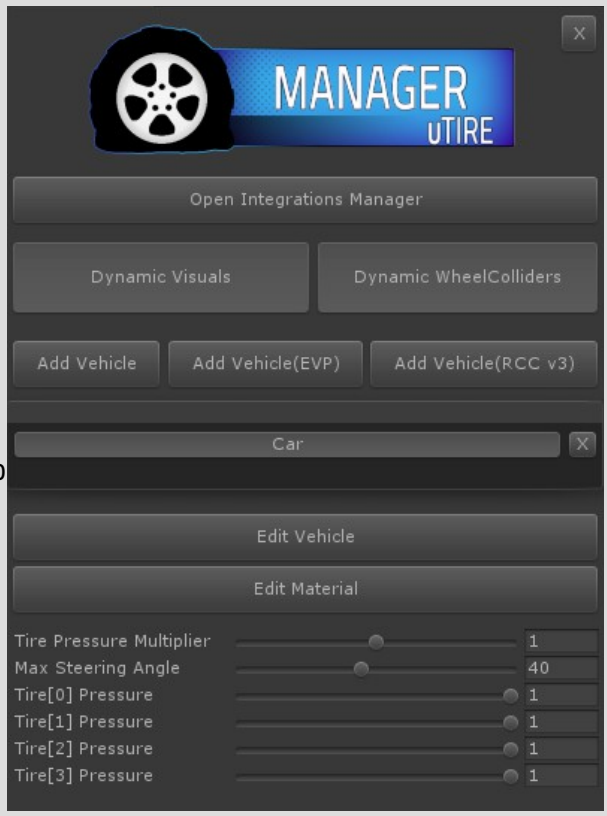
uTire

Manual

What is this asset about?

It's a solution for visual tire deformation. When the wheelCollider is compressed(calculated automatically) or underpressured(you can change this manually) the tire will appear flat. During hard breaks, or fast turns it can really add an extra bit of realism at a tiny performance cost – as it's not just a simple bulge\flattaning effect, when the vehicle goes fast enough and turns the tire will appear to be sticking to the road as well. Might be a bit hard to understand, but as soon as you see it in motion it makes sense.

Quick Getting Started Guide

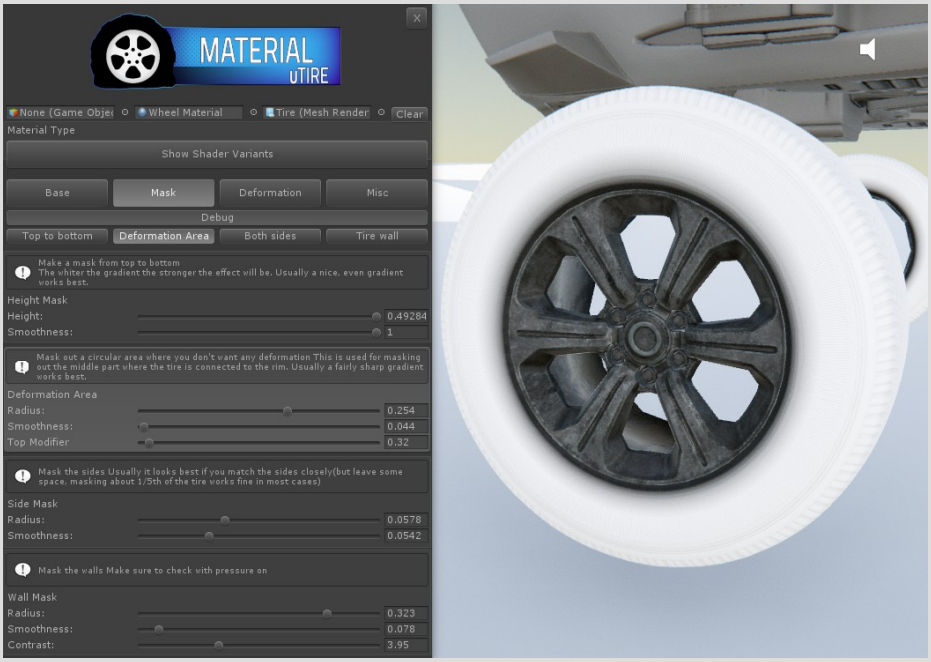


- Go to Tools\TSD\uTire Manager, this will create a new object in the scene(which will handle the tires) and also open the Manager Window so you can add vehicles.

- Click on „Add Vehicle", this will open a new window. Drag in the vehicle's rigidbody into the "Rigidbody" field, then the WheelColliders and their corresponding MeshRenderers. If you want the tires to actually change their size in the simulation make sure to set the „Flat Radius Multiplier" to lower than 1.0(you can also hit the „select" button, this will zoom to the selected wheel and you can drag the gizmo in the viewport to change the radius when the wheel is completely compressed). You can change the flat radius later. You can close this window now, go back to the Manager.

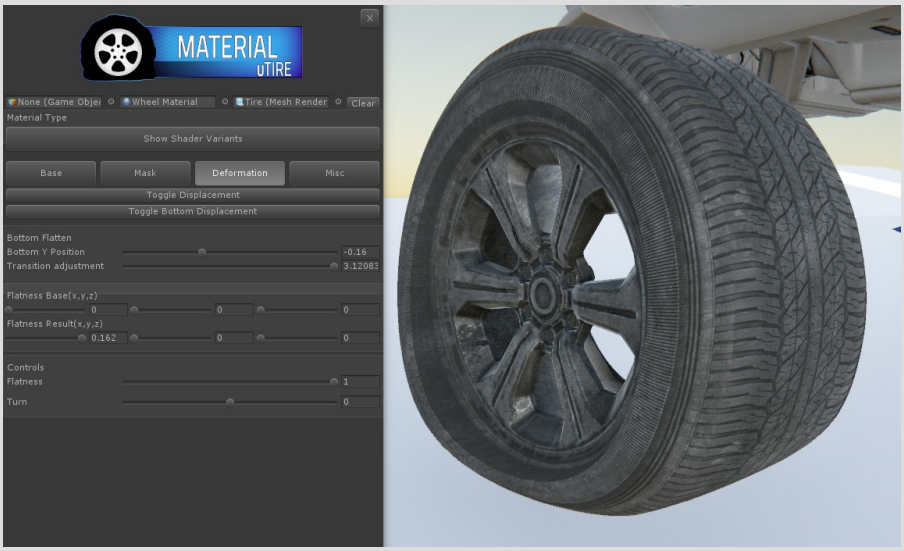
- Select the new vehicle from the list, then hit the"Edit Material" button. Since you aren't using a valid shader yet the material editor will let you chose to either change the material(it will override the original!) or to create and assign a new one(the new material will be placed in the same directory as the original, under the same name except it will get a „_TireDeformation" suffix). By default it starts with the Tessellation variant, but you can change this at any time(under the „Shader variants" tab). Drag in the albedo, normal, and Metallic\Smoothness\AO maps if you have them. Now to the fun part!

- Click on the „Mask" tab, then on the „Debug" button, this will activate the debug view for the masks. These should be pretty straight forward, just follow the hints(also note that when you change between debug views the material editor will highlight the sliders you should tweak).



- By default the system is trying to estimate how much should the tire mesh bulge in the sides, most of the time it works well(for realistic looking tires) but if you have very specifc needs(or a non regular mesh) you can manually change this by adjusting the “Flatness Result” sliders.

- At this point most likely the bottom part looks horrible, but we'll fix it in a second. First, set the „Bottom Y Position” value, this will determine the highest point where we want the tire to be flat, keep your eyes on the scene view and make sure it's lower than the rim(a few inches\centimeters around it will look fine in most cases). Now use the „Transition adjustment” slider to change (this determines the harnshess of the transition between the flattaned parts and the rest)



- And we are done, hit play and see what happens!

This tutorial is also available on YouTube, you can watch every step in real time from start to finish here

https://youtu.be/zebrQ_Qogxo

Everything Explained Step By Step

The Manager

This is the heart of the system. It will iterate through the registered vehicles(and their wheels), calculate the compression and direction and pass it to the shader. This process is automatic, other than registering a vehicle you don't have to do anything – which can be done at runtime or at design time as well. In the manager's inspector you only have two options, you can toggle separately the visual displacement and the WheelCollider update – if you have a fine tuned vehicle and you want it to stay **exactly** like it is then there you go. The visual displacement is just an eyecandy, it won't affect physics or the environment in any way. After the initial setup there is no GC allocation during runtime.

The Shader

At its core the system is based around a very simple idea, interpolate between two states, similarly to blendshapes – but there is a few, very important difference. First of all, everything is set in the shader itself, the data is already on the GPU, there is no CPU overhead during the displacement. Almost everything is calculated in the vertex program(for those who are unfamiliar with shaders,the general rule of thumb is if you can do it in the vertex shader do it there, it's by far the fastest option). What's cool about this system is after you make a few masks(in the shader, you don't have to use a vertex painter or Photoshop, with the custom editor it takes maybe a minute or two after you get used to the system to setup any tire). For the exact setup please read the Quick Start section of the manual.

Editor Windows

uTire Manager

Here you can register or remove vehicles, enable\disable the visual and wheelCollider updates.

Vehicle Setup

Here you can setup your vehicle(its RigidBody, WheelColliders and MeshRenderers), also you can change the wheelCollider's compressed size (this is a multiplier, a value around .7-.85 works best in most cases, but only consider this as a baseline – if your vehicle will be unstable make sure to raise this value though). Keep in mind this will change how your vehicle behaves.

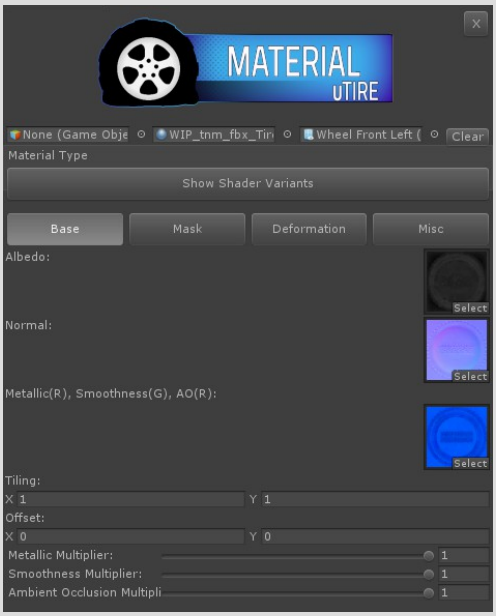
Integrations

If you own EVP, RCC or WheelController3D you can install the integrations(either by opening the menu from Tools->TSD->uTire Integrations Installer or from the uTire Manager window), the setup process will be almost completely done for you then. **Make sure to only install the integrations if you already imported RCC, EVP or WC3D or you will get compilation errors.**

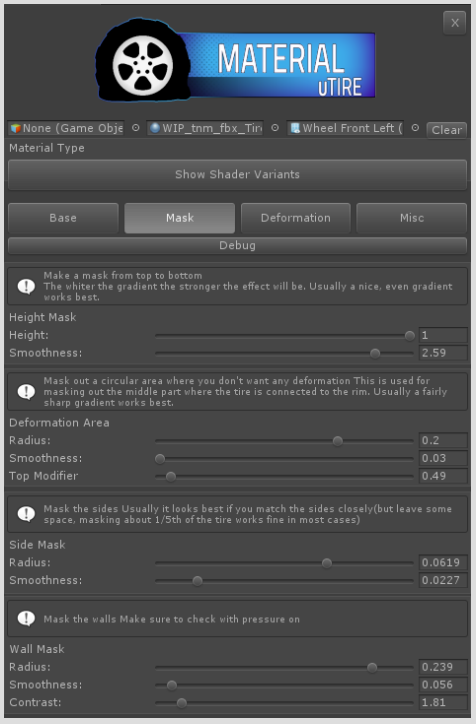
Material Editor

Here you can setup the material(duh), let's take a look at everything as you will spend most of your time here. If you open the material editor from the manager it will already select the material you will edit and its corresponding mesh, but if you open the window(tools->TSD->uTire Material Editor) you will have to provide a mesh and a material(or you can just drop in a game object and it will load the first meshrenderer and material).

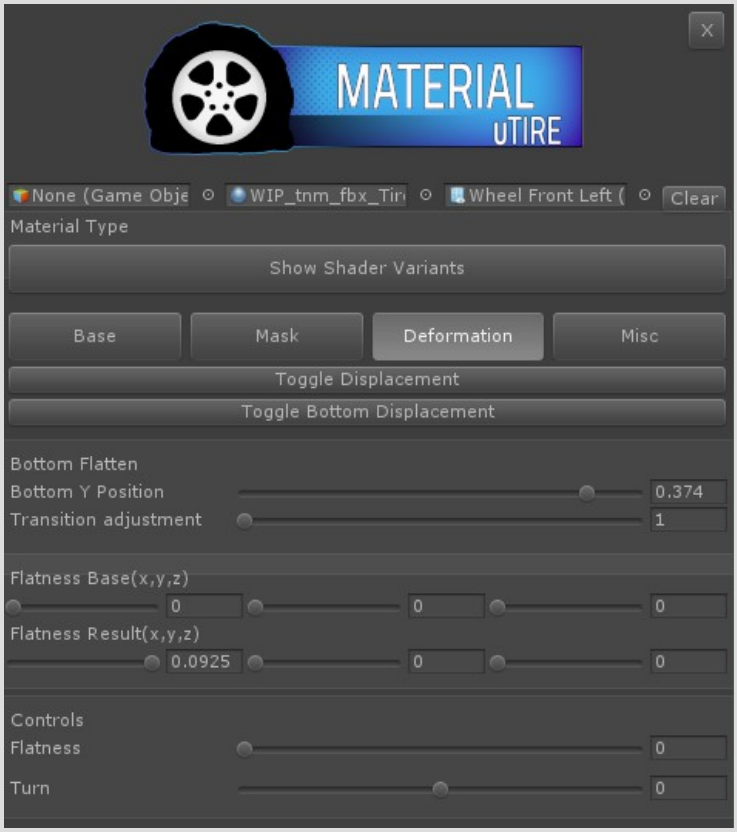
Base Tab. You can change the Albedo, Normal and Combined maps(Metallic is stored in the red channel, Smoothness is in the green channel and Ambient Occlusion is in the Red channel). If something seems to be off you can also adjust these values via basic sliders. Tiling and offset can be changed here as well.
If you use the Basic version of the shader most of these options will be disabled.



Mask Tab. You can set the deformation area here, this is by far the most important thing to get right. Make sure to take a look at the provided materials first, the setup is fairly intuitive but as it's not a standard approach, you might find it confusing for the first time.



Deformation Tab. Here you can set the deformation on the sides(the bulge effect) and how high should the bottom part go when the tire is compressed. As these are totally mesh dependent it isn't really a one-size-fits-all situtation, you will have to adjust it on a case by case basis. If you want to use the dynamic wheelCollider size feature you should make sure the bottom is about at the same height as the wheelCollider when it's compressed.
The Controls part in the bottom is set dynamically at runtime by the manager, but during setup you should check how would it look compressed and during turns as well.
It does



Scripting Examples

The runtime example scene contains a simple script to demonstrate how to register a vehicle during runtime. Open the scene at Assets\2SD\uTire_Examples\Runtime Example\Scene\uTire Runtime Example Scene.unity to take a look at it

The defect example scene contains a simple script to show how to modify the tire of a vehicle easily during runtime. Open the scene at Assets\2SD\uTire_Examples\Dynamic Defect Example\Scenes\uTire Dynamic Defect Example.unity to take a look at it.

Please keep in mind these scripts are for demonstration purposes only, to make it as simple(and easy to understand) as possible these aren't optimized.

The script reference can be accessed here(also there is a copy in a zip file in the Documentation folder)
http://2strokedesign.com/_uTire_Doxy/Documentation%20Doxy/html/index.html

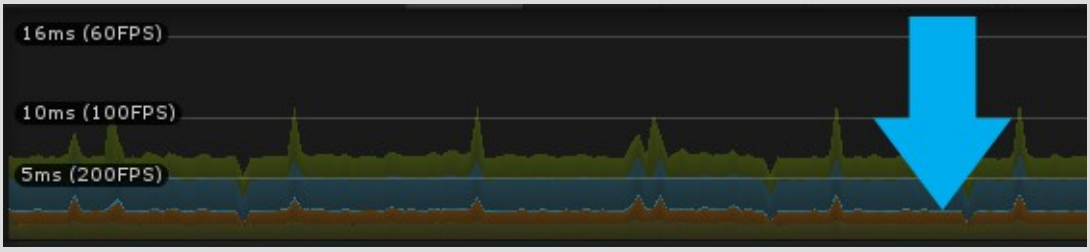
Limitations

- In order for Displacement to work properly the whole tire must be assigned to the same smoothing group. For existing models a quick and dirty solution is to make a copy of the original file, reimport it with normals set to „Calculate" with „Smoothing Angle" anything above 90. Then use this new mesh's tires on your vehicle instead(just keep in mind if your normal map depends on smoothing groups this will make them look wrong, and of course you'll have a completely unnecessary mesh in your Assets folder – but as a temporarily hack it works).

- The tire mesh have to be rotated properly(according to Unity's standard) in your modeling package – Z is forward, Y is up and X is sideways.

- For performance considerations currently the system assumes the terrain is always under the tire, in the world space Y axis. In most cases this is perfectly fine, but in extreme situations(like in huge loops) the deformation will look totally off.

Performance



It's fast! The exact number depends on how many tires are on the screen and how many are updating currently on the CPU(this part is extremely fast, the Manager with 8 cars(32 wheels) took 0.04-0.05ms(20-25.000 FPS) each frame on a slightly overclocked R7 1700.
Since the bulk of the shader is running in the vertex program it's about as fast as you can get, but obviously if you go heavy on features like Tessellation you will experience a performance hit. Compared to the standard shader it's virtually the same. The image above is an actual screenshot of the profiler, if you don't have performance issues now you can safely use this system.

FAQ

Q: Is this soft body physics?
A: No. Basically a simple interpolation between a predetermined „flat" state and a „full" state, based on the WheelCollider's compression, rotation and the RigidBody's speed.

Q: You mentioned tessellation, what if my target platform doesn't support it?
A: There are multiple shader variants included, tessellation, standard PBR, and a simplified one specifically for mobile(without support for PBR or normal map, it only takes a diffuse texture but the vertex deformation is exactly the same). Also you can easily customize(or extend) any of these with the Amplify Shader Editor.

Change Log

v1.0.5
2018.04.13
+Wheel Controller 3D integration

*Fixed null reference exception when changing the deflated radius via slider
*Fixed null reference expection when focusing on wheel mesh
*Fixed 'Simple' shader variant(now works on mobile as intented)
*Fixed material setup's slide limits(now it will adjust itself properly according to the maximum bulge value)
*Fixed included materials
*Changed the folder structure of the included meshes/textures
*Changed the preview of the deflated radius(now it's drawn at the mesh instead of the WheelCollider)

-Removed unnecessary files

Note
The way uTire handles the wheel changed completely in order to be a lot more flexible(it isn't tied to the built-in WheelCollider anymore), but because of this you'll have to setup your vehicles again(just the connections and the deflated radius, the materials will behave the same)

v1.0
Initial release, 2018.03