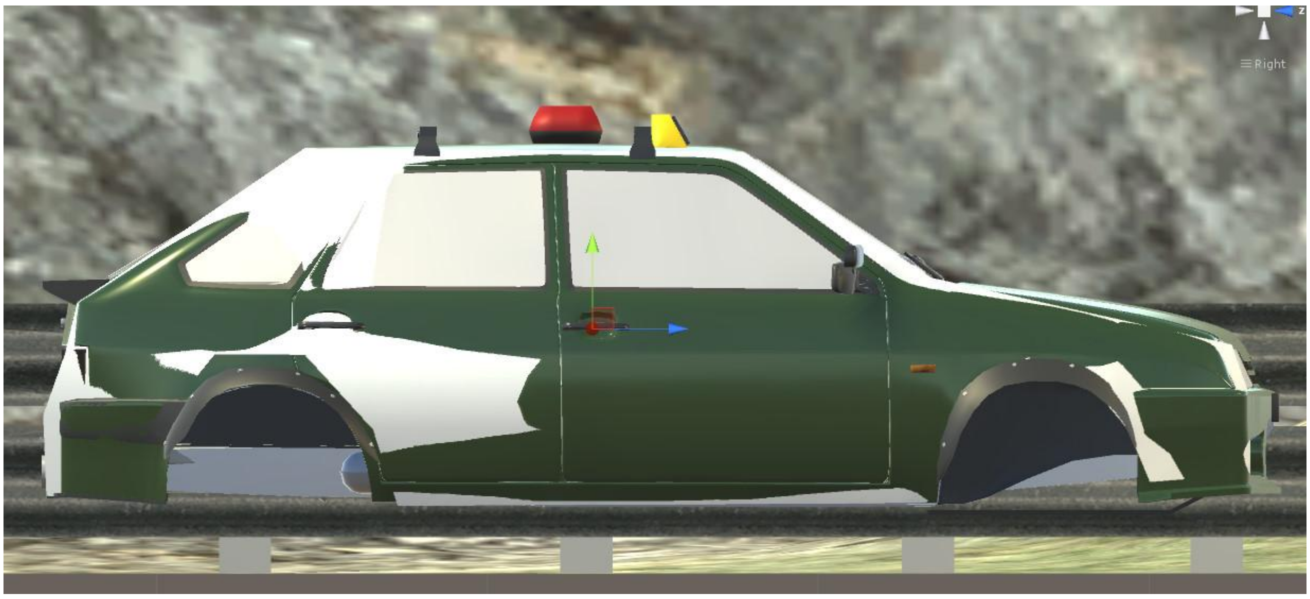
2. Create an empty gameobject, name it Dummies. Then create under it 3 gameobjects: RacetrackDummy, GarageDummy, PreviewPartDummy. Place GarageDummy in center of your garage point, RacetrackDummy at racetrack point, PreviewPartDummy – in front of preview camera.

3. Create 3 empty gameobjects: Cars, Garage, Shop.

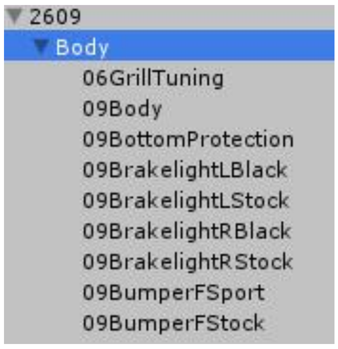
4. Add to main camera 2 scripts: CamControl and CarConstructionCore. Adjust them: ShopGO – Shop, GarageGO – Garage, CarsContainer – Cars.

2. Setting up the car

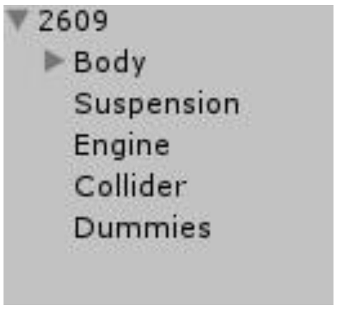
1. Add the car body model to scene.
2. Create a new gameobject with name of your car.
3. Move the new gameobject at the center of the car. This will be the camera target.



4. **Roots**. Create following sub-gameobjects: Body, Suspension, Engine (these 3 should be named exactly this way, as the script searches them by names), Collider, Dummies, and put all body parts under Body gameobject:



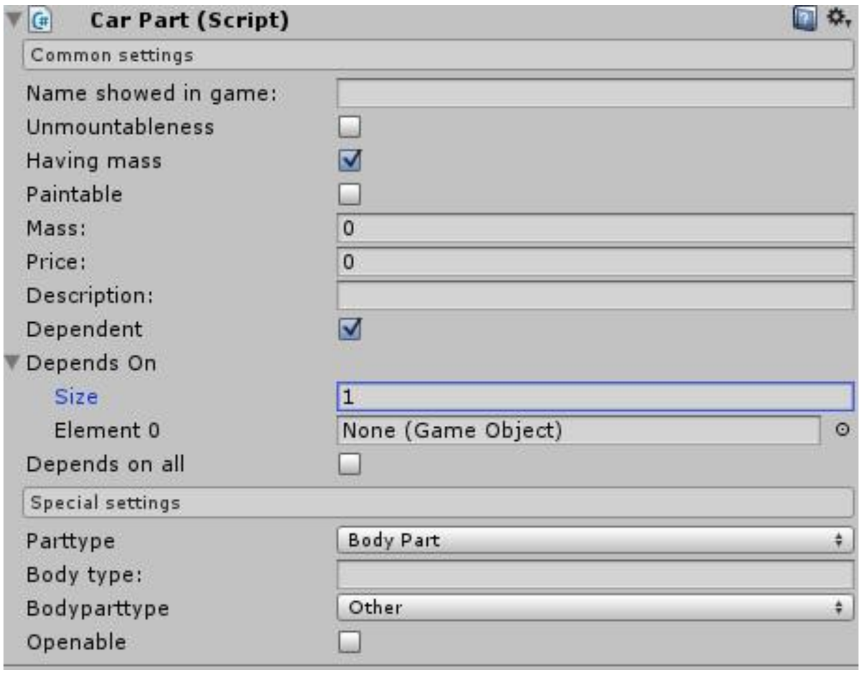
```
▼ 2609
  ▼ Body
      06GrillTuning
      09Body
      09BottomProtection
      09BrakelightLBlack
      09BrakelightLStock
      09BrakelightRBlack
      09BrakelightRStock
      09BumperFSport
      09BumperFStock
```



```
▼ 2609
  ▶ Body
    Suspension
    Engine
    Collider
    Dummies
```

<u>**Notice: it is extremely important to keep the hierarchy of objects of car the way it is in example scene/described here.**</u>

5. **Body: CarPart**. To all body car parts (09GrillTuning, 09Body, 09BumperFSport...) add script CarPart and adjust it. Dont add script to roots!(Body,Engine,Suspension...) Only to parts.



Car Part (Script)

Common settings

Name showed in game:	
Unmountableness	☐
Having mass	✓
Paintable	☐
Mass:	0
Price:	0
Description:	
Dependent	✓
Depends On	
Size	1
Element 0	None (Game Object)
Depends on all	☐

Special settings

Parttype	Body Part
Body type:	
Bodyparttype	Other
Openable	☐

Attributes description:

a. Common settings.

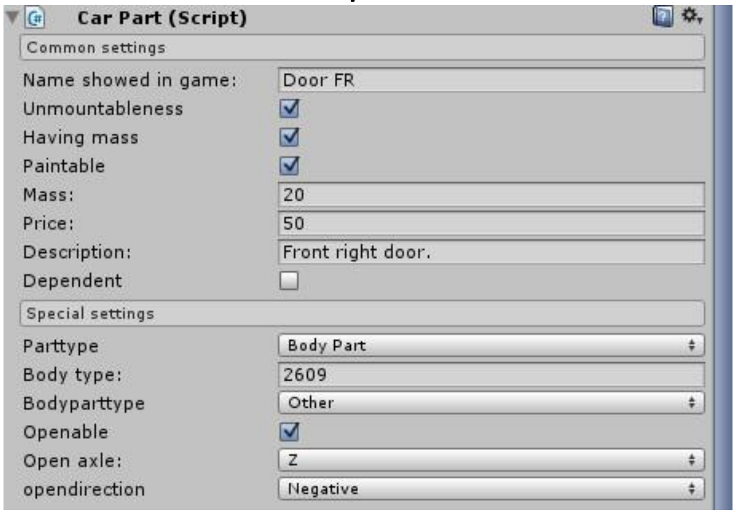
- <u>Name showed in game</u> – name of part showed in parts list and shop.

- <u>Unmountable</u> – if true, the mesh collider component will be added to this part. Check, if it should be unmountable.
- <u>Having mass</u> – if true, rigidbody will count its' mass.
- <u>Paintable</u> – check if the part should be paintable. Notice, that paintable parts should have CarBody or Rim material. (You may add your own paint group in PaintPart() of CarConstructorCore).
- <u>Price</u> – price in shop.
- <u>Description</u> – description in shop under part preview.
- <u>Dependent</u> – if true, DependsOn array will popup. If DependsOnAll is false, the part will be mounted if any of parts in DependsOn is mounted. If DependsOnAll is true, the part will only be mounted if ALL of parts in DependsOn list are mounted. For example, left mirror depends on front left door. It means, we should check Dependent, and put the DoorFL at DependsOn list of MirrorLStock and MirrorLSport. BUT, the timing belt of engine depends on both crankshaft and camshaft, so in this case we put crankshaft and camshaft in DependsOn list, and set DependsOnAll true.

b.Special settings.

- Body type – what car this type is for. This field should match with Car name in ParametersKeeper script of car.
- Bodyparttype – sub type of car part.
- Openable - whether this part should open or not. If checked, it rotates in axis specified in rotationAxis variable, in direction specified in opendirection.

Example:



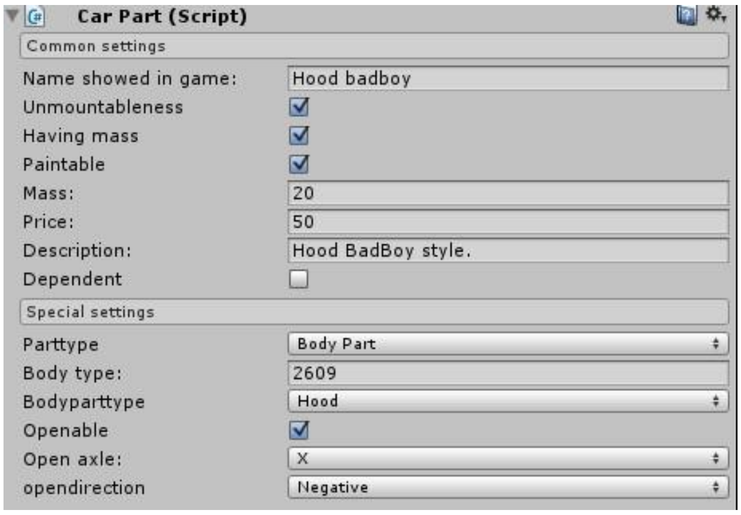
Car Part (Script)

Common settings

Name showed in game:	Door FR
Unmountableness	✓
Having mass	✓
Paintable	✓
Mass:	20
Price:	50
Description:	Front right door.
Dependent	☐

Special settings

Parttype	Body Part
Body type:	2609
Bodyparttype	Other
Openable	✓
Open axle:	Z
opendirection	Negative



Car Part (Script)

Common settings

Name showed in game:	Hood badboy
Unmountableness	✓
Having mass	✓
Paintable	✓
Mass:	20
Price:	50
Description:	Hood BadBoy style.
Dependent	☐

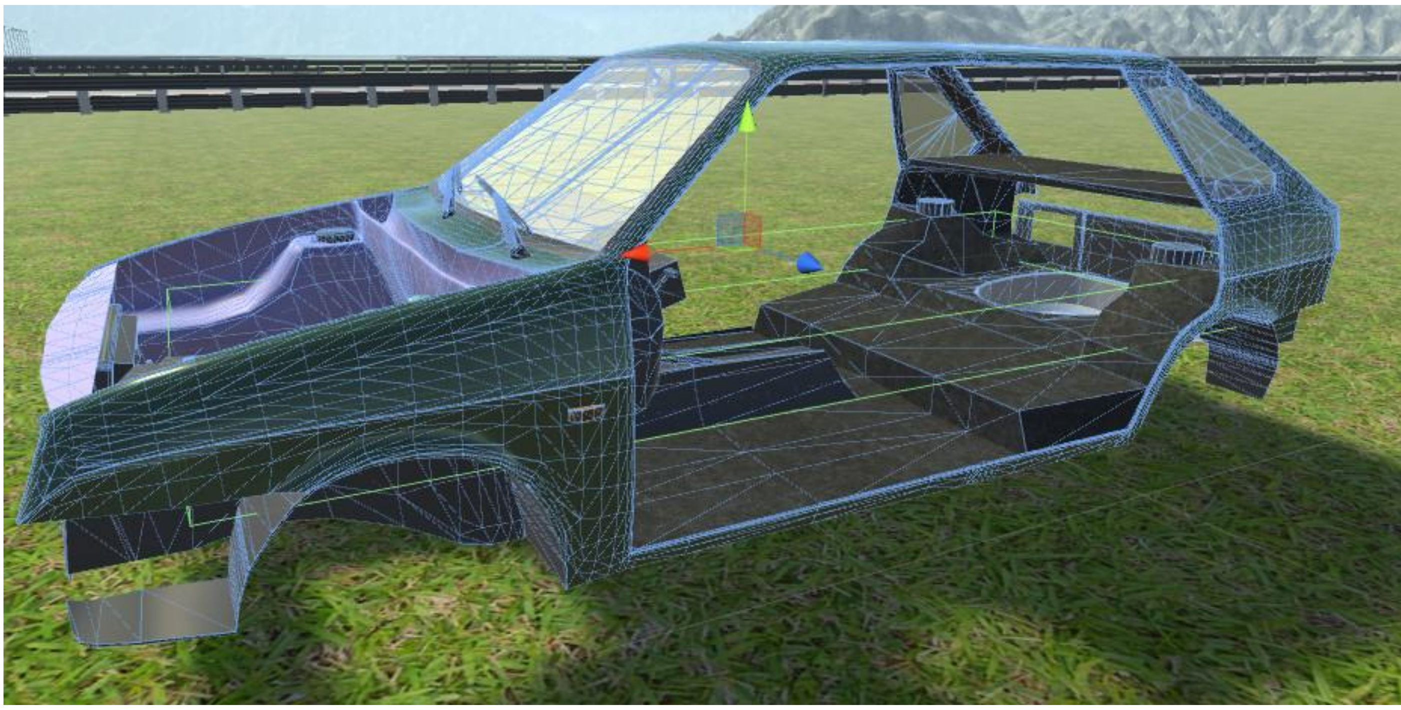
Special settings

Parttype	Body Part
Body type:	2609
Bodyparttype	Hood
Openable	✓
Open axle:	X
opendirection	Negative

CarBody (the car frame) under no circumstance should have checked unmountable box, as this is the car sceleton. CarBody should have mass around 700-900 kg (depending on car parts mass).

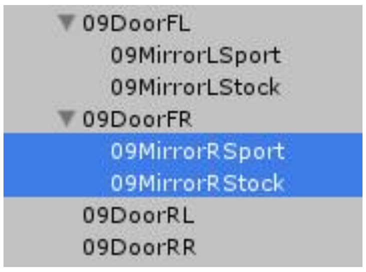
*Yes, this is kinda tiresome, but there’s no other way to configure every car part.

6. Add a box collider component to Car Body. It’s needed to make body able to be painted. Explanation: The painting of all objects uses raycasting and mesh colliders. But car body is not unmountable, so it will not have mesh collider. So, we need to manually add a box collider. Make it less than car body form:



7. **Collider**. Add collider and put it to Collider gameobject. You can use box collider, or create your own mesh, remove renderer and assign mesh collider to it. Assign Ignore raycast layer to Collider.

8. **Parenting.** Make parts that are attached to another parts their children. For example, MirrorLeftStock must be a child of DoorFL.



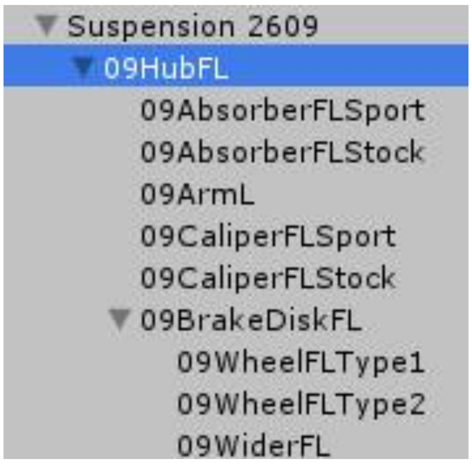
```
▼ 09DoorFL
    09MirrorLSport
    09MirrorLStock
▼ 09DoorFR
    09MirrorRSport
    09MirrorRStock
  09DoorRL
  09DoorRR
```

9. **Dummies.** Create an empty gameobject named EngineDummy. Place it near to where should the engine block be located. In future, we will place it more accurate.

10. **Steering wheel**. If you want the steering wheel to rotate, create empty GameObject as child of Body, name it SteeringWheelDummy, and align it to center of steering wheel. Then make the steering wheel a child of SteeringWheelDummy.

11. We've finished with Body, let's adjust **Suspension**. Add the SuspensionFL into scene. Notice, that the suspension should have

the specific hierarchy that you have to follow strongly. I don't guarantee that dynamic suspension works properly if the hierarchy is broken. So, select all suspension elements, except hub, and make them children of hub. Then, make Wider and all wheels children of Brake Disk. At least, the hierarchy should look this way:



```
▼ Suspension 2609
  ▼ 09HubFL
      09AbsorberFLSport
      09AbsorberFLStock
      09ArmL
      09CaliperFLSport
      09CaliperFLStock
    ▼ 09BrakeDiskFL
        09WheelFLType1
        09WheelFLType2
        09WiderFL
```

12. **Suspension: CarPart.** To all suspension car parts (09AbsorberFLSport, 09ArmL, 09WheelFLType1...) add script CarPart and adjust it.

Attributes description:

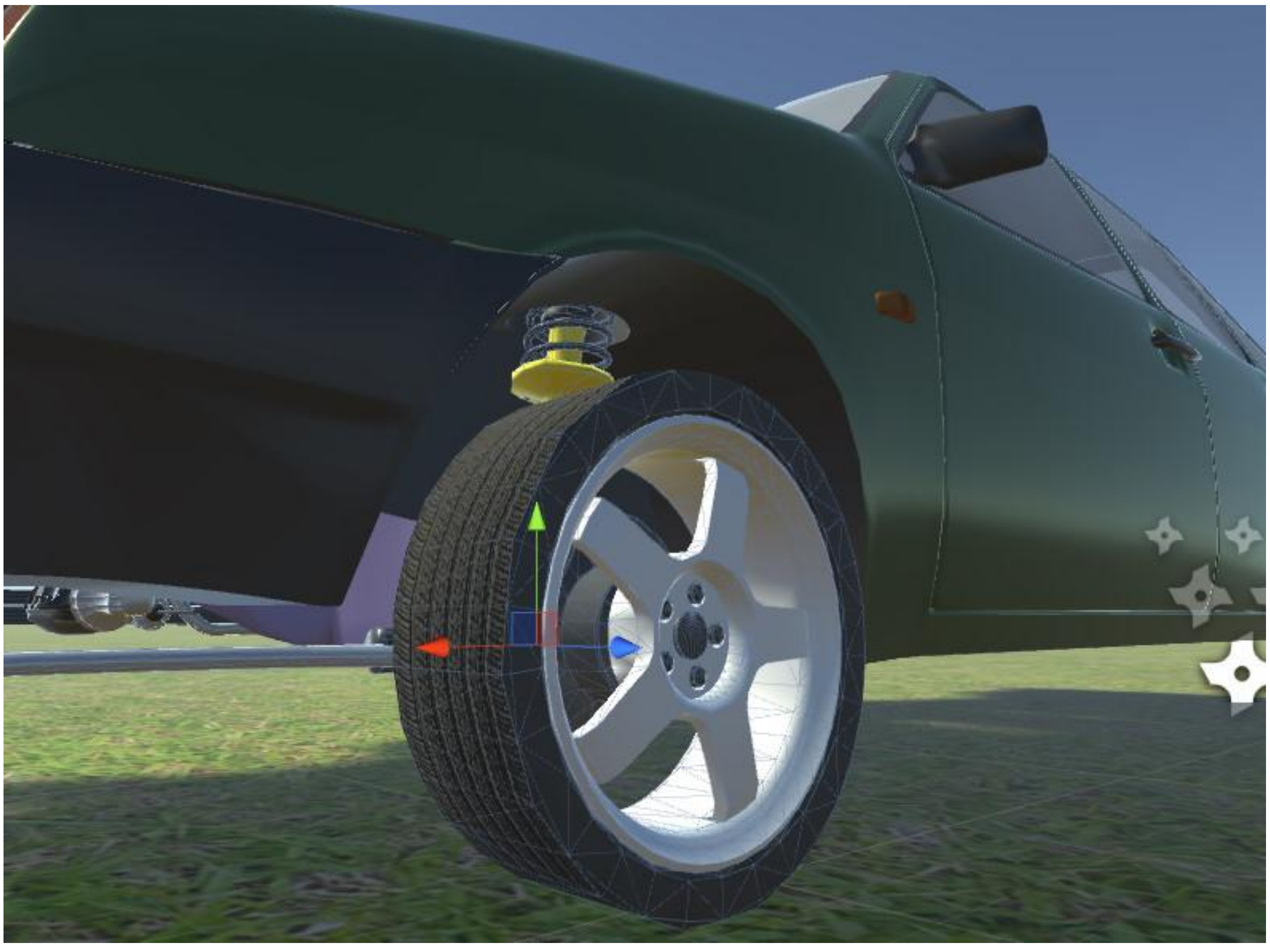
- Suspension part type – sub type of suspension part.
- Suspension Type – just like Body type. Should match to CarName.
- Location – location of part (Front left, Front right...).
- Braking Efficiency (BrakeDisk and caliper only) – brake torque this part adds.
- Spring height, absorber damping, spring rate (Absorber only) – wheel collider parameters this part will send.
- Visual wheel offset (Wheel only) – if the wheel mounts at wrong place, change this parameter.

-Wider offset – the offset this wider adds.

13. Make HubFL a child of Suspension gameobject of your car.

14. **Repeat** 11, 12 and 13 steps for SuspensionFR and SuspensionRear. But the rear one has Beam axle. Adjust its' CarPart script, and let it be rear right hub's child. We'll configure it later.

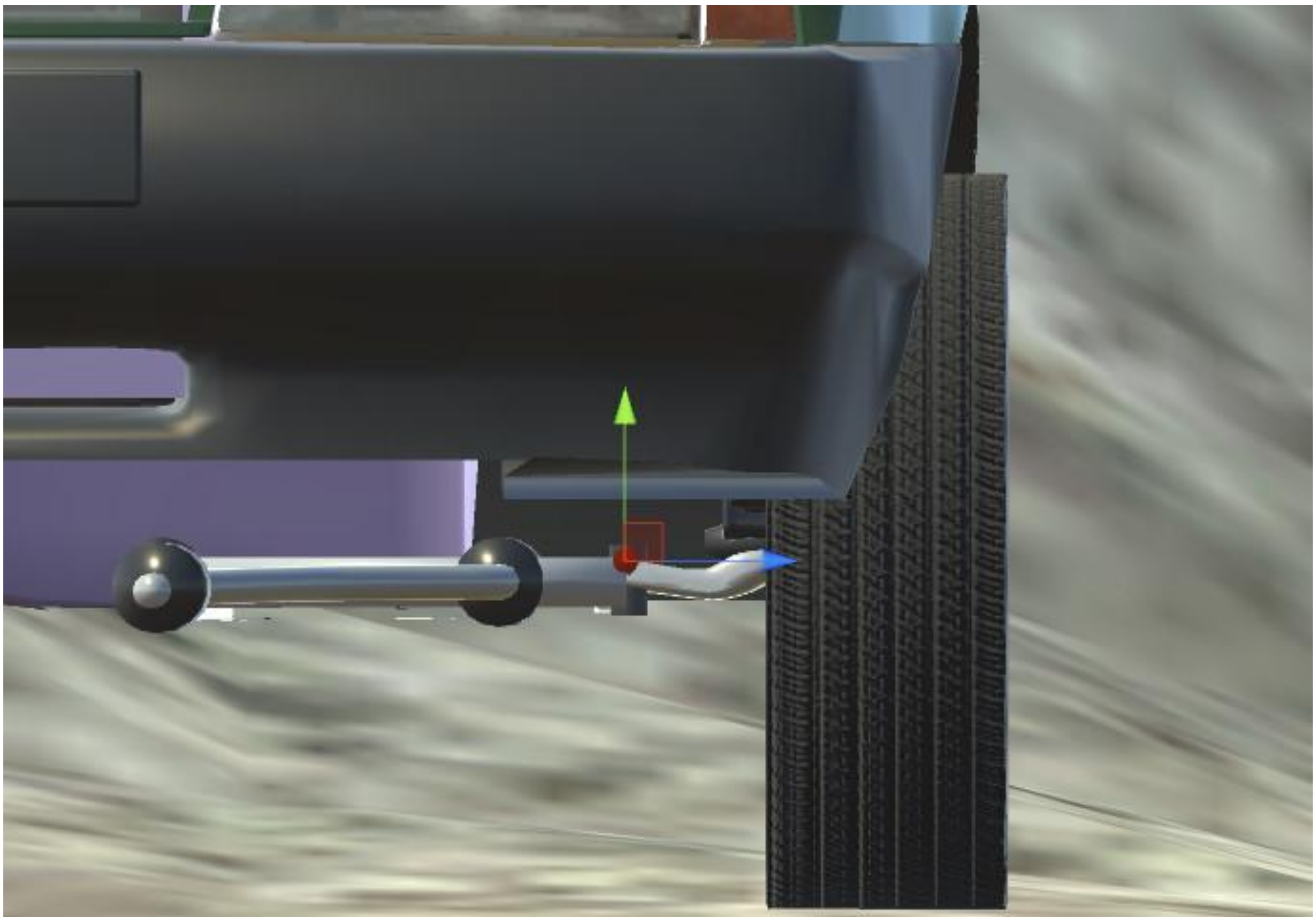
15. **Suspension Dummies.** Now we have to align the suspension relative to car. Select HubFL, and place it at desired place.

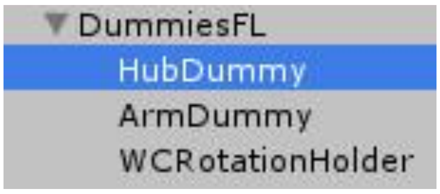


Then, create an empty gameobject, name it DummiesFL, and make it child of Suspension gameobject. After that, create 3 empty gameobjects in it: HubDummy, ArmDummy, WCRotationHolder.

Place HubDummy at pivot point of the Hub, ArmDummy at pivot point of the Arm, and leave WCRotationHolder at its' place. Explanation: in game, when you mount hub /arm, it will take position of its' dummy. WCRotationHolder is a temporary gameobject that holds wheelcollider rotation.

Since the Dummies are set up, you may move Hub wherever you want, their position will be right in game.



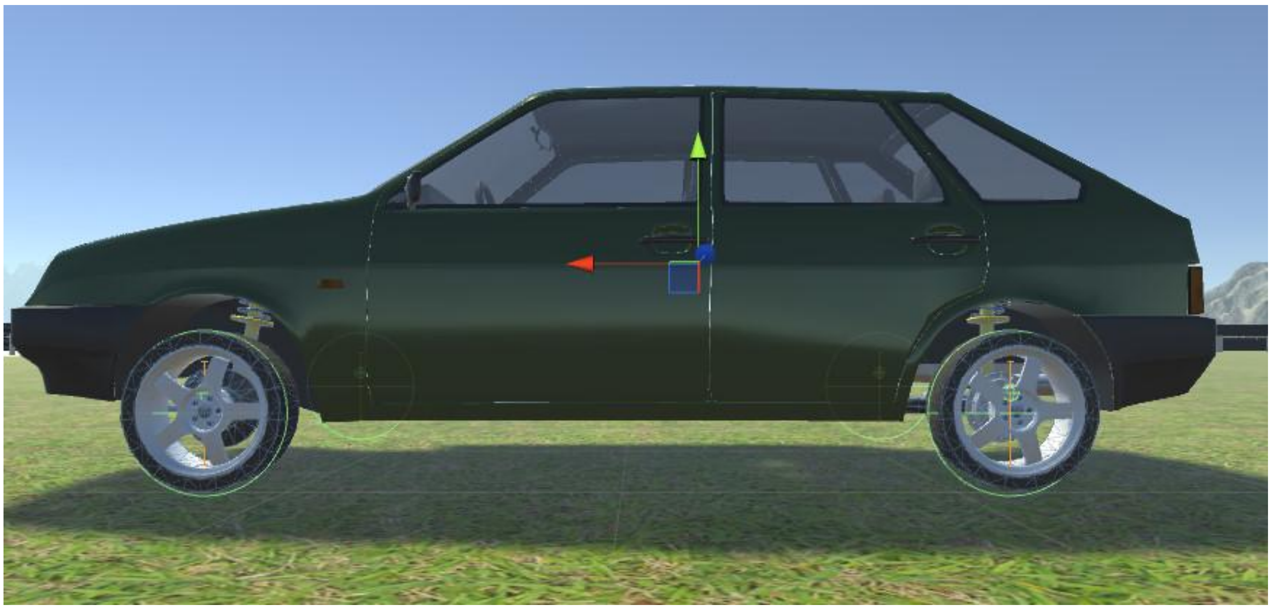


DummiesFL
HubDummy
ArmDummy
WCRotationHolder

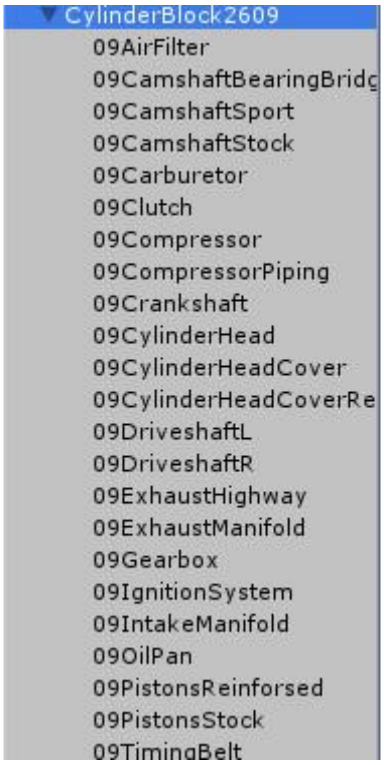
16. Perform **step 15** for other 3 hubs.

17. Now let's configure **Beam axle**. Create an empty gameobject under DummiesRR named BeamAxleDummy, and place it at HubRR pivot point. And then, make beam axle a child of BeamAxleDummy.

18. Create 4 empty gameobjects, and assign then WheelCollider component. Place them in their places, adjust it however you want, and make the radius a little more than visual wheels.



19. The suspension's done. Let's configure **engine**. Add the engine model into the scene. Then select all engine parts and make them children of Engine block (Cylinder block).



```
▼ CylinderBlock2609
    09AirFilter
    09CamshaftBearingBridg
    09CamshaftSport
    09CamshaftStock
    09Carburetor
    09Clutch
    09Compressor
    09CompressorPiping
    09Crankshaft
    09CylinderHead
    09CylinderHeadCover
    09CylinderHeadCoverRe
    09DriveshaftL
    09DriveshaftR
    09ExhaustHighway
    09ExhaustManifold
    09Gearbox
    09IgnitionSystem
    09IntakeManifold
    09OilPan
    09PistonsReinforsed
    09PistonsStock
    09TimingBelt
```

20. **Engine: CarPart.** To all engine car parts (09AirFilter, 09ArmL, 09WheelFLType1...) add script CarPart and adjust it.

Attributes description:

- Engine type – Same to Body type and Suspension type. Should match to CarName of ParametersKeeper.
- Engine part type – engine sub type.
- Spinning – if this part should rotate (pulleys, cogwheels).

- Add power – if this part adds power. For example, camshaft stock may add 50, and camshaft sport – 150.
- Spinning axle – spinning axle.
- Gear ratios (Gearbox only) – gear ratios.
- Top Gear (Gearbox only) – top gear.
- Drive type (Engine block only) – RWD/FWD/AWD.
- Engine sound (Engine block only) – engine sound.

21. Make CylinderBlock a child of Engine gameobject.

22. If the car will be RWD, create an empty gameobject, name it DriveshaftRWDDummy, and make it a child of Engine gameobject. If FWD, create 2 empty gameobjects, name them DriveshaftLDummy and DriveshaftRDummy, and make them children of Engine gameobject.

23. Engine is done. Add the **ParametersKeeper** script to Car root gameobject. Adjust it: assign variables under headers “Assign these variables”. All other are Read only.

24. Add the CarController (or your own) script to Car root gameobject and configure it.

25. Browse the car materials and change them in following way: change parameter Emission to 0.001 to all the materials that have to be highlighted when they’re under mouse.

26. Add the AnimatedCarParts script to Car root gameobject.

26. Make your car a child of Cars.

27.Perform final preparations: run the game, and check engine position – if it’s not okay, change the EngineDummy position.

28. Run the game and check how it works. If everything’s fine, then make duplicates of the Body, Suspension, Engine gameobjects, and make them children of Shop gameobject. Delete unnecessary for shop parts from duplicates: like wheel colliders, dummies, car body, etc.

Important: if you delete some of parts of car’s engine/suspension (for example, for start engine/suspension configuration), check all other’s parts DependsOn blocks: if any one is missing, reassign it from shop’s ones. If you don’t do it, it will cause Null refenerce exception.

Edy’s Vehicle Physics integration

The integration of EVP is pretty easy. Instead of CarController, add to car VehicleController of EVP. Configure its’ Wheels, add wheelcolliders, and assign WCRotationHolders to Wheel Transforms (!).

If you want to apply engine power to Vehicle Controller, extract the CompatibilityWithEVP archive, and add ApplyMotorpowerToEVP script to the car.

That’s all, the EVP works with KCC. Do all further steps according to EVP manual.

Some script explanations

If you want to change car part types list, do following steps:

1. In CarPart script under enum BodyPartType/EnginePartType/SuspensionPartType add or remove items. Make sure to add new types only at the end of the enum.
2. In CarConstruction script configure following functions: checkEngine(), checkSuspension() and checkBody(). These functions define which parts are necessary for car and which are not. If any of parts mentioned in these functions is not installed on car, then car won't go.
3. Adjust in inspector variables of CarPart script on car parts gameobjects and you're done.

The installation of car parts is being stored in bodyparttypeindex,engineparttypeindex and suspensionparttypeindex arrays. Here's how it works: at start the script checks all car parts. For example, when it meets Bumper front stock, it puts bodyparttypeindex[3] as true. When script meets Bumper front tuning, it makes the bumper front tuning gameobject inactive as the front bumper is already installed (bodyparttypeindex[3] is true). Then, if you uninstall front bumper, bodyparttypeindex[3] as false, what makes you able to mount another bumper. The suspension and engine work same.

Known issues

- Steering wheel doesn’t rotate/rotates in wrong axis.
  Make sure that you assigned SteeringWheelDummy to ParametersKeeper script. If it rotates in wrong axis, unparent steering wheel from SteeringWheelDummy, rotate SteeringWheelDummy in needed axis, and parent steering wheel to Dummy again.
- Hub/arm turned in a wrong axis.
  The solution is same to steering wheel, unparent-rotate-parent.
- Hub travels in horizontal axis.
  Select this hub, and change its variable Hub turning axis. To define what axis is right, move it up and down and look at its transform component: the one that changes is the right one.
- Wheel mounts on a wrong place.
  Make sure that brake disk’s pivot point is at the center of brake disk, and if it is, change the Visual wheel offset of the Wheel.
- Driveshaft/Beam axle is rotated wrong.
  The solution is the same as for steering wheel: unparent-rotate-parent.

Feel free to write to keereedev@gmail.com if you have any issues.