WHEEL CONTROLLER 3D

Manual for version 3.x

Setup

Steps to set up your vehicle with WC3D are much similar to the steps needed for the default wheel collider. This setup guide will assume you have the vehicle set up for vehicle collider and only show steps needed to use such vehicle with WC3D.

Before starting make sure that your model has proper rotation which for Unity is Z axis forward, X axis right and Y axis up. Most models made for other software than Unity tend to have Z axis up and this needs to be fixed before starting. Here is the official guide:

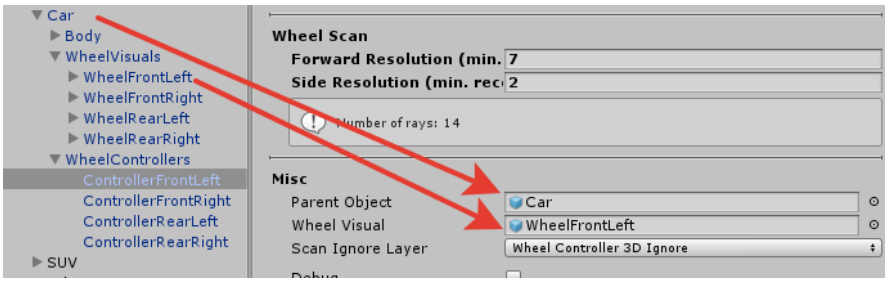
https://docs.unity3d.com/Manual/HOWTO-FixZAxisIsUp.html

WC3D, unlike wheel collider, can handle wheel rotation and positioning by itself. If the wheel consists of multiple parts add all the components of the wheel as children to an empty game object and then assign that object as the wheel visual (more about that later).

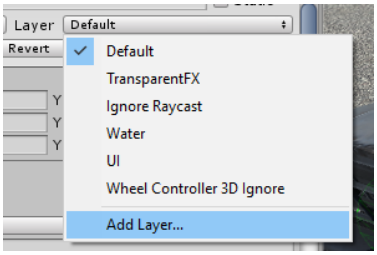
Wheel's visual object also needs to have proper rotation AND pivot point set to the center of the wheel for which you can also use the guide in the link above.

Converting a Vehicle From WheelCollider to WC3D

1. Remove WheelCollider component from all the wheel collider objects and add WheelController scripts using “Add Component” button.
2. Add “Parent Object” (required) and “Wheel Visual” (optional) objects to the bottom of WC3D script:



3. Create a new layer:



4. Select that layer at the bottom of the WC3D editor in the field “Scan Ignore Layer”. All the colliders on the vehicle need to be set to this layer as else wheels might collide with the vehicle itself and cause vibration / jumping or similar.
5. Set rigidbody interpolation to None. Important for stable spring, damper and friction calculations – highly recommended step. Asset will still work with interpolation but there might be effects like vehicle acting funny at high speeds if enabled.
6. WC3D will set all the values and curves to their defaults. At this point you will want to adjust “Tire Radius” and “Wheel Width” fields of the inspector and check if “Side the wheel is on” field is correct. If the vehicle side has been detected incorrectly there is a large chance that you model is not properly rotated (check initial paragraph).
7. If all the steps have been done correctly pressing play will now result in a working wheels and suspension. If not, check the next page for troubleshooting.

Moving from WC3D v2.x to v3.x

There have been some major changes from version 2.3 to 3.0 and so changes to existing vehicles and scripts will be needed. Most of these changes are either to simplify things, add new features of because of performance reasons.

Here are some things that will need changing:

1. Namespaces - v3 uses NWH.WheelController namespace for all the scripts included. Error "The type or namespace name..." can be solved by adding "*using NWH.WheelController3D;*" to each of the scripts that reference WC3D.
2. WheelController.FrictionPresets is now WheelController.FrictionPreset and has different usage inside scripts:

   ```
   wheelController.SetActiveFrictionPreset(WheelController.FrictionPreset.Ice);
   ```

   There is no longer an option / need to select lateral and longitudinal friction presets separately.
3. Camber is now a curve but can still be set using float value with the new function:

   ```
   wheelController.SetCamber(float value);
   wheelController.SetCamber(float camberAtTop,float  camberAtBottom);
   wheelController.SetCamber(AnimationCurve curve);
   ```

   Current camber value can now be retrieved through `wheelController.camber`.
4. Scan Ignore Layer will need to be set again due to the new dropdown selector.
5. Slip calculations are much tighter now than before and so some parameters of the forward and side friction might need to be reduced to get the same behavior. Spring and damper will behave mostly the same.

Troubleshooting Common Problems

Here are some of the most common problems with WC3D setup:

1. There is no rigidbody and / or valid collider on the vehicle. This will cause vehicle to fly into the air and jump around wildly.

   a) Add rigidbody / collider.

2. Scan Ignore Layer has not been set.

   a) Add Scan Ignore Layer and do not forget to apply it to all of the colliders on you vehicle

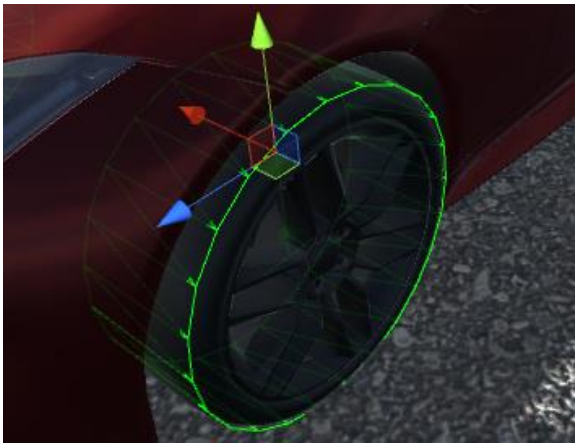
3. Vehicle is vibrating.

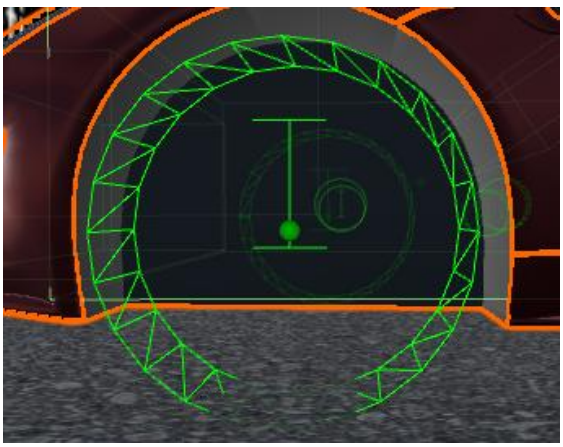
   a) Check that your vehicle has reasonable weight for the default values or change the values on the WC3D. Default values are for a vehicle with rigidbody mass of about 1200-2000.

   b) Adjust your center of mass with "CenterOfMass.cs" script in WheeController/Demo/Scripts folder. Too high center of mass wil cause both WheelController3D and wheelcollider to vibrate. Apart of adjusting center of mass you can also change the Fixed Timestep to 0.015 or 0.01 in the Time settings of Unity project but this will affect performance.

4. Vehicle is just gliding around / spinning / going in wrong direction, etc. This means that model's rotation is wrong.

   a) Check the "Debug" field in WC3D editor and turn on Gizmos in the editor. You will see some rays originating from the center of the wheel. Green ray should point forward, yellow to the inside of the vehicle and red one up. Also, the green **I** shape should mimic the spring position inside a car. If this is not correct, either your car's model rotation is wrong or the object containing the WC3D script is not rotated properly. This is what an object containing WC3D should be positioned like:





Editor Field Explanations

Wheel

Side the wheel is on	This is auto-detected but if needed can be adjusted manually. Determines direction of offsets, camber, etc. Use “Center” for bikes, trikes and other vehicles that have single wheel per axle.
Tire Radius	Total radius of the wheel.
Wheel Width	Width of the wheel.
Wheel Mass	Mass of the wheel. Affects how fast the wheel will spin up when there is no traction / wheel is in the air.
Rim Offset	Offset of the wheel from the center. Equivalent to the ETxx marking on the rim of a vehicle. Larger value will move the wheels outwards. Changes how steering behaves.
Use Rim Collider	Option to use a collider mimicking the top half of the wheel to prevent objects passing through wheel from the side and wheels from passing through ground under extreme velocities when raycast fails to detect collision in time. Not needed on “normal” cars but useful on vehicles such as monster truck where most of the wheel is outside of the vehicle.

Geometry

Camber Curve	Animation curve where X axis represents wheel travel in range from 0 (spring fully compressed, wheel is up) to 1 (spring fully extended, wheel is down). Y value indicates the camber value. Negative camber means that top of the wheel will be closer to the center of the vehicle than bottom.

Spring

Max Spring Force	Force that will be applied when spring is fully compressed. Too low value will cause vehicle to bottom out too easily resulting in jerky behavior and too high value might cause shaking.
Spring Travel	Spring travel from fully extended to fully compressed. If using with standard fixed update of 0.02s (50Hz) going under 0.15 might be problematic because of large intervals between frames which means that one frame spring can be fully extended and the next bottom out.
Spring Force Curve	Curve that shows how spring force will be distributed along the spring travel. X axis is spring compression percentage and the Y is force coefficient with which “Max Spring Force” field will be multiplied. Should start at X, Y = 0 and end at X, Y = 1. Straight line between those two points is usually adequate.

Damper

Bump N/m/s	Indicates the force in Newtons that will be applied for spring velocity of 1 meter per second when spring is compressing.
Rebound N/m/s	Same as Bump but when spring is extending.
Damping curve	Curve where X indicates absolute value of spring velocity and Y coefficient by which Bump and Rebound values will be multiplied. Default curve starts at X,Y = 0 and ends at X = 100, Y = 400. Curve ending at X = 1 is too short and will result in wobbly behavior as spring velocity can be higher that 1 m/s. It is recommended that curve ends at least X = 10 or even higher for longer springs. E.g. spring of 1 meter that is compressed inside one frame of 0.02s will result in velocity of 50 m/s.

Friction

Preset	Sets the friction curve to one of the presets. Presets can be added / edited in the WheelController.FrictionPreset.cs script If needed. More about this in the following chapter.
Slip Coefficient	Slip is multiplied by this value (X axis of the friction curve). Larger value will make vehicle more prone to snap oversteer and drift while lower value will make vehicle appear to slide more before the wheel “grips”.
Force Coefficient	Force applied to the wheel is multiplied by this value (Y axis of the friction curve). Higher value will mean that vehicle will grip the surface better and appear to have more arcade-y feel. Lower value will make the vehicle appear as if on ice.
Max. Force	Max friction force that the wheel will be able to apply to the vehicle. Useful for drifting games. Leave at 0 for no limiting.

Wheel Scan

Forward Resolution	Indicates number of rays that will be cast in the forward direction along the plane with normal of wheel.right. Minimum value is 5 and odd values are recommended.
Side Resolution	Number of parallel planes equally spaced along the direction of wheel.right. Setting this value to 1 (minimum value) will cause wheel to detect along the lateral center only, while values of 2 and higher will cover the whole wheel.

Misc

Parent Object	Parent (root) object of the vehicle. Has to have rigidbody component. This field is required.
Wheel Visual	Object(s) that contain wheel meshes and is/are visual representation of the wheel. This field is optional, visual objects can be moved by some other script if needed just like with WheelCollider.
Scan Ignore Layer	Layer which will be ignored by the wheel's rays. If the vehicle contains colliders that do not have this layer set wheel might detect them as ground and start jumping around or even floating.
Debug	Turn on to see direction, normal and other rays along with ground detection point. Requires gizmos to be turned on.

Vehicle Behavior With Different Friction Settings

To help with vehicle setup this chapter describes how a vehicle will handle with different settings.

WC3D works on the slip based principle. Tire always has some slip but it is very small under normal condition – always near the left side of the curve below.

Image above shows friction curve with slip on the X axis and friction on the Y axis. Part of the curve on the left side marked as “has traction” is where tires normally operate, and part on the right side is what would be called “drifting”, “wheel spin” or similar. Depending on the shape of the curve vehicle will act differently upon losing traction. Here are parameters of the curve and how they affect handling:

- **B** (stiffness) – With higher value curve will be faster to reach peak value meaning it will be steeper. This means that wheel will require less slip to lose traction and such vehicle will be more unstable – i.e. spin out more easily. With lower values of stiffness vehicle will be more stable and harder to spin but with very low value it will start to act as if on ice.

Figure 1 Very relaxed friction curve.

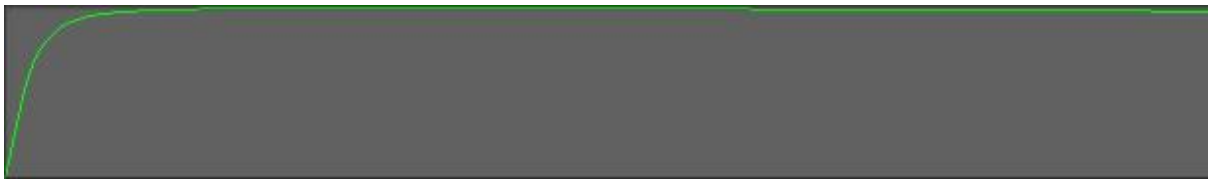


Figure 2: Very stiff friction curve.

- **C** (shape) – While stiffness determines where the peak will be, shape determines how strong the knee in the curve will be. High shape value will make it pointy while low value will make it more flat.
- **D** (peak) – The higher it is the more traction wheel will have. It does exactly the same thing as Slip Coefficient field does. To simplify, resultant friction is equal to

peak value times slip coefficient times some other values - this means that if peak is set to 0.5 and slip coefficient to 2 result will be exactly the same as if both were set to 1 since 0.5*2=1*1.

- **E** (falloff) - Determines how fast curve will fall off after the peak. It depends on other parameters but in general lower value will mean that curve ends closer to 0, and value of 1 will mean that it is almost horizontal after the peak. If set to low value vehicle will start to slide once traction is lost and if set to high value there will be almost no difference between "has traction" and "lost traction" parts of the curve.
- **Slip Coefficient** - X axis of the friction curve is multiplied with by this value meaning that for the same slip value wheel will lose traction faster if set to >1 and it will lose traction slower if set to
- **Force Coefficient** - Y axis of the friction curve is multipled by this value meaning that with higher force coefficient wheel will be able to put more force down for the same slip. To high a value (around 6 or more - depends on the vehicle) might make vehicle unstable as forces will start becoming so high that even a smallest amount of slip could make the whole vehicle jump.

FAQ

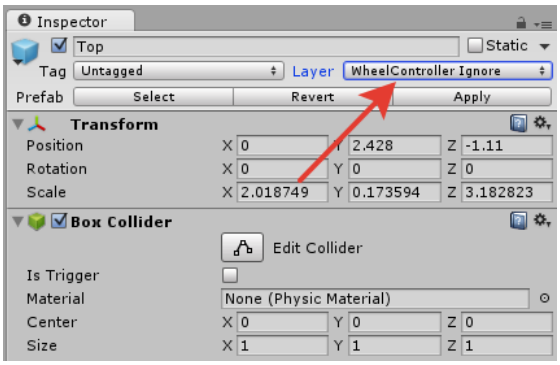
1. Why does my vehicle jump around?
    a. It might be that the WheelController is colliding with your vehicle's colliders (colliders representing the body of the vehicle). Same as #5 in this FAQ.
    b. Damper values are 0 or near.
    c. Damper and/or spring curves need some tweaking. Since force values are multiplied by the value of Y at given X, graph that is always at 0 will result in no spring or damper force. If you are using default curves this is not the problem.

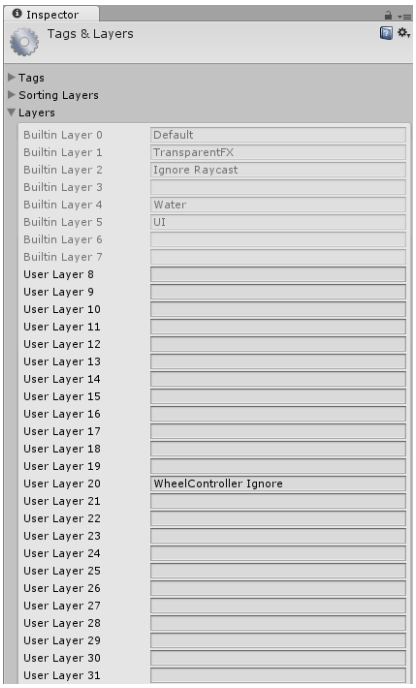
2. I already have a script that is already moving the wheel visuals, but your script has a „Visual“ gameobject field – what do I do?
    a. „Visual“ field in the WheelController's inspector is not mandatory. If you leave it empty everything will work just fine – except the visuals will not be moving – the same as with the Unity3D's default wheelcollider where you have to assign rotation and position of the visual by yourself. You can use GetWorldPose() function for that (more details in API manual).

3. Instead of being still, my vehicle is moving forwards-backwards-forwards-... between obstacles.
    a. By default a vehicle made with this asset does not have any drivetrain losses (e.g. oil in transmission, axles, tire losses). If the wheel is between two obstacles and hits one of them, it will bounce of and start rolling in other direction until it hits the other obstacle and so it will bounce again. That will repeat inifinitely (think of a perfect bouncing ball with no friction or energy losses of any kind). To prevent such behavior apply a little brake torque to the wheels at any moment (you can apply motor and brake torque at the same time, unlike with the default wheelcollider). Brake torque will then act as a drivetrain loss.

4. I scaled the vehicle and it is floating above the ground (or similar).
    a. This is due to radius of the wheels (and everything else) in the script being in world size and not scaling when you scale the parent. Turn on debugging in the script and gizmos in the editor so you can see the wheel. Adjust wheel radius, wheel width, rim width, spring length and other parameters that are size-related and you might also want to adjust rigidbody weight (if making toy cars, etc.).

5. Vehicle just flies up into the air.
    a. This is probably due to WC3D detecting the vehicle as ground. Even if it visually may seem like there is no chance that wheel is touching one of the vehicle's colliders, this may happen due to the discrepancies between positions in Update() and FixedUpdate(). Since all the physics positions in this script are calculated in FixedUpdate() that is running at different frame rate than Update() and since physics positions are not interpolated between steps (unlike the vehicle's meshes and colliders), at higher vehicle speeds there might be significant offset between the wheel center and the actual physics calculation which may result in raycasts „drifting“ into one of the body colliders. To fix this problem follow steps 3 and 4 from the Setup guide at the beginning of the manual.





6. Jittery wheel on flat surface.
    a. This might be a result of too powerful spring or damper configuration or too steep curve. It also can occur if the curve doesn't start at [0,0]. What happens is that when wheel touches ground spring or damper overreact and push the wheel off the ground. Same thing happens over and over again and what you see is vibration.

7. Camera is jerky.
    a. This is a result of turning rigidbody interpolation off. The only way to fix it is by updating your camera in FixedUpdate() so that the camera position is updated at the same time as the object. This is not optimal for performance but it is the best way to prevent interpolation messing with

transform.position – especially at higer velocities. When interpolation is enabled rigidbody position is always somewhere between position from prevoius frame and target (current) position meaning it lags behind. This makes spring and damper react later than optimal which causes over-reaction and you get jumpy vehicle. Also, at higher speeds transform.position will start reporting inaccurate position that can be up to the wheel's radius behind the actual wheel.

For any questions, problems and suggestions contact us at

info@nwhcoding.com