Universal Wheel Physics

General description of package:

Universal Wheel Physics, is raycast-based wheel behaviour solution alternative to standard Unity's wheel collider(WC). Biggest difference between them is that UniversalWheel(UW) is more... universal!(who would guess!). That is mainly because you are not forced to set it at fixed position and angle, so you can have wheels even at your vehicle's roof! Also there is no 20 WC per vehicle limit, and simulation is more stable.
Default UW settings are suitable to typical, 1200kgs car.
I hope you will have as much fun using this package as I had developing it :)

Setup guide:

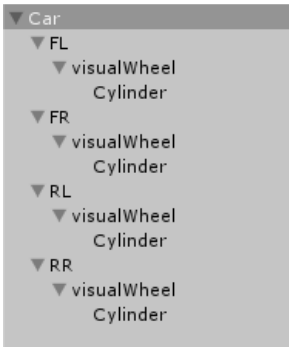
There is example scene and prefabs for you to check out, play with, and see how everything works.
In prefab folder you have simple car that is effect of following below instructions.

1.Creating our vehicle's main body:

-Create empty game object, and call it with your vehicle name, I name it Car.
-Add rigidbody component to it and set mass to 1200.
-Add boxCollider component to it, and set it's size to 2,1,4
-Create Cube, set it as child of our Car, set it's position to 0,0,0, set it's size to 2,1,4. It will be our visual representation of car's body

2.Creating our vehicle's wheels:

-Create empty child object and add UniversalWheel.cs component to it
-Drag your vehicle's root to rigidbody field of UW, to assign our vehicle's rigidbody
-Create empty child for our UW game object, and set it's position to 0,0,0 and rotation to 0,0,0 and name it visualWheel
-Create cylinder game object and set it's Z rotation to 90, and Y scale to 0.1(to make it look like a wheel)
-Assign visualWheel to our Visual Wheel field of our UW component
-Set our cylinder to be child of visualWheel and set it's position to 0,0,0
-Now duplicate our UW to have 4 of them
-For first two of them set Z position to 1.5
-For last two of them set Z position to -1.5
-For first and third set X position to -1
-For second and fourth set X position to 1
-Set for all of them Y position to -0.5
-Select your all wheels, and add them UniversalWheel.cs component
-You can name them accordingly for convenience(FL,FR,RL,RR)
-Now your car's hierarchy should look like this:



3.Driving:

-Now just add him ForcesController.cs component, and assign all UW into appropriate slots in ForcesController, and everything is ready! You can press play, and use arrow keys to move car, space to brake and right shift to hand-brake.
Have fun :)

Class members description:

UniversalWheel.cs is main class that handles whole simulation of wheel behaviour.

Fields:
draw debug lines – draw general visual representation of wheel's shape, with suspension travel, visible in Unity Scene view. Lines color changes to represent current suspension state: green means completely expanded, blue means partial compression, red means bottoming out.

collision mask – pretty self-explanatory, here you can set which layers you want your wheel to collide with.

rigidbody – rigidbody affected by wheel. All wheel's forces will be applied to it. This rigidbody doesnt have to be your wheel's parent.

visual wheel – object(transform) that will be visual representation of your wheel. Universal Wheel Collider handle it's rotation , so just assign your wheel here, and it will rotate properly.

affect other rigidBodies – do you want your wheel to push back other rigidbodies with it's suspension(spring and damping) force. Your wheel's suspension will react and compress when in contact with other rigidbodies anyway. If you want it to completely ignore some objects, use collision mask.

fluent wheel movement – normaly, when your car drive on square-shaped obstacle, your wheels will teleport from ground to this position in one frame(teleport way), which looks strange and unrealistic. To precent this, you can use this option, so wheel movement will be spreaded of few frames instead of one, so it will look fluent and more natural.

Visual wheel move speed – the bigger value- the faster visual wheel position will change to it's physical equivalent.

Mass – mass of your wheel. It affects how torque is applied to your wheels, bigger mass-slower acceleration. This value is also used to calculate wheel's rotational inertia.

Radius – physical radius of your wheel. When draw debug lines is on you can see your wheel's radius in scene view as circle.

Suspension travel – your spring/damper lenght. It defines how far your wheel can move downwards. For example off-road cars have big suspension travel to better move over obstacles, when sports car will have small suspension travel, as they dont need it on flat roads. Car with small suspension travel also handles better in corners.

Spring rate – your wheel's spring "stiffness".

Spring preload – how much of total spring force is applied at the very beginning of suspension travel(with suspension uncompressed). Defined in percentage: 0 = 0%, 0.5 = 50%.

damping – shock absorber's damping value. Force that "slows down" suspension movement. Here is good explanation - https://en.wikipedia.org/wiki/Damping

forwardFrictionCoef - coefficient of forward friction.
forwardSlipCoef - coefficient of value that is feed to forward friction curve(on it's x axis).
sidewayFrictionCoef - coefficient of sideway friction.
sidewaySlipCoef - coefficient of value that is feed to sideway friction curve(on it's x axis).

maxSideForce – maximum force that can be applied to car in wheel's sideway direction. Can be useful to prevent car from strange behaviour(jitter) in extreme conditions, for example with very big masses and forces. When set to 0 its ignored. Should never be lower than 0.

maxForwardForce - maximum force that can be applied to car in wheel's forward direction. Can be useful to prevent car from strange behaviour(jitter) in extreme conditions, for example with very big masses and forces. When set to 0 its ignored. Should never be lower than 0.

forwardFrictionVars/sidewayFrictionVars – these are vector4 variables that defines Pacejka's friction curve's shape. Every time you change one of it's values, friction curve is recalculated and updated.
For more details read here -
https://en.wikipedia.org/wiki/Hans_B._Pacejka#The_Pacejka_.22Magic_Formula.22_tire_models

axisTorque/axisBrake – these are values that you should set to apply rotational forces to your wheel. UniversalWheel.cs never changes these values.

Properties and functions:
float RPM – returns current wheel's rpm(revolutions per minute).

Vector3 WheelPos – returns current physical wheel position in world space.

float SuspensionTravel – returns distance from your UW transform position to current WheelPos.

float SidewaySlip and ForwardSlip – returns speed difference between wheel and ground for appropriate sideway/forward movement of wheel (of wheel's local vectors).

bool HasContact – returns true if wheel is actually touching ground.

bool GetGroundHit(out RaycastHit hit) - returns true if wheel is actually touching ground. If it is, then it fills hit structure.

//
Recommended to use lower or equal to default 0.02 fixed timestep, with bigger values UWP wont work properly. You can set this in Edit -> Project settings - > Time.