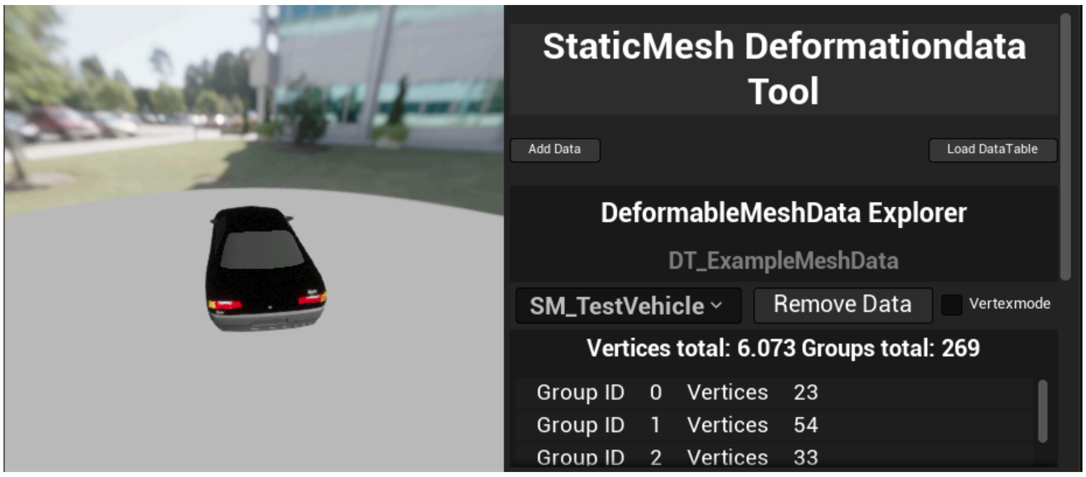
Debugging Delight Studios
2024

Easy Deformation Component Documentation V1.3

Join our 📌[Discord](#) for quick support

StaticMesh Deformationdata Tool



Vertices total: 6.073 Groups total: 269

```
Group ID   0   Vertices   23
Group ID   1   Vertices   54
Group ID   2   Vertices   33
```

Structure

New data can be created and existing data can be loaded using the top two buttons "Add Data" and "Load Datatable". In the "DeformableMeshData Explorer" individual meshes can be selected, which can then be seen in the viewport on the left.

Functionality

The "StaticMesh Deformationdata Tool" is the fundamental building block of the plugin. This is where the data creation process for the "Deformation Component" starts. The tool can read static meshes and prepare them for the component. During preparation, all vertices and their associated positions are packed into a struct. Depending on the configuration, the vertices are also divided into groupings that have a positive effect on performance.
This data is then saved in a datatable and then loaded and used by the deformable mesh component in the respective actor.

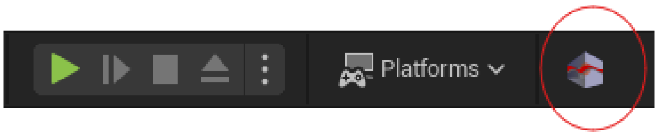
You have to pay attention to this!

Important! The ***tool*** only supports static meshes! Only a static mesh can be deformed by the *deformation component*, but your vehicle itself can be a skeletal or static mesh based vehicle. What you also have to consider: your vehicle mesh must be separate from the wheels! If you have a Skeletal Mesh based vehicle, please note that you have to create a Static Mesh for the tool, which you can find out how to do here!

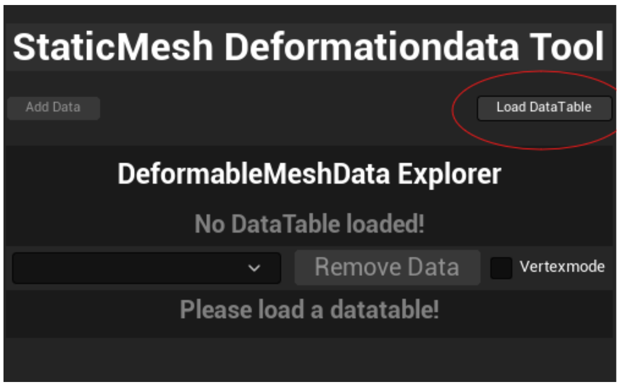
First steps

1. **Install and activate our plugin. [Howto?]**

2. **Open Static Mesh Deformation Data Tool**



3. **Load a compatible datatable**


StaticMesh Deformationdata Tool

Add Data

Load DataTable

DeformableMeshData Explorer

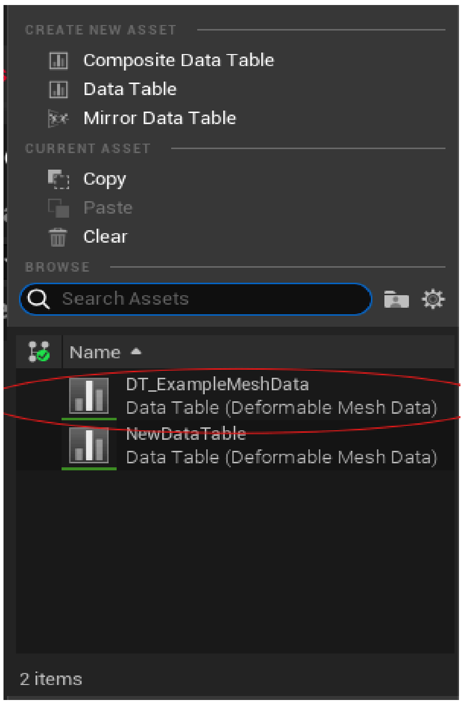
No DataTable loaded!

Remove Data

Vertexmode

Please load a datatable!

If a compatible datatable exists, it will be suggested to you in the list:


CREATE NEW ASSET

Composite Data Table

Data Table

Mirror Data Table

CURRENT ASSET

Copy

Paste

Clear

BROWSE

Search Assets

Name

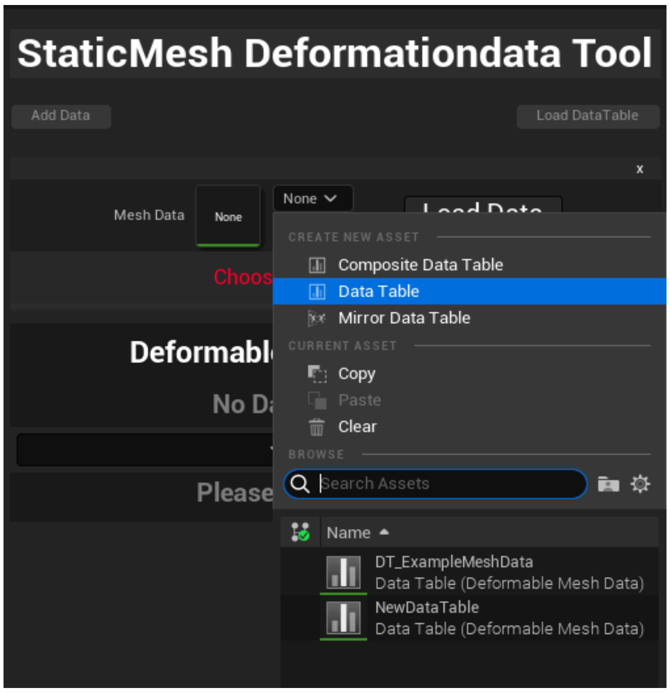
DT_ExampleMeshData
Data Table (Deformable Mesh Data)

NewDataTable
Data Table (Deformable Mesh Data)

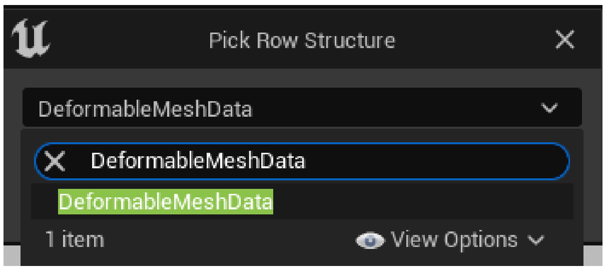
2 items

<u>If the list is empty it means that there is no compatible datatable and you still need to create it.</u>

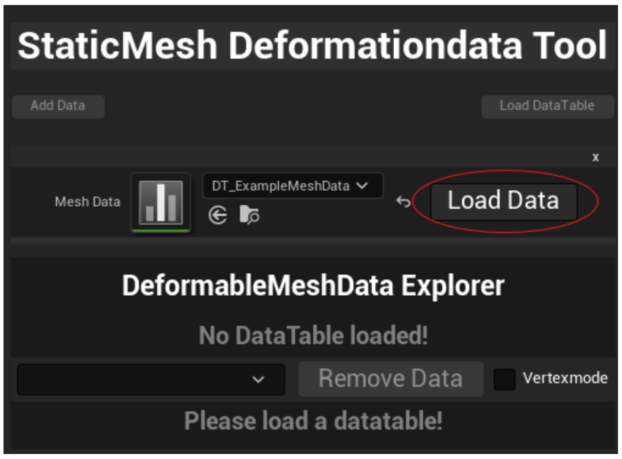
4. **(Optional) Create a compatible datatable**
   The component requires a special structure of the data table so that the data for the component is arranged in a readable manner.



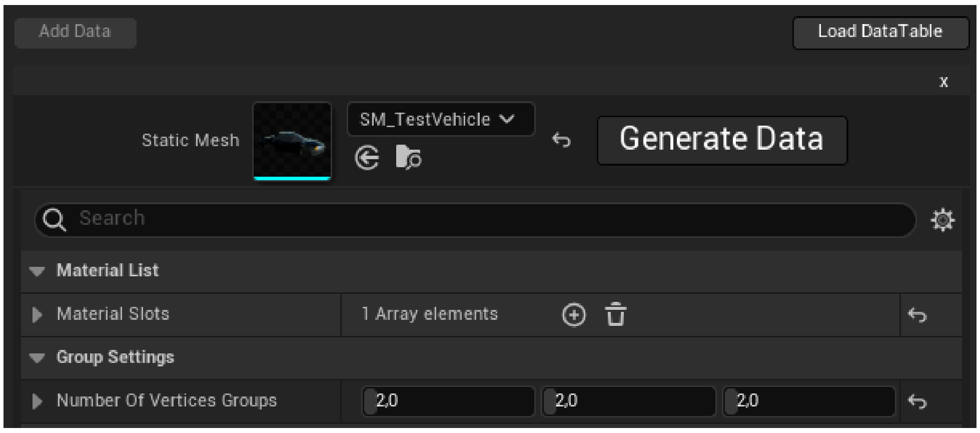
   To do this, press ”Data Table” under “Create new asset” and save it as you wish. Then select “**Deformable Mesh Data**" as “Data Row Structure”:



5. **Select your datatable and press “Load Data”**



6. **Create DefromableMeshData using the “Add Data” (top left).**

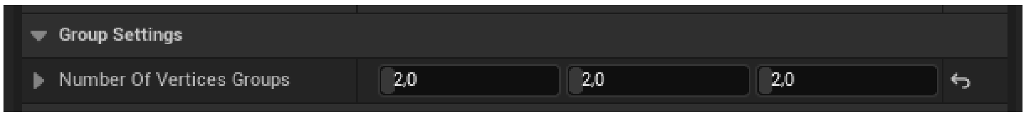


Static Mesh: SM_TestVehicle

Material List
- Material Slots: 1 Array elements

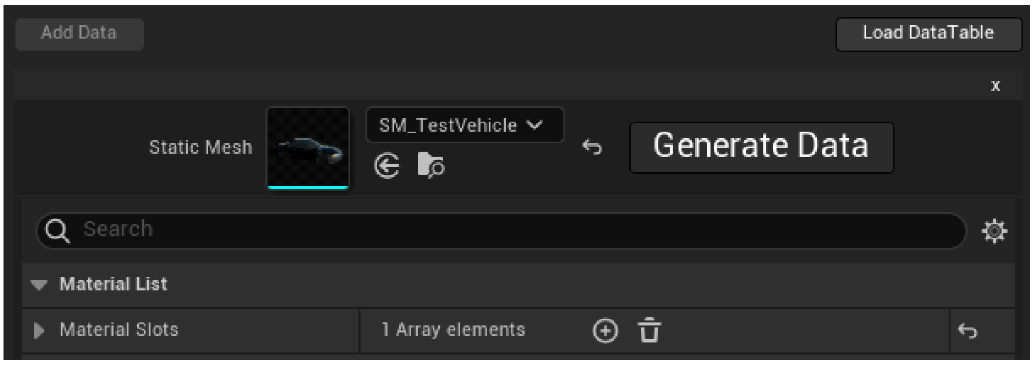
Group Settings
- Number Of Vertices Groups: 2,0 | 2,0 | 2,0

To do this, select your desired static mesh. Under ‘Number of Vertices Groups’ you can determine the number of groups of vertices on the X, Y and Z axes (see image). It makes sense to distribute the groups unevenly if the mesh is not square. The principle applies that the more detailed the model area, the more sense it makes to generate more groups on this axis. For example, if the mesh is very high (in the sense of height), it makes sense to create more groups on the Z axis than on the other axes. The more groups there are, the smaller the number of individual vertices in each group that need to be captured at once on impact.



Group Settings
- Number Of Vertices Groups: 2,0 | 2,0 | 2,0

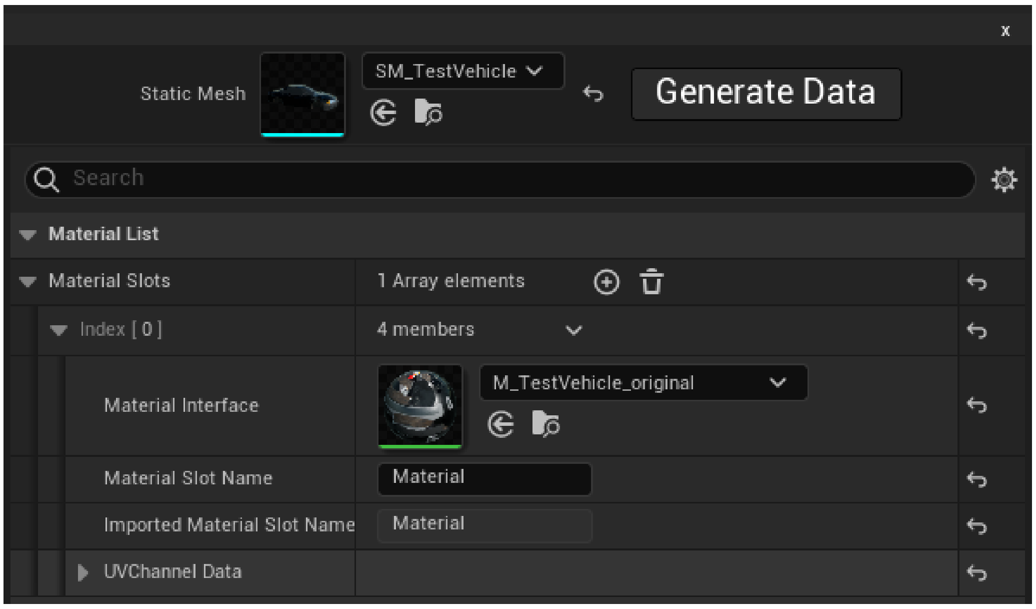
7. **(Optional) Select material**
   In this step, the materials of the static mesh can be changed. The source static mesh will be changed. Materials that support at least deformation should be used for this (more on this here). By default, all materials of the static mesh are already selected under "Material Slots".
   Please note: In some cases, the materials of poor static meshes that do not have reasonable UVs, for example, may not be displayed correctly in the component.



Static Mesh: SM_TestVehicle

Material List

Material Slots: 1 Array elements



Static Mesh: SM_TestVehicle

Material List

Material Slots: 1 Array elements

Index [0]: 4 members

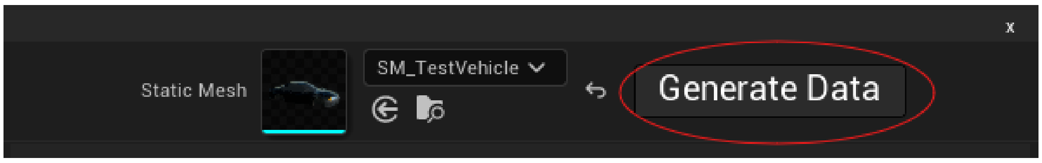
Material Interface: M_TestVehicle_original

Material Slot Name: Material

Imported Material Slot Name: Material

UVChannel Data

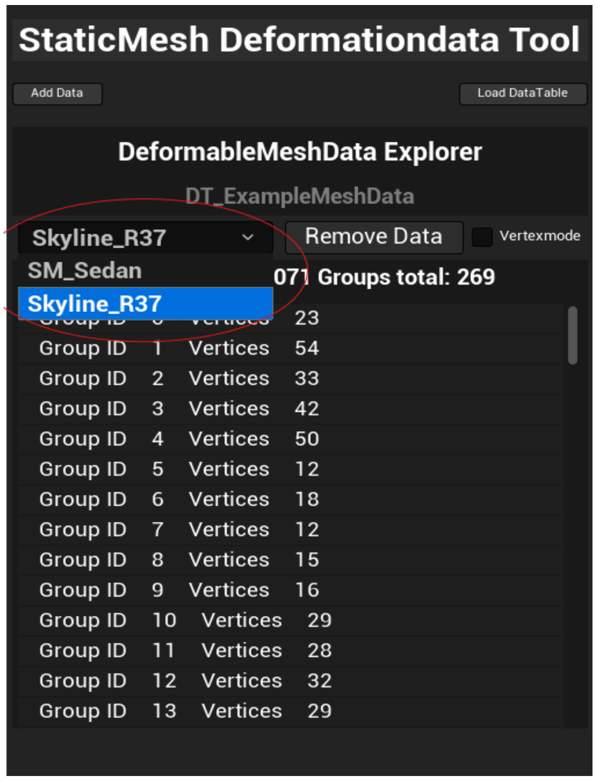
8. **Generate mesh data**
   Last press on “Generate Data” to transfer the static mesh and all the settings made into the data table.



DeformableMeshData Explorer

Open data

The “DeformableMeshData Explorer” gives you an insight into your created datatables and prepared meshes. First load your datatable using the function “*Load DataTable*” into the tool and then select the desired mesh in the selection box to see its groupings and the mesh itself in the viewport:



StaticMesh Deformationdata Tool

Add Data | Load DataTable

DeformableMeshData Explorer

DT_ExampleMeshData

Skyline_R37 | Remove Data | Vertexmode

SM_Sedan

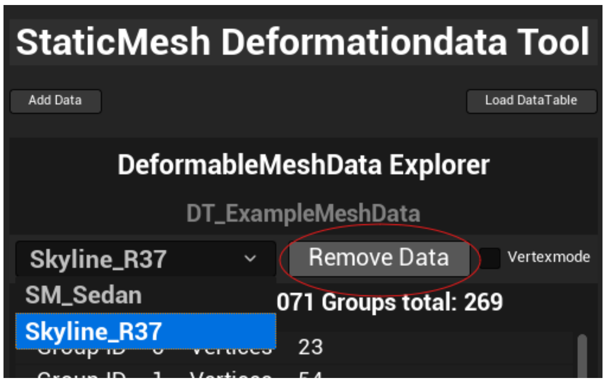
Skyline_R37

071 Groups total: 269

```
Group ID   0   Vertices   23
Group ID   1   Vertices   54
Group ID   2   Vertices   33
Group ID   3   Vertices   42
Group ID   4   Vertices   50
Group ID   5   Vertices   12
Group ID   6   Vertices   18
Group ID   7   Vertices   12
Group ID   8   Vertices   15
Group ID   9   Vertices   16
Group ID   10   Vertices   29
Group ID   11   Vertices   28
Group ID   12   Vertices   32
Group ID   13   Vertices   29
```

Delete datatable entries

To delete data, select the corresponding mesh in the selection box (in the example image *Skyline_R37*) and then press the “Remove Data” button.



StaticMesh Deformationdata Tool

Add Data | Load DataTable

DeformableMeshData Explorer

DT_ExampleMeshData

Skyline_R37 | Remove Data | Vertexmode

SM_Sedan

Skyline_R37

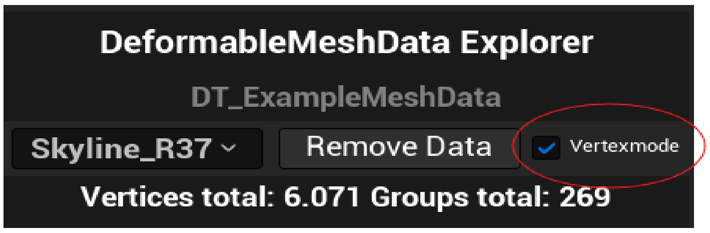
071 Groups total: 269

```
Group ID   0   Vertices   23
```

Navigation in the viewport

The mesh can be inspected using the viewport. Navigation takes place with the right mouse button pressed and W/A/S/D. The speed can be set faster or slower using the mouse wheel.

Vertexmode

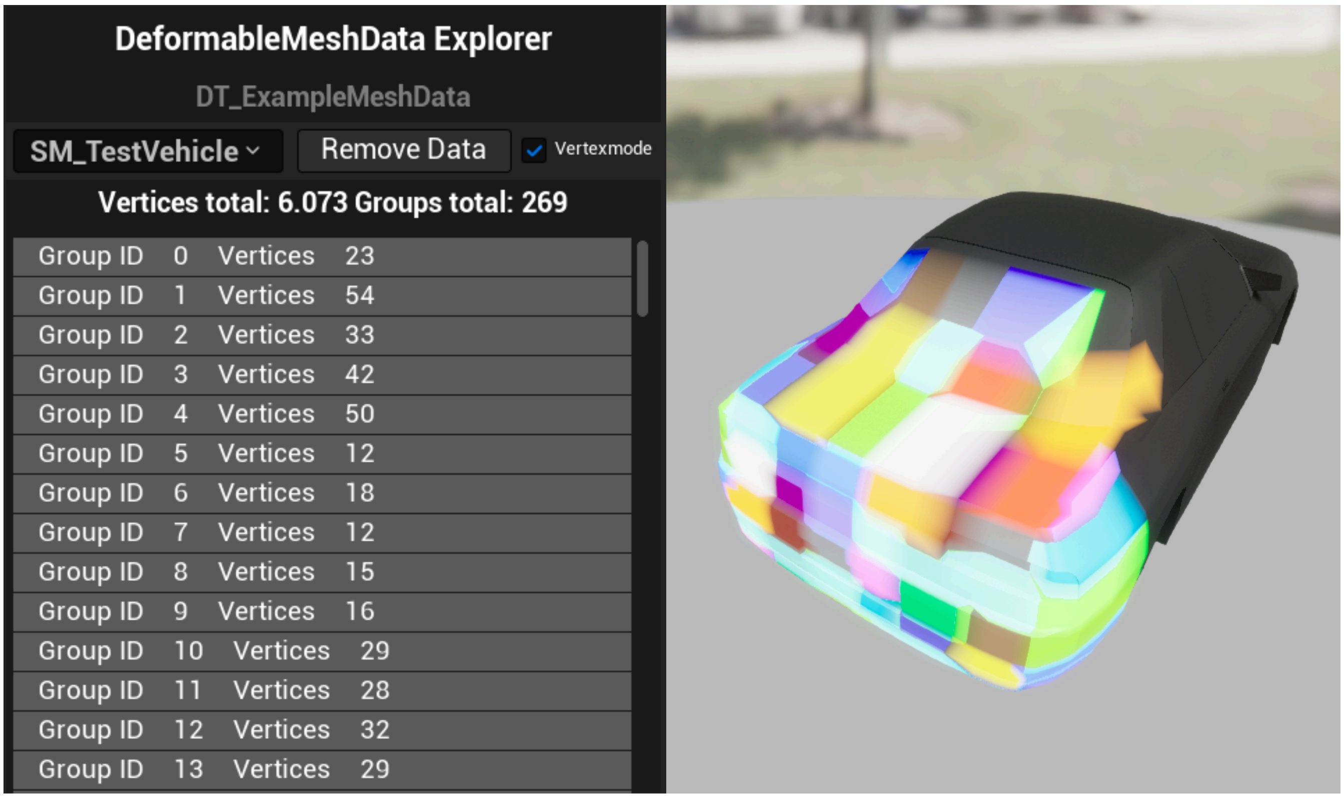


DeformableMeshData Explorer

DT_ExampleMeshData

Vertices total: 6.071 Groups total: 269

The 'Vertex mode' visualises the groupings of the mesh in colour in the viewport. To do this, the mode must be activated and then groups that are to be displayed must be selected from the list below:



DeformableMeshData Explorer

DT_ExampleMeshData

Vertices total: 6.073 Groups total: 269

Group ID	0	Vertices	23
Group ID	1	Vertices	54
Group ID	2	Vertices	33
Group ID	3	Vertices	42
Group ID	4	Vertices	50
Group ID	5	Vertices	12
Group ID	6	Vertices	18
Group ID	7	Vertices	12
Group ID	8	Vertices	15
Group ID	9	Vertices	16
Group ID	10	Vertices	29
Group ID	11	Vertices	28
Group ID	12	Vertices	32
Group ID	13	Vertices	29

Deformation Component

BP_DeformableMeshComp

Functionality

The *Deformable Mesh Comp* is the counterpart to the *Static Mesh Deformation Data Tool*. This is where the magic happens. Here the collected data from the datatable that we previously created in the tool is processed. The component simply works according to the plug and play system: **You simply place it below the vehicle mesh, give the component some information and the component regulates the rest independently.**

Details on how the component works

The component does this internally in the background

1. Loads the previously created datatable into memory using Async Load. A procedural mesh is then created, which we will use later for the deformation. This procedural mesh is a child of the vehicle mesh.

2. The procedural mesh itself is created from the previously prepared data of the tool from the datatable.

3. The component hides the vehicle mesh (*parent component*) and switches on its “Simulate Hit Events” (important only for vehicle mesh!). The reason for this is that we now bring our procedural mesh to the foreground because that is where we will apply the deformation upon impact. **The original vehicle mesh only serves us as a hit detection mesh**. Important: The whole thing works just as well with a skeletal mesh as with a static mesh as the vehicle mesh, which is not important for the *component*. Next, the component binds the “On Component Hit” event to the vehicle mesh with our deformation logic.

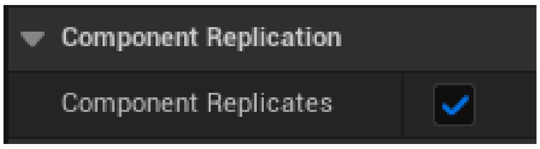
This is roughly summarized in deformation logic

During an impact, an area is formed at the location of the impact, the size of which varies Impact Radius is dependent.
In this area we look at which vertices are affected by the impact. This would not be easily possible by going through each individual vertical in a for each loop, looking at its position and checking whether this is in the formed area.
This for each loop would then be so extremely complex for the CPU while the game is running (i.e. the more detailed the mesh, the longer the loop) that it could lead to FPS drops or even application crashes.
That's why we decided to pre-sort and group all vertices with the tool before runtime so that in the event of an impact we only have to check whether a group of vertices is affected and then only check this small, relevant group of vertices in the loop whether it is affected. If relevant, these vertices are shifted in position from the opposite direction of impact and updated in the procedural mesh.

In order to make the whole thing even more realistic visually, we use this affected group in another procedure in which we change the vertex color to which a material instance can then react and then color scratches (via a mask) and dents (via the normal input) in this area.

Multiplayer / Network Replication (V1.1)

The component can be used in multiplayer. All you need to do is switch on the "Component Replicates" setting of the deformation component, which is the case by default. **If the component is used in single player, this setting can and should be deactivated in order to save performance.**


Component Replication
Component Replicates ✓

How it works

When network replication is active, all hits are recognised by the server and forwarded to all *relevant*[1] clients. Each client calculates the deformation independently based on the data received from the server. In general, the hits are always recognised by the server, with the exception of when the actor is a pawn and this pawn is controlled by a player. In this case, the hits are recognised by the controlling client and sent to the server.
Each hit is also stored in the *hit history*. This hit history is used to be able to restore the current deformation at any time. This is particularly important as a player who joins later should see the same deformation as all other players. The same concept applies to actors that only become relevant for a client later. In both cases, the hit history is replayed client side so that all players subsequently see the same deformation.

Limitations

We are currently aware of two limitations in the way our network replication works. Firstly, there is always a certain delay in deformation, as all data must first be sent to the server and then back to the client. This can be optimized in future using client-side prediction.
Secondly, the *hit history* only supports a maximum of ~300 entries. The reason for this is that Unreal Networking has a fixed maximum packet size[2] and this is currently reached after ~300 hits. Once the maximum number of hits has been reached, the oldest hits are always deleted from the hit history for all subsequent hits and are therefore lost.

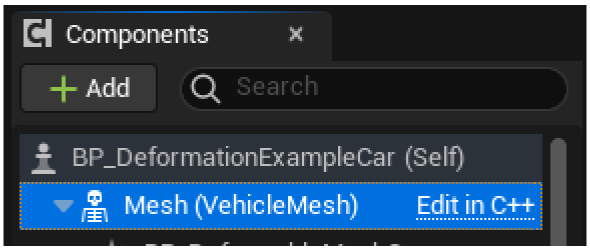
[1] Network Relevancy
[2] Also called "maximum bunch size"

Component setup

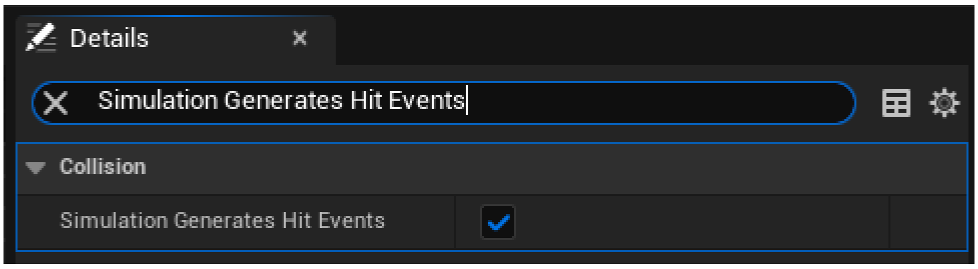
The component can be combined with many other plugins, for example the "*Chaos Vehicle*" plugin from Epic Games or the "*Advanced Vehicle System*" plugin by Overtorque Creations. For simplicity's sake, let's start with *Chaos Vehicle*.

Chaos Vehicle and AVS

1. First create a normal chaos vehicle, for example with this Instruction or our instructions.
2. The root component ("VehicleMesh") must then be adjusted



   a. Navigate in the Details panel under "*Collision*" and switch "*Simulation Generates Hit Events*" **and** "*Simulate Physics*" on.

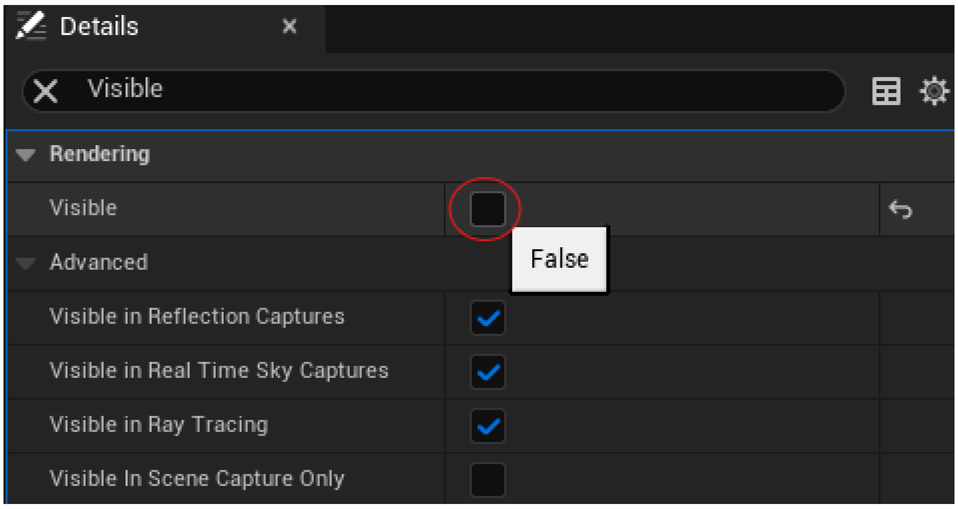


Simulation Generates Hit Events

Collision

Simulation Generates Hit Events ✓

   b. Now search for "*Visible*" and put the box on *False*.



Visible

Rendering

Visible ☐ False

Advanced

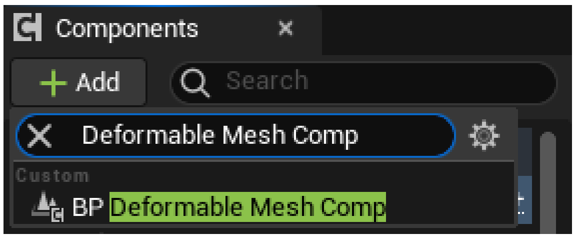
Visible in Reflection Captures ✓

Visible in Real Time Sky Captures ✓

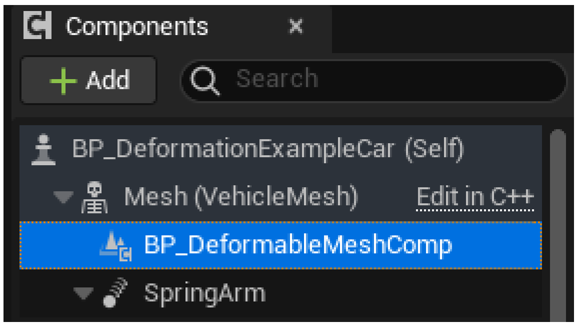
Visible in Ray Tracing ✓

Visible In Scene Capture Only ☐

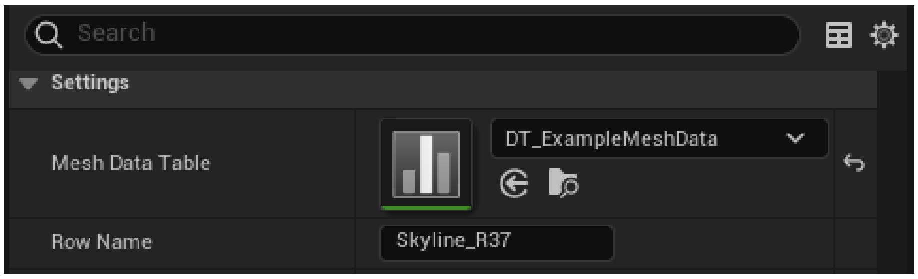
3. Press that "*Add*" Button, search for **"Deformable Mesh Comp"** and add the component below the root component:



Important: The component must be a child of the static/skeletal mesh that is to deform. In our case the "VehicleMesh":

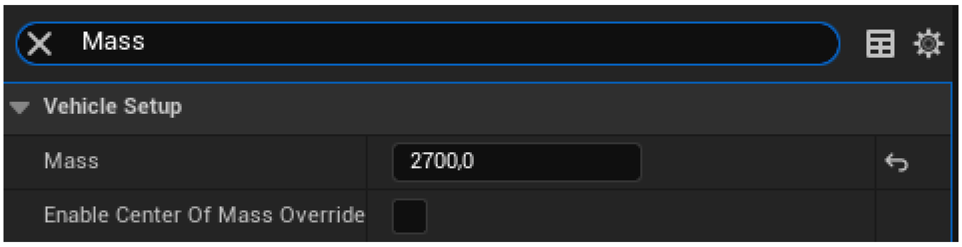


4. Select the just created "BP_DeformableMeshComp" and open the Details panel. Find the "Settings" category, select your created DataTable as "Mesh Data Table" and also enter the name of the mesh in "Row Name", as shown in the image.



Settings
Mesh Data Table: DT_ExampleMeshData
Row Name: Skyline_R37

5. Next, you define the weight of your chaos vehicle. It is strongly advised to choose a realistic weight, as the damage calculation depends on the weight. Notice: In case you don't use chaos vehicles and skeleton meshes, the *Mass* of the parent static mesh component is used.



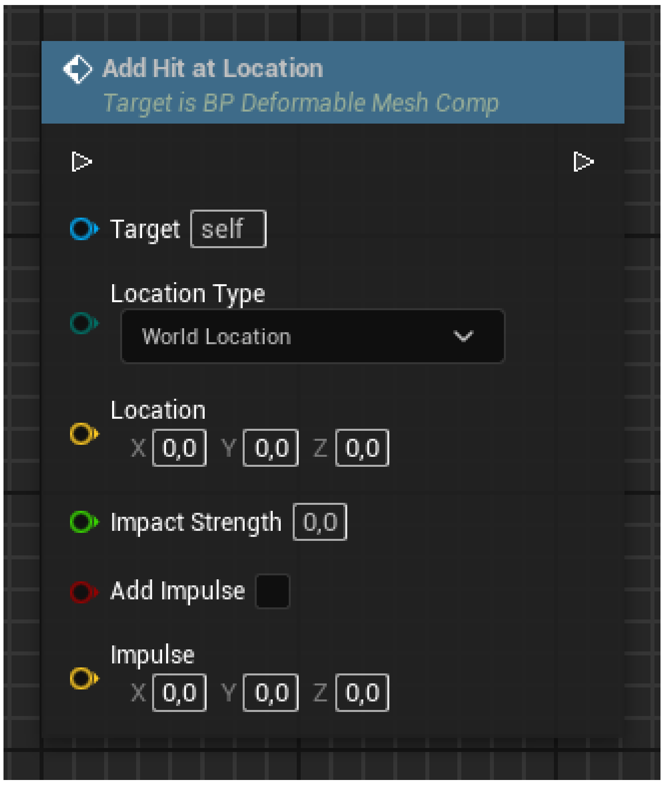
Mass
Vehicle Setup
Mass: 2700,0
Enable Center Of Mass Override

Non-Vehicle Actor

The deformation component does not necessarily have to be used in combination with a vehicle. It can also be used with normal actors. These actors do not even necessarily have to simulate physics. For non-physics-simulating actors, however, it should be noted that a mass must be specified as a mass override. The calculation of the deformation requires a mass! The setup procedure differs only slightly from the procedure for Chaos Vehicle. The first step is omitted and the mass (step 6) does not have to be specified in the vehicle, but if at all in the component (as a "mass override").

Component functions

Add Hit at Location *(V1.1)*


Add Hit at Location
Target is BP Deformable Mesh Comp
Target self
Location Type
World Location
Location
X 0,0 Y 0,0 Z 0,0
Impact Strength 0,0
Add Impulse
Impulse
X 0,0 Y 0,0 Z 0,0

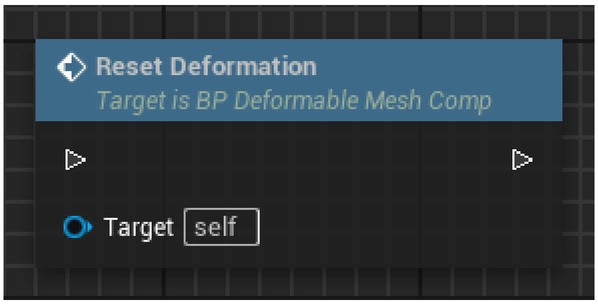
This function can be used to perform a hit (+ deformation) at a given position. The position can be a position in the world or relative to the actor.

The *impact strength* is in *m/s²* and can be determined using the Debug impulse setting.

This function can also be used to automatically add an impulse at the specified position. It internally uses the default *AddImpulseAtLocation* node.

🌐This function is replicated and can be called by the controlling client or the server (authority).

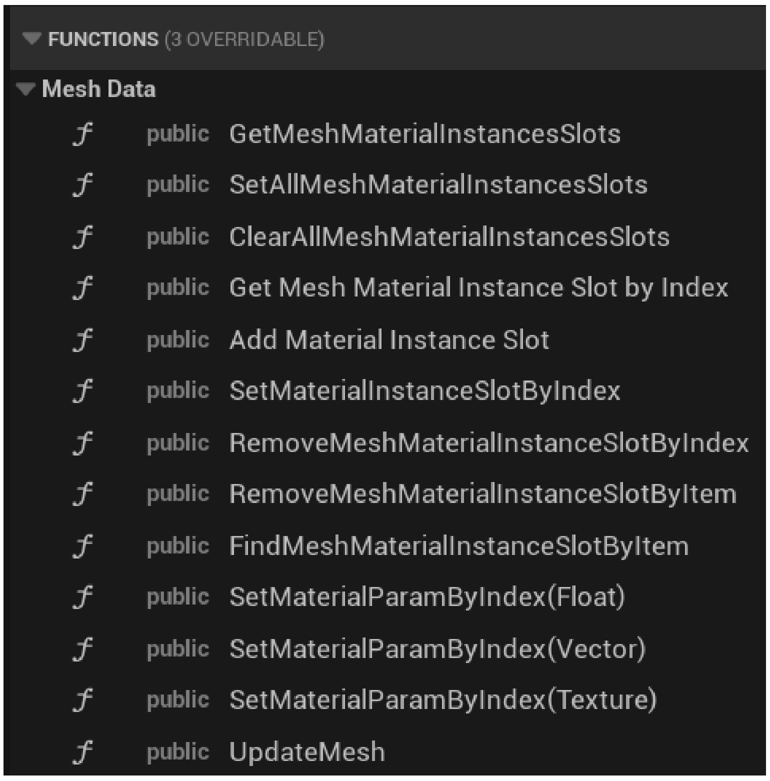
Reset Deformation


Reset Deformation
Target is BP Deformable Mesh Comp
Target self

This function can be used to reset the deformation.

🌐This function is replicated, but may only be called by the server (authority).

Other useful functions (Materials *V1.3*)



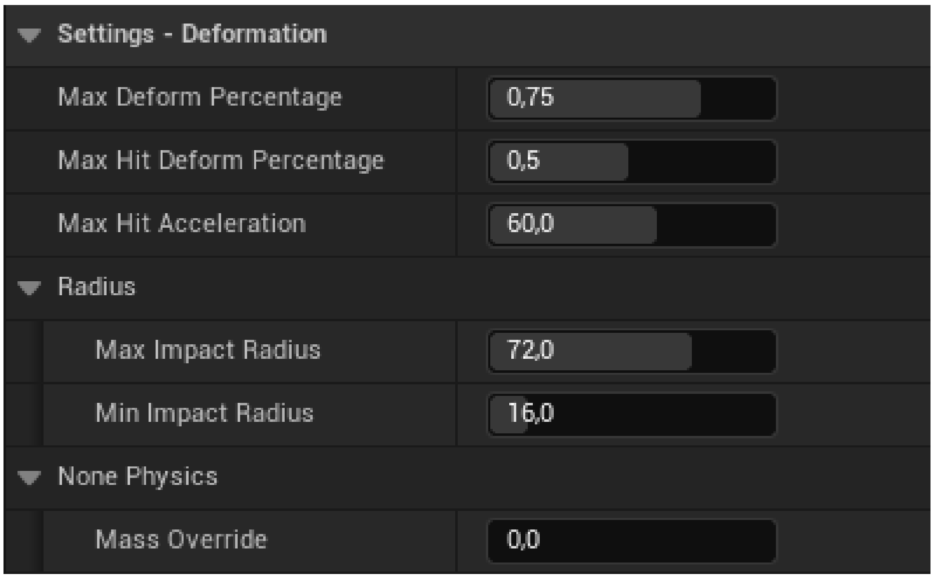
```
ƒ  public  GetMeshMaterialInstancesSlots
ƒ  public  SetAllMeshMaterialInstancesSlots
ƒ  public  ClearAllMeshMaterialInstancesSlots
ƒ  public  Get Mesh Material Instance Slot by Index
ƒ  public  Add Material Instance Slot
ƒ  public  SetMaterialInstanceSlotByIndex
ƒ  public  RemoveMeshMaterialInstanceSlotByIndex
ƒ  public  RemoveMeshMaterialInstanceSlotByItem
ƒ  public  FindMeshMaterialInstanceSlotByItem
ƒ  public  SetMaterialParamByIndex(Float)
ƒ  public  SetMaterialParamByIndex(Vector)
ƒ  public  SetMaterialParamByIndex(Texture)
ƒ  public  UpdateMesh
```

These functions can be used to edit the materials of the default mesh.

After a change has been made, *UpdateMesh* should always be called to refresh the visible mesh.

Settings of the component

Deformation Settings



Settings - Deformation

Max Deform Percentage	0,75
Max Hit Deform Percentage	0,5
Max Hit Acceleration	60,0

Radius

Max Impact Radius	72,0
Min Impact Radius	16,0

None Physics

Mass Override	0,0

Max Deform Percentage

Maximum deformation in percent. The exact value depends on the distance of the vertex to the mesh origin, e.g. a vertex that is far out can achieve a much higher deformation than a vertex that is located directly on the origin.

Max Hit Deform Percentage

Maximum deformation in a single impact, e.g. 10%, means that at least 10 hits (10 x 10% = 100%) are required to achieve the maximum deformation.

Max Hit Acceleration

Acceleration (m/s^2) required to achieve "Max Hit Deformation Percentage". Can be measured with the debug setting "Debug Vehicle Velocity".

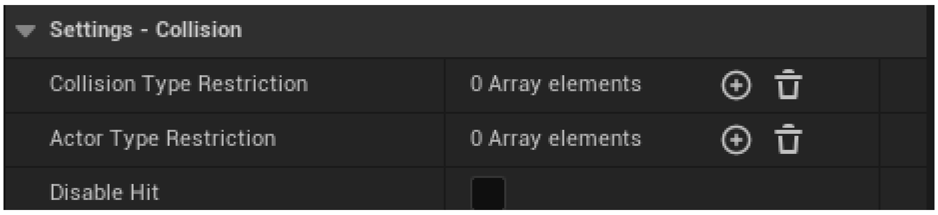
Radius Settings (Impact Radius)

The minimum/maximum size of the impact (in cm). It affects how many vertices can move in a single impact. The minimum size corresponds to a very low force impact and the maximum size corresponds to a very high force impact.

Mass Override *(V1.1)*

This value overrides the mass of the parent component. If specified, this mass is used to calculate the deformation. This setting is mandatory if the parent component does not simulate physics. See Non-Vehicle Actors.

Collision Settings



Settings - Collision

Collision Type Restriction	0 Array elements	
Actor Type Restriction	0 Array elements	
Disable Hit	☐	

Collision Type Restriction

(Optional) Here you can add collision types with which the component should only simulate collision. By default there is no restriction.

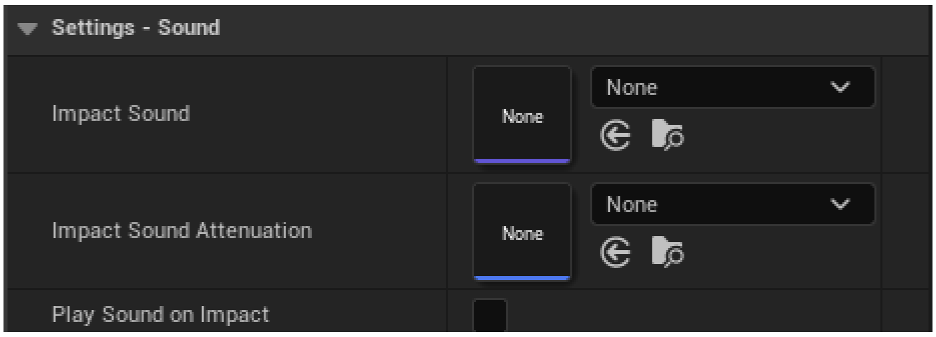
Actor Type Restriction

(Optional) Here you can add actor types with which the component should only simulate collision. By default there is no restriction.

Disable Hit

Here you can deactivate deformation.

Sound Settings



Settings - Sound

Impact Sound	None	None
Impact Sound Attenuation	None	None
Play Sound on Impact	☐	

Impact Sound

Here you can insert a sound file that will be played at the point of impact.

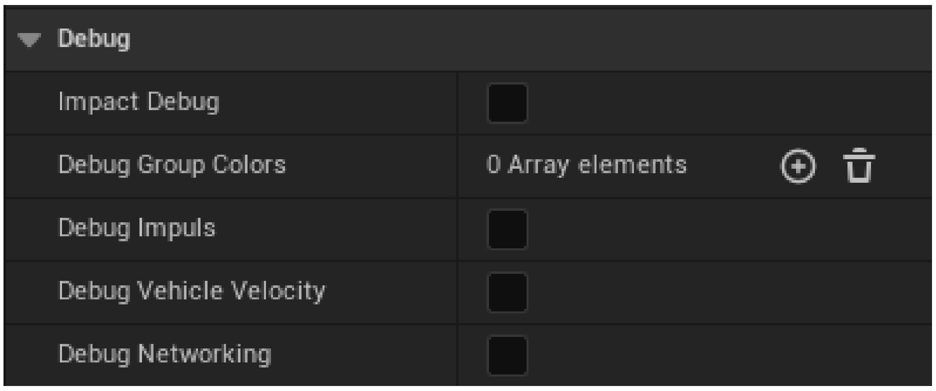
Impact Sound Attenuation

Insert a sound attenuation config file here that matches your impact sound.

Play Sound on Impact

Here you activate your selected impact sound upon impact
should be played.

Debug Settings



Debug

Impact Debug	☐
Debug Group Colors	0 Array elements
Debug Impuls	☐
Debug Vehicle Velocity	☐
Debug Networking	☐

Impact Debug

Here you can display the impact radius visually in-game for debugging purposes.

Debug Impulse

Shows impulse values that act on impact (kilonewtons and speed in *m/s²*). These two values are important for the impact calculation:
$Impact\ Strength\ [m/s^2]\ =\ Normal\ Impulse\ [N]\ /\ Mass\ [kg]$

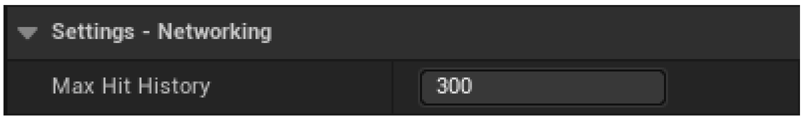
Debug Vehicle Velocity

Shows speed values in cm the second upon impact.

Debug Networking *(V1.1)*

This setting can be used to log the network relevance of an actor and the number of replicated hits that need to be replayed, for example, for a player who joins later.

Networking Settings *(V1.1)*



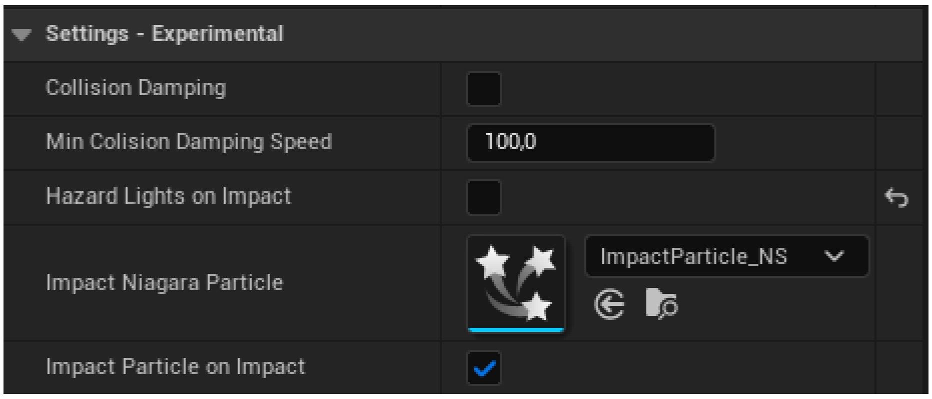
Settings - Networking

Max Hit History	300

Max Hit History *(V1.1)*

Maximum number of replicated hits that, for example, a player who joins later receives and replays. Limited by maximum Unreal package size, see Limitations.

Experimental Settings



Settings - Experimental	
Collision Damping	☐
Min Colision Damping Speed	100,0
Hazard Lights on Impact	☐
Impact Niagara Particle	ImpactParticle_NS
Impact Particle on Impact	✓

Collision Damping

Dampens the force that throws the vehicle back like a rubber ball after an impact.

Min Collision Damping Speed

Minimum speed at which collision damping should take effect.

Hazard Lights on Impact

Switches on the vehicle's hazard warning lights in the event of an impact (requires a vehicle material specially made for this purpose and serves only as an example of what is possible with light despite deformation).

Impact Niagara Particle

Insert a Niaga Particle System here, which should simulate particles and smoke upon impact.

Impact Particle on Impact

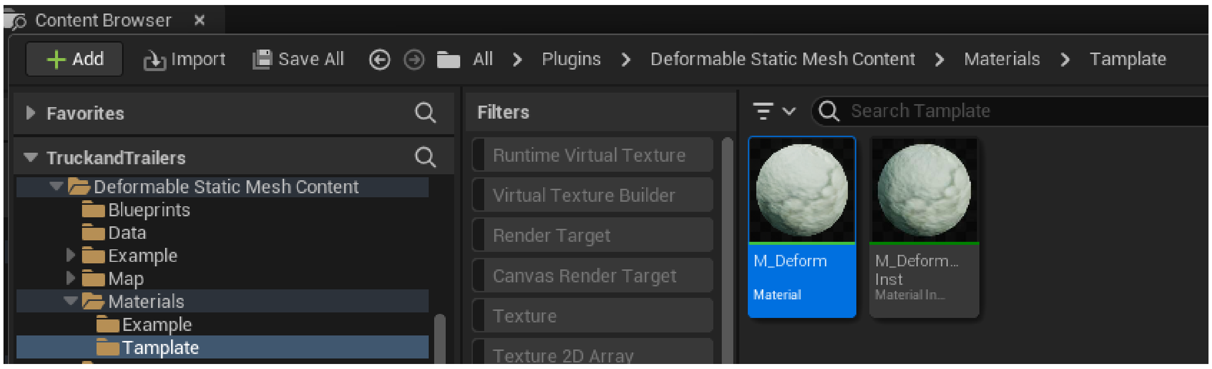
Turns on the Impact Niaga Particle System.

How to set up (Material)

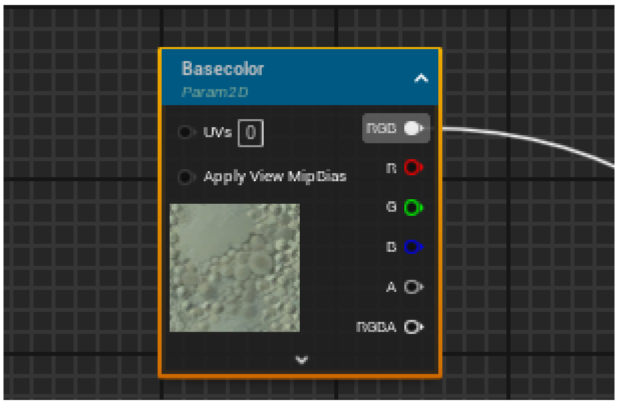
How to create a material for your vehicle/mesh

Also available as a video tutorial.

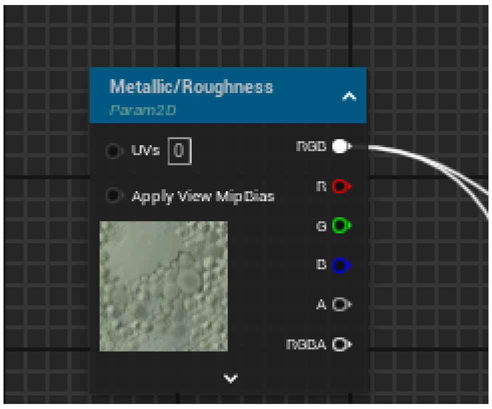
First open / create a copy of the material file *M_Deform* located in "*/Plugins/DeformableStaticMeshContent/Materials/Template*":



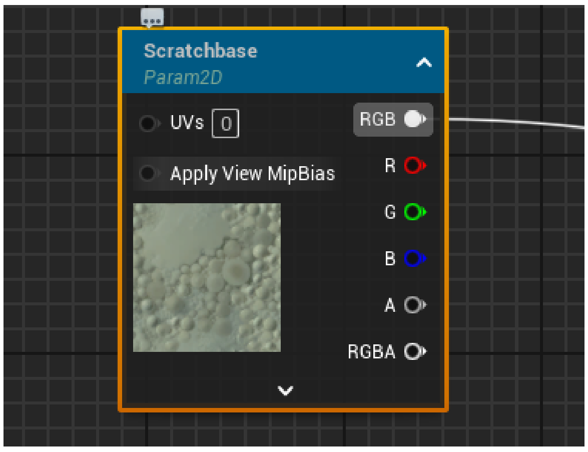
Now add your Base Color texture under *Basecolor*.



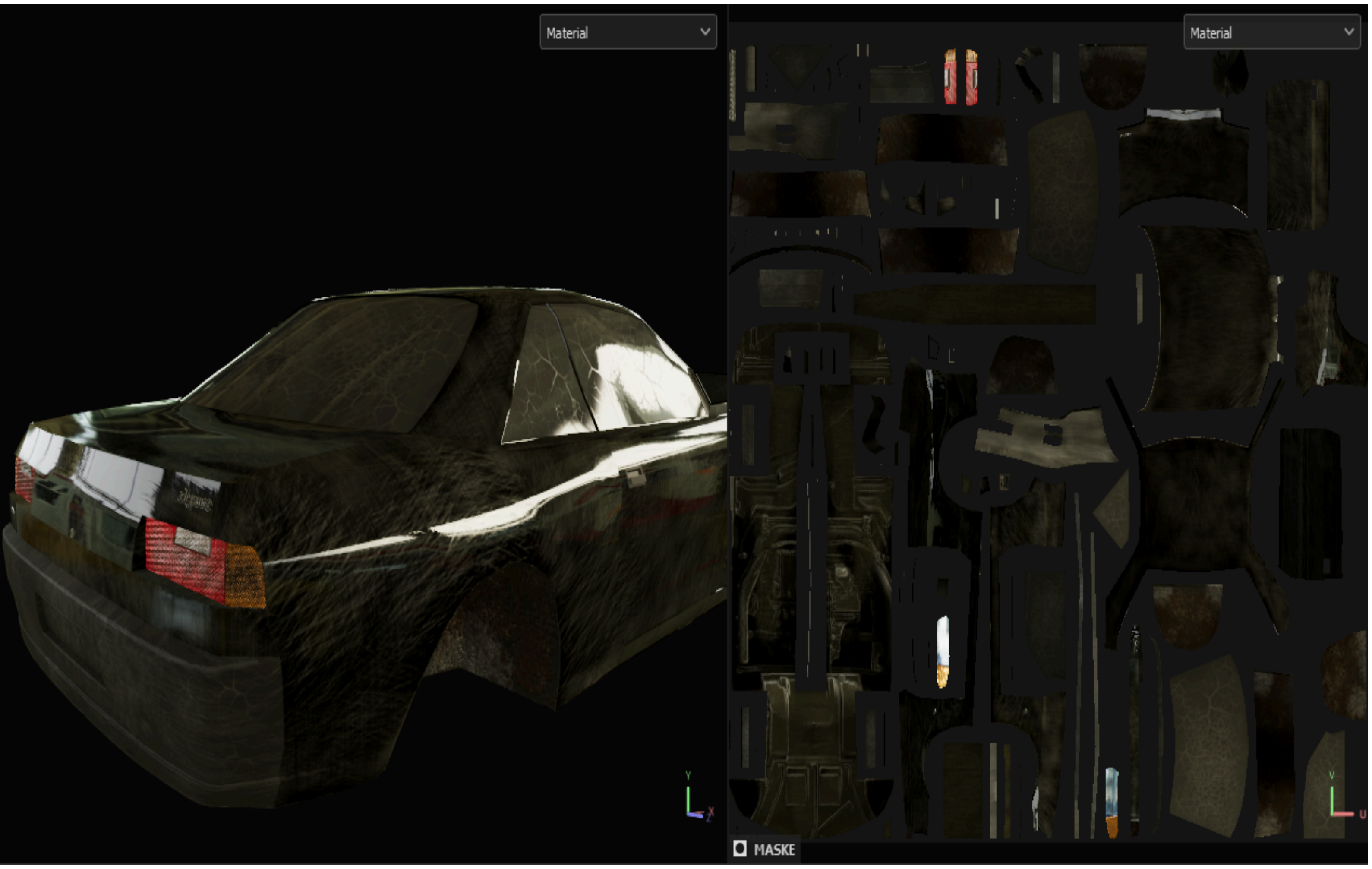
For *metallic/roughness*, your metallic roughness texture belongs in (use the correct color outputs configured in your texture and then connect them to the result node.



The *Scratchbase* Node now contains a base color texture into which you can add scratches (which you have to create with Substance Painter, for example).



<u>It looks like this, for example.</u>



Changelog

Version 1.0 (07-2024)

- Initial release
- Basic functionalities

Version 1.1 (08-2024)

- Network Replication
- Support for non-physics-simulating Actors and Non-Vehicle Actors
- Support for impulses through new AddHitToLocation function
- Various refactorings of the component
- Removed: "Vehicle Material" setting

Version 1.2 (09-2024)

- 12 new functions have been added to influence the materials from the Procedural Mesh, for example to control the vehicle lighting.
- New getter function has been added to expose the Procedural Mesh.
- A bug that only one material slot is taken into account (the last index) is also fixed.

Version 1.3 (10-2024)

- Demo map:
  - A bug has been fixed in the demo map where the tires constantly slipped into the ground
  - Collisions in the demonstration vehicle have been reworked
- C++ optimizations have been made. Among other things, all for loops processing large data sets have been parallelized and are now executed using a multithreading process, which applies dynamic chunking depending on system performance and the size of the data set.
- Multiple UVs are now supported, up to 4 per section of the static mesh.
- The damage model has been completely revised. Among other improvements, you can now define the damage falloff via a curve. In the curve, you can determine precisely where the center of the damage is and how it should be distributed. You can also apply noise to make the damage appear more dynamic.
- An "*On Impact*" event has been added to allow binding, for example, to damage display widgets or anything that needs to be updated after an impact.
- Collision damping has been improved and now functions more effectively.
- And much more.