Assetto Corsa Pipeline for Community Modders R2.0

BEFORE YOU START:

This is an updated version of the original **car-pipeline-1.03** document that was released alongside the game's original release. The car production pipeline has gone through massive changes since then, cars included in the original game had vastly different requirements than current releases, which is why an update has been due.

This document contains information that matches the quality requirements for official KUNOS releases since the Japanese Pack DLC, through the Porsche Packs and the most recent Ready To Race DLC packs. It is safe to say that the current requirements are unlikely to change in the remaining lifecycle of Assetto Corsa.

However, some new functions (especially digital display scripts etc.), are likely to be added, which is why we recommend you visit the official support forum to be up to date on developments in this respect.

Note that due to the nature of the development cycle, we have continued to add new functions and features into the simulator, which is why this document is not comprehensive. However, each section will include a link to the most relevant community threads on the official support forum, where community artists can find, and also ask for, support to guide them through their projects. Because the core engine has not changed, some parts of the document, such as animations and general model management, remain unchanged.

For compatibility purposes, you can still find the contents of the old sdk in an archive within the new SDK folder.

Note that most pieces of advice in this document are intended for our internal professional artists. Naturally, the creation of community-made content is not so strict and it's down to your preference and expectations how much you wish to stick to it.

Also note that Kunos staff artists do not work with Blender, so guidelines and rules do not always transfer correctly to that software and thus we have no experience in helping you find the right methods for the creation of community content. Again, we can only urge you to visit the modding section of the official support forum to find information on Blender.

IMPORTANT: In addition to this document, the sdk/dev folder contains a number of useful folders with examples and guides for scene hierarchy, various animations, and a driver rig to be used in 3ds Max.

1. REQUIREMENTS

B. IMPORT/EXPORT SETTINGS

I. We export all files in the format supported by the **FBX version up to the 2014/2015 plugin** for XSI and 3dsMAX. Avoid using attributes that are not supported by this export format (such as physics constraint or mesh smooth operators etc.) The 2016 plugin is unsupported!

II. The FBX data used by the AC engine are following:

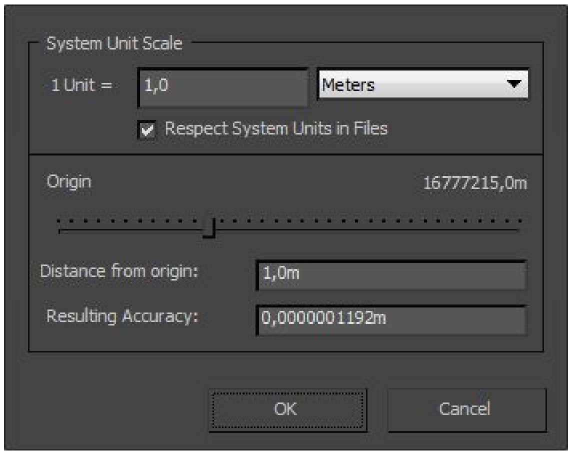
- **Polygon mesh**
- **Normals (custom normals are supported)**
- **Texture coordinate UV (one layer only is read from the AC engine)**
- **Bones with vertex weight**
- **Nulls/dummies/nodes**
- **Hierarchy structure**
- **Animation data**
- **Basic mesh transformation (scale, rotation, position)**

NOTE: The AC engine does not support 2 OBJECTS with the same NAME in the same model. This will cause the game to crash. Make sure that you pay attention to this rule. Of course, when you have multiple LODs in the same scene, you have to use the same names for functional objects and dummies, but there must not be matching names inside the same export (i.e. within LOD A for example).

Every mesh MUST have one TEXTURE UV set.
The mesh must be (when possible) in quads. Do NOT triangulate the mesh if it is not necessary.
For a skinned mesh you can have as many bones as needed, but every single vertex can be influenced by up to 4 bones and not more.

During the import process the AC Editor ignores all unnecessary data included in the FBX.
Below are the settings to use to correctly export the assets with the 2 supported programmes.
Remember to **set up** your system units before exporting (in XSI it is not required).

For 3DS Max the following settings are required:


System Unit Scale

1 Unit = 1,0 Meters

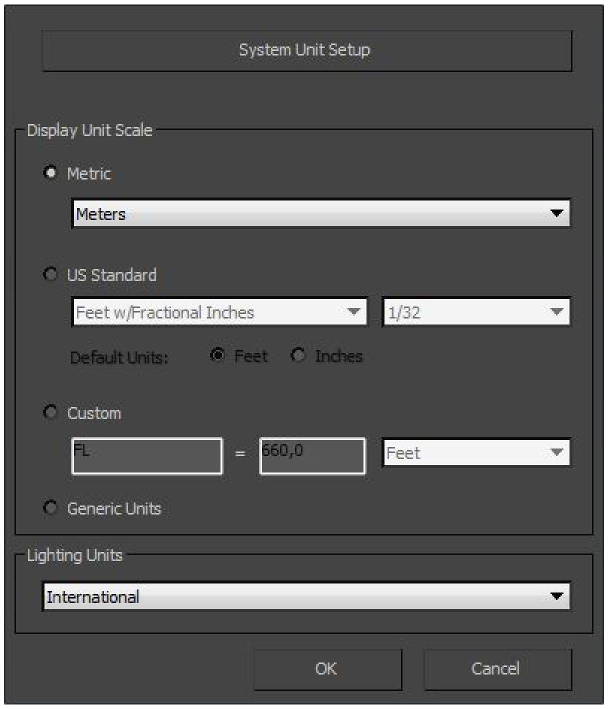
[✓] Respect System Units in Files

Origin 16777215,0m

Distance from origin: 1,0m

Resulting Accuracy: 0,0000001192m

OK Cancel


System Unit Setup

Display Unit Scale

[● Metric] Meters

○ US Standard Feet w/Fractional Inches 1/32

Default Units: [● Feet] ○ Inches

○ Custom FL = 660,0 Feet

○ Generic Units

Lighting Units

International

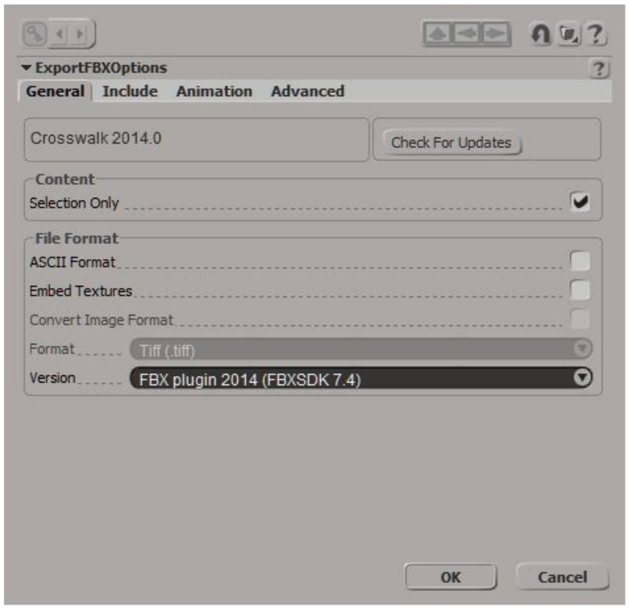
OK Cancel

Make sure to set up the system units BEFORE creating the dummies and exporting the car.

As a limitation in 3DS Max, if the system unit scale is in mm or cm, even if the dimensions are correct, the dummies of the exported model will still have the wrong scale. If the model and the dummies have been created in the wrong scale, one remedy is to export the model as an .fbx and re-import it in a scene using the correct system unit scaling.

NOTE: Make sure that you **reset Xform** after every modification that affects scale. It is advised not to scale suspension and wheel nulls/dummies.

In Autodesk SOFTIMAGE XSI 2014 use the following settings:


ExportFBXOptions

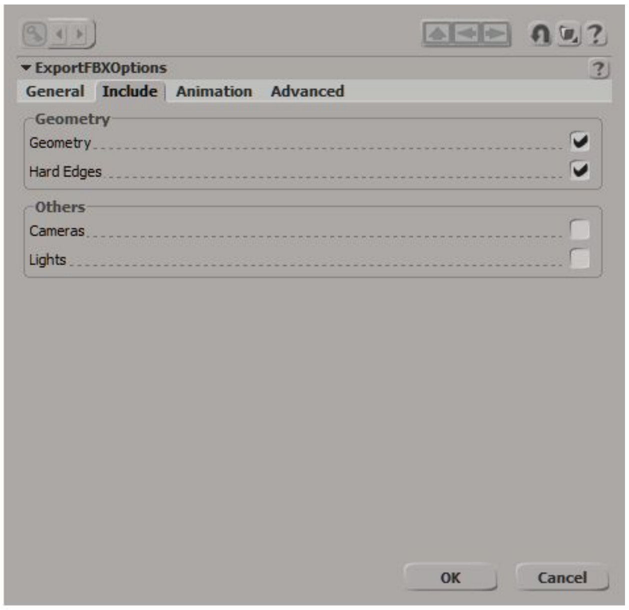
General | Include | Animation | Advanced

Crosswalk 2014.0 — Check For Updates

Content
- Selection Only ✓

File Format
- ASCII Format ☐
- Embed Textures ☐
- Convert Image Format ☐
- Format: Tiff (.tiff)
- Version: FBX plugin 2014 (FBXSDK 7.4)

OK | Cancel

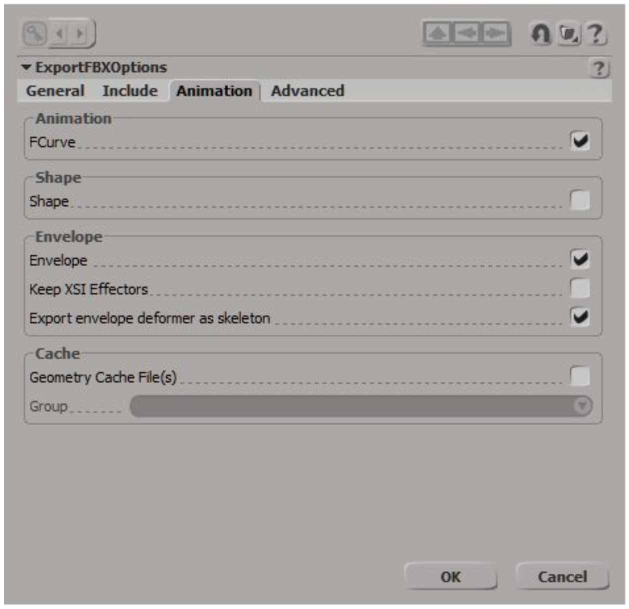

ExportFBXOptions

General | Include | Animation | Advanced

Geometry
- Geometry ✓
- Hard Edges ✓

Others
- Cameras ☐
- Lights ☐

OK | Cancel


ExportFBXOptions

General | Include | Animation | Advanced

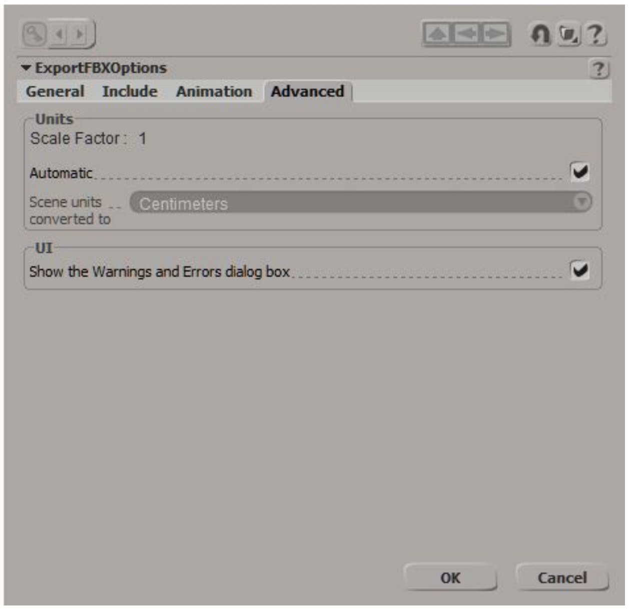
Animation
- FCurve ✓

Shape
- Shape ☐

Envelope
- Envelope ✓
- Keep XSI Effectors ☐
- Export envelope deformer as skeleton ✓

Cache
- Geometry Cache File(s) ☐
- Group

OK | Cancel


ExportFBXOptions

General | Include | Animation | Advanced

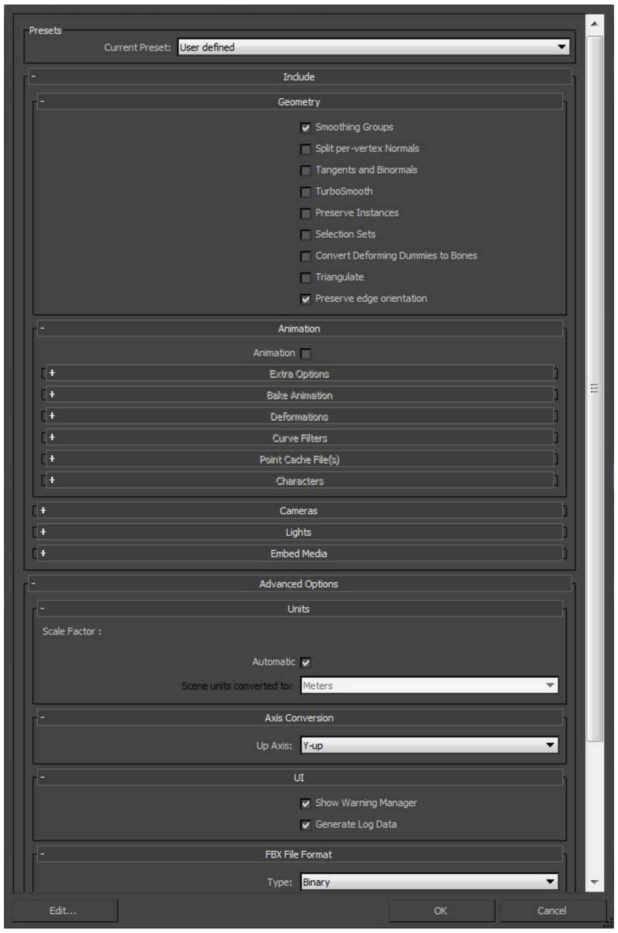
Units
- Scale Factor : 1
- Automatic ✓
- Scene units converted to: Centimeters

UI
- Show the Warnings and Errors dialog box ✓

OK | Cancel

In Autodesk 3DS MAX 2013 use the following export settings:

a) to export the base asset with no animation   b) to export animated nulls/dummies:



Presets
Current Preset: User defined

Include

Geometry
- [✓] Smoothing Groups
- [ ] Split per-vertex Normals
- [ ] Tangents and Binormals
- [ ] TurboSmooth
- [ ] Preserve Instances
- [ ] Selection Sets
- [ ] Convert Deforming Dummies to Bones
- [ ] Triangulate
- [✓] Preserve edge orientation

Animation
- Animation [ ]
- + Extra Options
- + Bake Animation
- + Deformations
- + Curve Filters
- + Point Cache File(s)
- + Characters

+ Cameras
+ Lights
+ Embed Media

Advanced Options

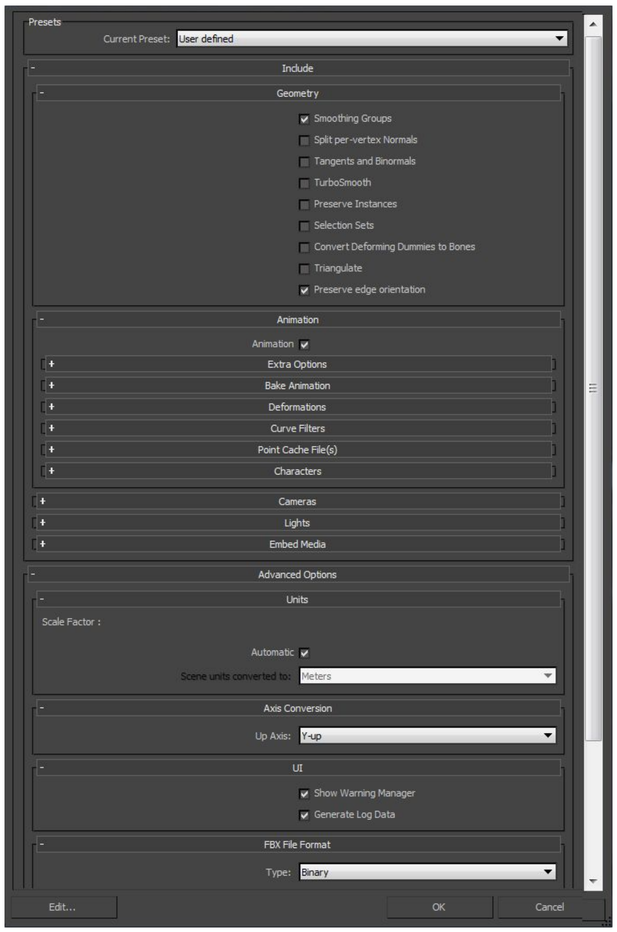
Units
Scale Factor :
- Automatic [✓]
- Scene units converted to: Meters

Axis Conversion
- Up Axis: Y-up

UI
- [✓] Show Warning Manager
- [✓] Generate Log Data

FBX File Format
- Type: Binary

Edit... OK Cancel



Presets
Current Preset: User defined

Include

Geometry
- [✓] Smoothing Groups
- [ ] Split per-vertex Normals
- [ ] Tangents and Binormals
- [ ] TurboSmooth
- [ ] Preserve Instances
- [ ] Selection Sets
- [ ] Convert Deforming Dummies to Bones
- [ ] Triangulate
- [✓] Preserve edge orientation

Animation
- Animation [✓]
- + Extra Options
- + Bake Animation
- + Deformations
- + Curve Filters
- + Point Cache File(s)
- + Characters

+ Cameras
+ Lights
+ Embed Media

Advanced Options

Units
Scale Factor :
- Automatic [✓]
- Scene units converted to: Meters

Axis Conversion
- Up Axis: Y-up

UI
- [✓] Show Warning Manager
- [✓] Generate Log Data

FBX File Format
- Type: Binary

Edit... OK Cancel

Important:

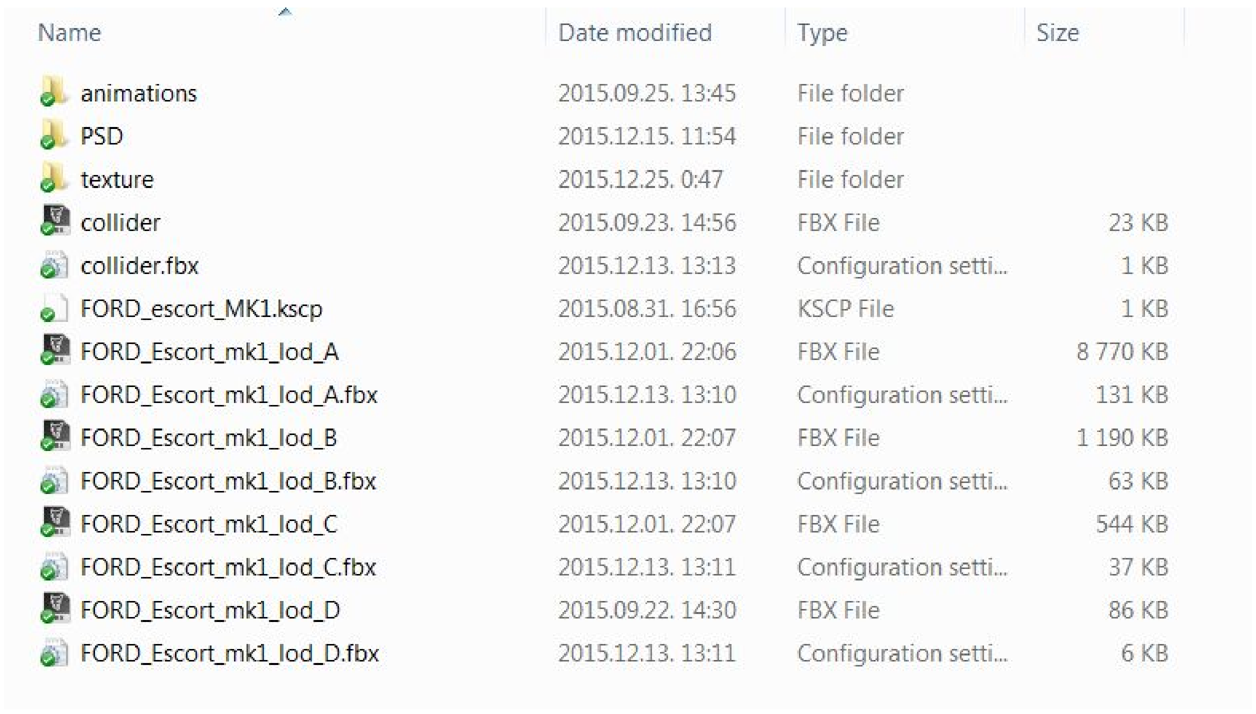
The mesh must be scaled 1:1, rotation must be frozen on the mesh (reset Xform in 3DS Max) and objects should not have animated transformations. Only the dummies/nulls may have different transformations. When they are animated they can be rotated and scaled, and some of them can act as bones for the skinned mesh.

2. BASIC GUIDELINES

PROJECT AND FOLDER STRUCTURE

A vehicle project consists of a total of 5 models, including the high-poly model, 3 additional Level of Detail (LOD) models and a low-poly collider. The naming of the source files must be consistent with the make/type of the vehicle at hand.

The recommended way to set up your project folder is the following:



Name	Date modified	Type	Size
animations	2015.09.25. 13:45	File folder	
PSD	2015.12.15. 11:54	File folder	
texture	2015.12.25. 0:47	File folder	
collider	2015.09.23. 14:56	FBX File	23 KB
collider.fbx	2015.12.13. 13:13	Configuration setti...	1 KB
FORD_escort_MK1.kscp	2015.08.31. 16:56	KSCP File	1 KB
FORD_Escort_mk1_lod_A	2015.12.01. 22:06	FBX File	8 770 KB
FORD_Escort_mk1_lod_A.fbx	2015.12.13. 13:10	Configuration setti...	131 KB
FORD_Escort_mk1_lod_B	2015.12.01. 22:07	FBX File	1 190 KB
FORD_Escort_mk1_lod_B.fbx	2015.12.13. 13:10	Configuration setti...	63 KB
FORD_Escort_mk1_lod_C	2015.12.01. 22:07	FBX File	544 KB
FORD_Escort_mk1_lod_C.fbx	2015.12.13. 13:11	Configuration setti...	37 KB
FORD_Escort_mk1_lod_D	2015.09.22. 14:30	FBX File	86 KB
FORD_Escort_mk1_lod_D.fbx	2015.12.13. 13:11	Configuration setti...	6 KB

The .ini files are created by the AC Editor and include the object and shader properties for the models. The folder called **texture** is obligatory for the editor to load the texture files.

The .kscp file is a project file created by the AC Editor.

BUDGET

The following triangle-counts are recommended in most cases.

Exterior:
LOD A exterior: 125,000 triangles
LOD B exterior: 20,000-25,000 triangles
LOD C exterior: 10,000-12,000 triangles
LOD D exterior: 2,000-3,000 triangles (as low as possible while you can keep the main shape)

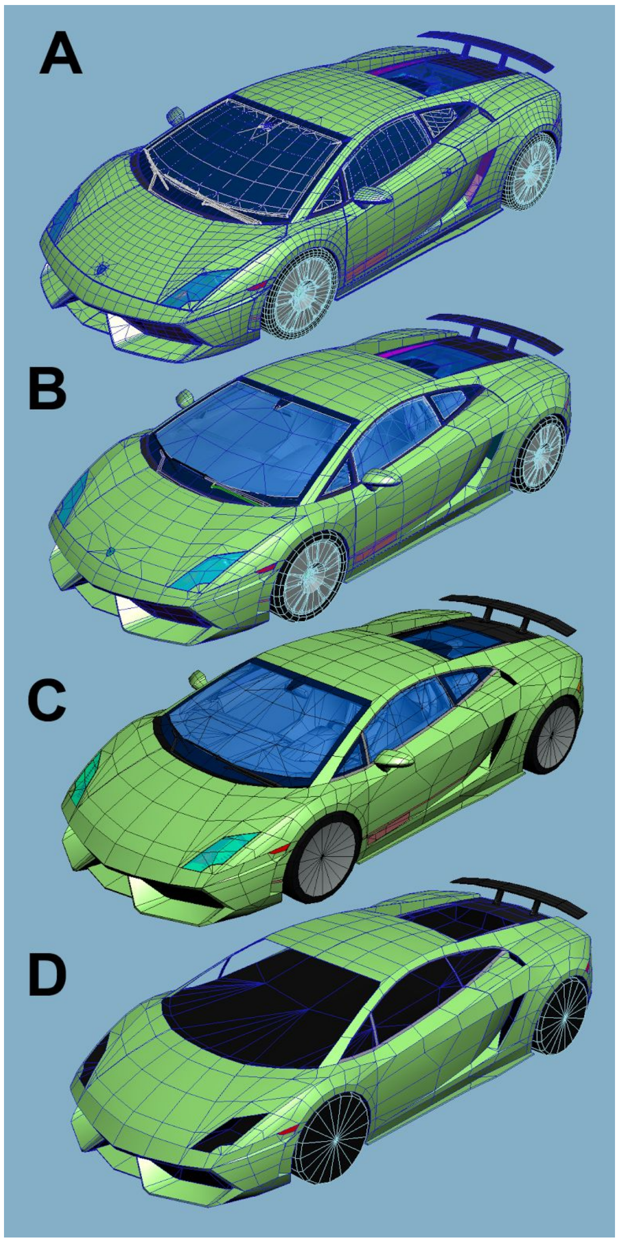
Interior:
HR Cockpit: 125,000 triangles
LR Cockpit: 7,000-10,000 triangles
LR Cockpit in LOD B: 4,000 triangles (as low as possible while keeping a decent quality)
LR Cockpit in LOD C: 2,000 triangles (some detail must remain above window level)

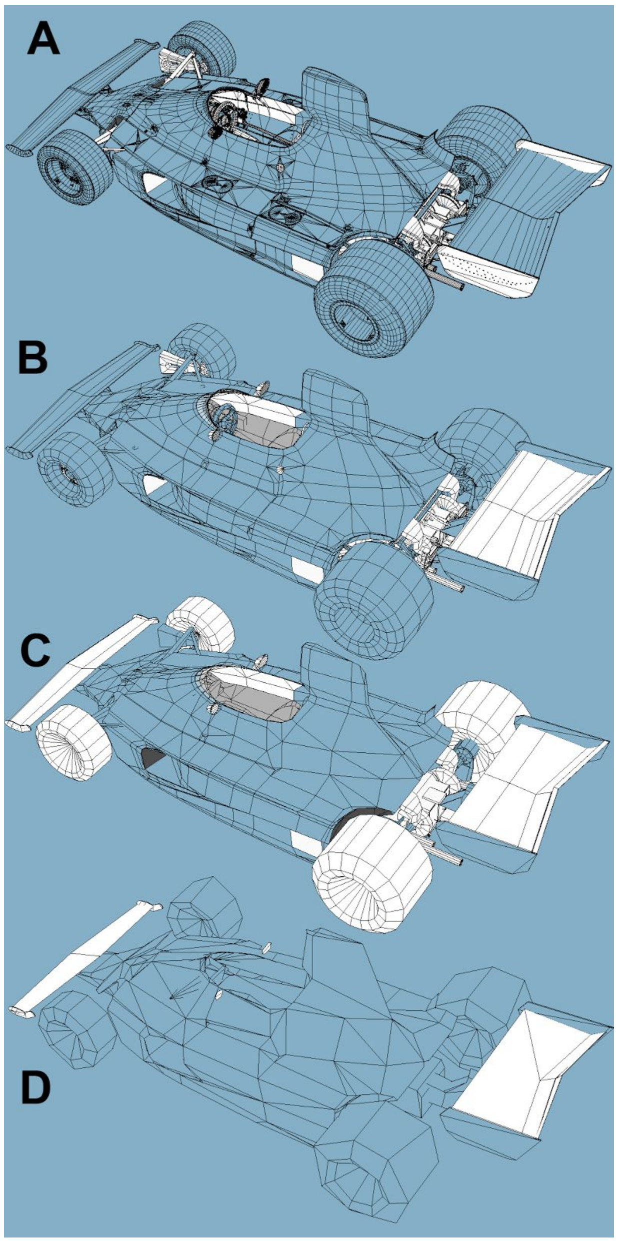
Of course, these numbers are generic and apply to most tin-top cars with elaborate interiors. Open-wheelers with small cockpits can use fewer triangles for the interior and more for details on the exterior, such as the engine and suspension. It is up to the judgement of the modeller to use this budget in accordance with the complexity of the model, but it MUST be **optimised** as much as possible without hurting the overall quality.

When producing the LODs, the most important guideline to follow is to reduce draw calls (number of objects) as well as the number of separate materials. For example, the LR interior should only use 1 material, but if there are customizable parts (such as different interior colour options, it must be possible to use the detail texture defining the colour on the LR interior model, too.

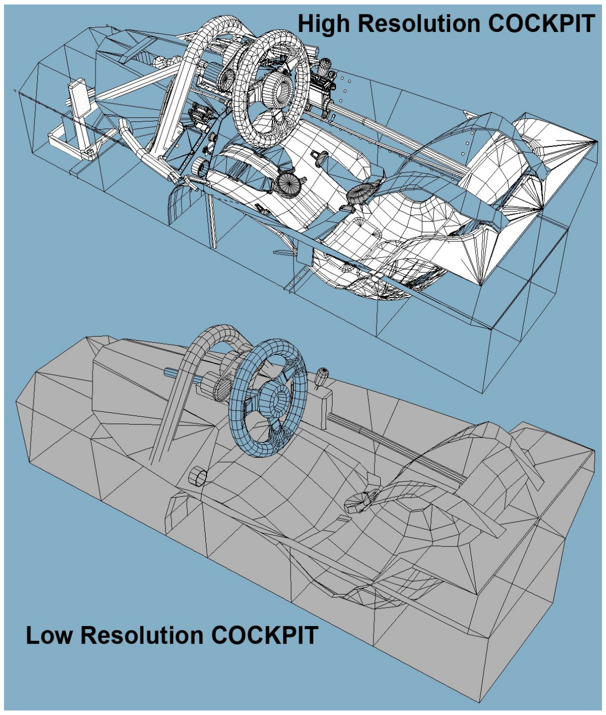
By LOD C, the number of materials and objects should drastically drop, while for LOD D no more than a maximum of 2 materials and a similar number of objects (no rotating wheels are required) should be used.

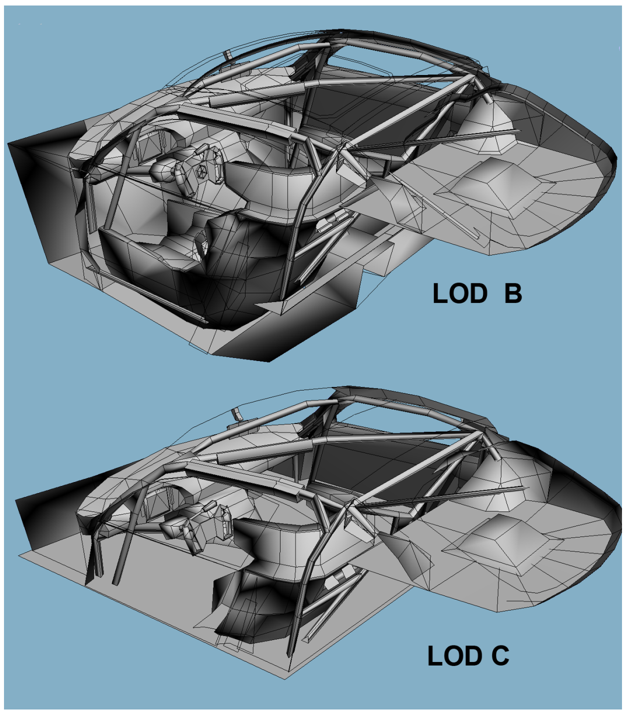
Here are some examples for the progressive degradation of the mesh in the LOD steps:





The same guidelines apply for the interior:





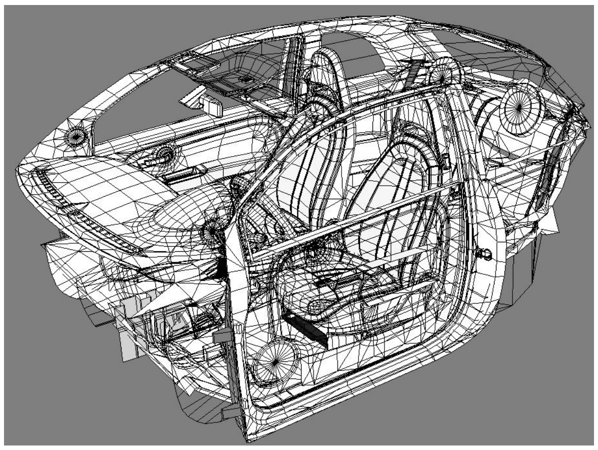
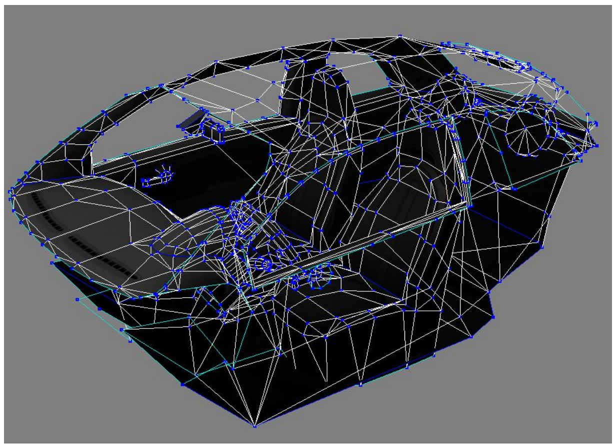
VERY IMPORTANT:

NOTE: Keep in mind that the LOD B will be visible at a distance of 15 meters or closer. If you create a well-made LOD B, you can reduce the distance of the LOD A switch to LOD B and increase game performance.

To achieve this, try to reduce the model keeping the curved parts smooth, parts that create evident reflections, such as glass or curved parts of the body. Try to see how it works in the game and refine it. The switch between the LODs must be as smooth as possible without any visible “jump”.

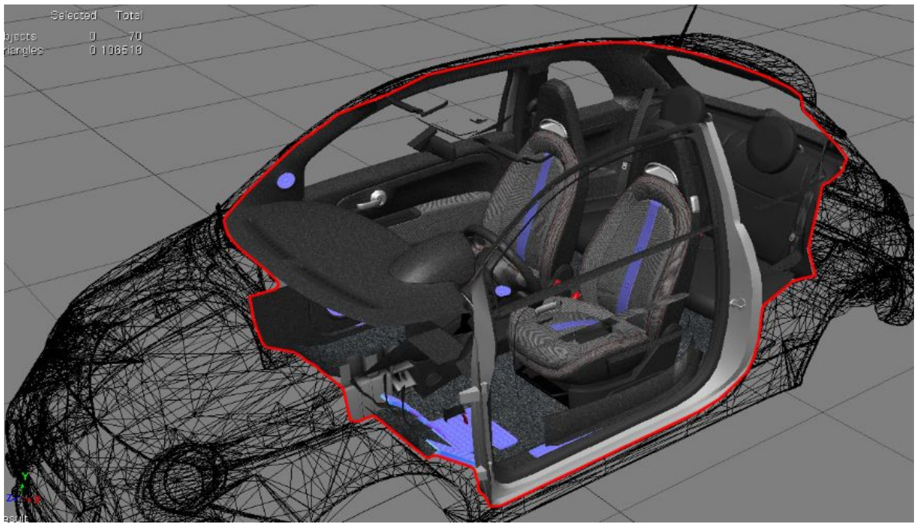
On the LR Cockpit you must keep the most visible parts relatively detailed to ensure that the switch is smooth. In tin-top cars this includes the top of the dashboard and the frame around the side windows and the rear window. In open-seaters the sensitive parts are usually the area around the steering wheel and behind the driver.

NOTE: The LOD B and LOD C **MUST** have a separate LR Cockpit mesh that matches the reduced topology of each LOD exterior mesh. The interior mesh must fit the exterior mesh and the outlining vertices must be **snapped**. Do NOT use the same LR Cockpit mesh for LOD A, LOD B and LOD C. Make sure there are no gaps between the interior and exterior mesh.

LOD A COCKPIT_HR 125,000 tris Cockpit HR LOD example	LOD A COCKPIT_LR 7,000 tris Cockpit LR LOD example
	

In LOD A, the cockpit (see the image above) has 2 LODs, one High Resolution (HR) for the cockpit camera and showroom view, and another Low Resolution (LR) LOD for most exterior cameras, replays, and distant views.

NOTE: The HR and LR cockpit LODs must always fit the exterior LOD A, because while driving, the EXTERIOR MESH that is present is the LOD A.



When the camera moves farther away, the cockpit LR will switch and you get a simplified version of the cockpit, with only one material (in most cases) and a look very similar to the HR version.

In some cases when the car has a **customisable** interior with multiple colour options, more than 1 material is allowed on the LR interior but as a general rule, try to keep it as low as possible.

SCRIPT TO MANAGE LODs

LODs are a set of simplified models that change in relation of the camera distance.
This process is necessary in order to optimize the framerate in the game.
The LOD switch can be controlled via script, named lods.ini, located in AssettoCorsa/content/cars/CAR-NAME/data. The script contains the following values:

```
[COCKPIT_HR]
DISTANCE_SWITCH=6 ;Indicates the distance (in meters) when the cockpit HR change to the cockpit LR (if present)
[LOD_0]
FILE=abarth500.kn5
IN=0
OUT=15 ;Indicates the distance (in meters) when lod_A changes with lod_B (if present)

[LOD_1]
FILE=abarth500_B.kn5
IN=15
OUT=45   ;Indicates the distance (in meters) when lod_B changes with lod_C (if present)

[LOD_2]
FILE=abarth500_C.kn5
IN=45
OUT=200  ;Indicates the distance (in meters) when lod_C changes with lod_D (if present)

[LOD_3]
FILE=abarth500_D.kn5
IN=200
OUT=1500   ;Indicates the distance (in meters) when lod_D disappears from visual.
```

NOTE: Verify that the distance of LOD “out” value matches the “in” value of the next LOD, otherwise your car will disappear before the switch.

ADDITIONAL INFO: The LOD B must have the same null hierarchy as the LOD A except for the nulls COCKPIT_HR, STEER_HR and the FLYCAMS, which should not be present. Based on how visible the elements are, it is up to your judgement to remove other non-essential nulls, such as wings, bumpers, the hood etc. in LOD C.

3. SCENE STRUCTURE

In order to work in-game, the car needs specific nulls to present in all LODs, making sure that all car parts are functioning properly. An example scene is provided in the Dropbox folder with a folder structure and hierarchy to follow (Scene templates/scene_nosuspanim_example_max).

Make sure you keep only one set of nulls in your scene and that you use a clear layer structure to hide/unhide layers for exporting various LODs (see example scene).

Nulls that MUST be present in ALL LODs:

SUSP_LF	suspension Left Front
SUSP_LR	suspension Left Rear
SUSP_RF	suspension Right Front
SUSP_RR	suspension Right Rear
WHEEL_LF	wheel Left Front
WHEEL_LR	wheel Left Rear
WHEEL_RF	wheel Right Front
WHEEL_RR	wheel Right Rear
COCKPIT_LR	cockpit Low resolution node
STEER_LR	steer Low resolution node
DISC_LF	Brake disc Left Front
DISC_LR	Brake disc Left Rear
DISC_RF	Brake disc Right Front
DISC_RR	Brake disc Right Rear

There are secondary nulls that are needed to complete the car, but are not essential, so it means that in some cases those object can be excluded from certain LODs:

COCKPIT_HR	cockpit High resolution node (A only)
STEER_HR	steer High resolution node (A only)
RIM_LF	Rim Left Front (A and B)
RIM_LR	Rim Left Rear (A and B)
RIM_RF	Rim Right Front (A and B)
RIM_RR	Rim Right Rear (A and B)
RIM_BLUR_LF	Rim Blurred Left Front (A and B)
RIM_BLUR_LR	Rim Blurred Left Rear (A and B)
RIM_BLUR_RF	Rim Blurred Right Front (A and B)
RIM_BLUR_RR	Rim Blurred Right Rear (A and B)

When it is required to manually animate the suspension, or the car has Dion axle suspension, you have to include some extra nulls in your scene:

REAR_AXLE	for the center of rotation of the Dion Trunk axle
HUB_LF	Hub for the suspension Left Front
HUB_LR	Hub for the suspension Left Rear
HUB_RF	Hub for the suspension Right Front
HUB_RR	Hub for the suspension Right Rear

Nulls/dummy used to define the broken glass mesh:

DAMAGE_GLASS_CENTER_1	(A and B)
DAMAGE_GLASS_FRONT_1	(A and B)
DAMAGE_GLASS_REAR_1	(A and B)
DAMAGE_GLASS_LEFT_1	(A and B)
DAMAGE_GLASS_RIGHT_1	(A and B)

The number at the end of name can increase if other nulls/dummy are present. The implementation of damage glass is fully explained in the **DAMAGE GLASS** section.

About Nulls/dummy to animation parts, see the **Additional nulls for animations** section.

For the DAMAGE of car elements, the following nulls must be placed in the PIVOT point of the object around which the object rotates upon impact. The naming conventions for damageable parts are the following:

FRONT_BUMPER	(A and B)
REAR_BUMPER	(A and B)
MOTORHOOD	(A and B)
REAR_HOOD	(A and B)
FRONT_WING	(A and B)
REAR_WING	(A and B)
REAR_EXTRACTOR	(A and B)

Additional dummies can be:

WIPER_#	for wiper animation	(A, B and C)
FRONT_LIGHT	for headlight animation	(A, B and C)
DISPLAY_DATA	for digital displays	(A only)
DOOR_L and DOOR_R	for exterior door animation	(A and B)
DOOR_L_1 and DOOR_R_1	for interior door animation	(A only)

MESH PARTS OF A GENERIC CAR MODEL

The components of a car must be divided in many parts in order to manage animated objects, meshes and other features present in game. Here is a list of mandatory and optional mesh objects.

Common exterior parts:

MAIN BODY	Required - must be present in LOD A and LOD B
DOORS	optional - only in LOD A if present on the model. In LOD B the doors are not animated but welded to the main body
MOTORHOOD	Depends on car type - if needed, must be present in LOD A and LOD B
FRONT BUMPER	Depends on car type - if needed, must be present in LOD A and LOD B
REAR BUMPER	Depends on car type - if needed, must be present in LOD A and B
WHEEL HUB	Optional - contains the brake calipers must exist on LOD A and LOD B
WHEEL RIM	Required - must be present in LOD A and LOD B. In LOD C the wheels are simplified.
WHEEL RIM BLUR	Required - a version of the rim but with a blurred texture, must be present in LOD A and LOD B
WHEEL TYRE	Required - must be present in LOD A and LOD B. In LOD C the wheels are simplified
BRAKE DISK	Depends on car type - if needed, must be present in LOD A and LOD B
FRONT LIGHT	epends on car type - if needed, must be present in LOD A, LOD B and LOD C
REAR LIGHT	Depends on car type - if needed, must be present in LOD A, LOD B and LOD C
WIPERS	Depends on car type - if needed, must be present in LOD A, LOD B and LOD C
FRONT WING	Depends on car type - if needed, must be present in LOD A, LOD B and LOD C
REAR WING	Depends on car type - if needed, must be present in LOD A, LOD B and LOD C

Cockpit mesh parts:

COCKPIT_HR	Required - high resolution cockpit, the one that you see in cockpit view. Low Resolution (LR) version also required, including LOD B and LOD C
STEER	Required - steering wheel HR and LR interior, LOD B and LOD C
STEER PADDLE	Depends on car type, required also in Low Resolution (LR) and LOD B
SHIFT	depends on car type - required also in Low Resolution (LR) and LOD B
SEATBELTS	Depends on car type - if needed, required also in Low Resolution (LR) and LOD B and LOD C. In LOD A, both ON and OFF position required. In LR, LOD B and C only the ON position model is needed! ON position needed only for the driver, not the passengers.

MESH NAMING CONVENTIONS

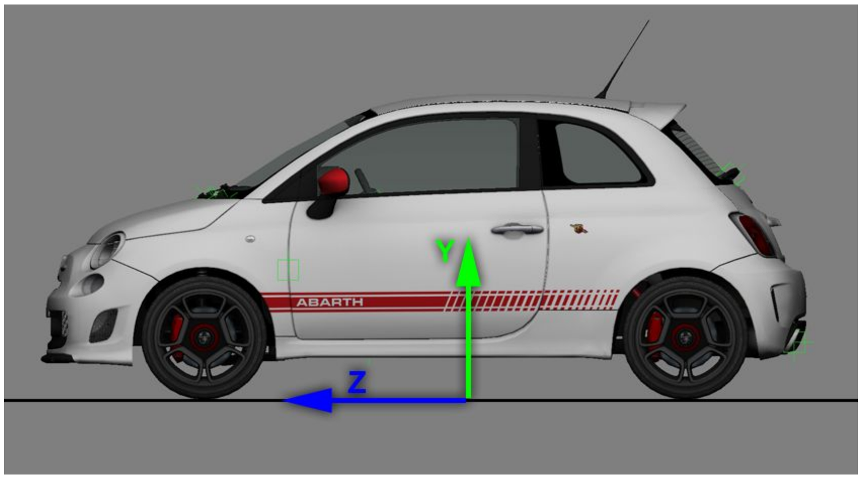
Be consistent in naming mesh objects within your scene. Use a pre-tag such as MESH_ or GEO_ for mesh objects to differentiate them from null objects. Make sure you keep the same names for functional objects (lights and other emissives etc.) throughout the entire scene for all LODs to ensure that the scripts works as intended for each LOD.

You can choose to name your objects based on location (when using multi-materials), such as GEO_front_bumper and GEO_main_body (in this case there will be sub-objects divided by the editor at exporting), or based on material grouping, such as GEO_paint_body and GEO_chromes_body.

SETTING UP THE CAR MODEL INSIDE THE 3D SPACE

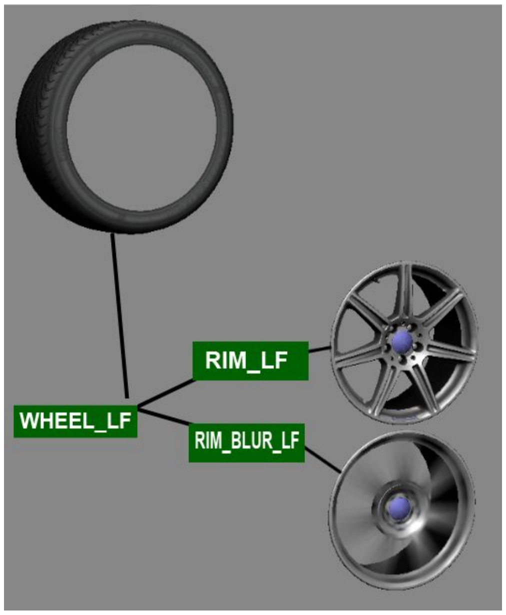
You can find example scenes in **Scene templates** folder in the sdk/dev folder! The car must be oriented as shown in the image: The Z vector must be the front direction. The model must be placed with the wheels touching the ground on the 0 coordinate (Y) (see image below).

The model bounding box must be centered in YXZ = 0.0.0. (See image below)
The car must have 4 different Level of Detail models that must share the same position and orientation!



HIERARCHY and ORIENTATION

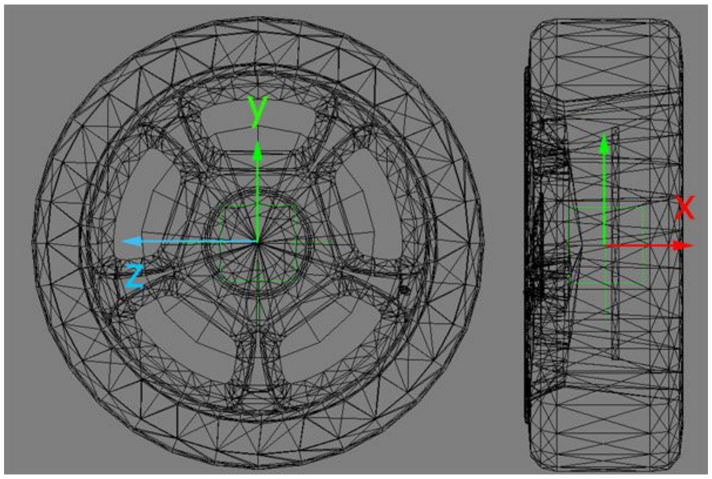
A template file is provided as an example, showcasing how to setup a correct hierarchy for a car. The file contains a set of NULLS or DUMMY objects, that define the CENTER position of any piece of the car. The names of these NULLS must follow specific rules showcased below.
The EDITOR will recognize these essential NULLs in order to define the rotation pivot of the wheels, the suspensions and any animatable object in the car.



NOTE:
Any object that is not a child of a NULL/ DUMMY will be managed like part of the CAR CHASSIS.
All the pieces of the geometry that belong to the car must be placed in a HIERARCHY to define the specific properties of each mesh object in the game.

Example image on the right: wheels objects are linked to the wheel null/dummy.

In the same way you must place all the others pieces like children of the correct NULL that is designed for the part that you are creating. So for the rim there is a dedicated NULL and so on for all the other parts. Remember that every NULL is also the CENTER of rotation. If your mesh is not properly placed under a NULL with a correct center of rotation, the mesh will rotate the wrong way.



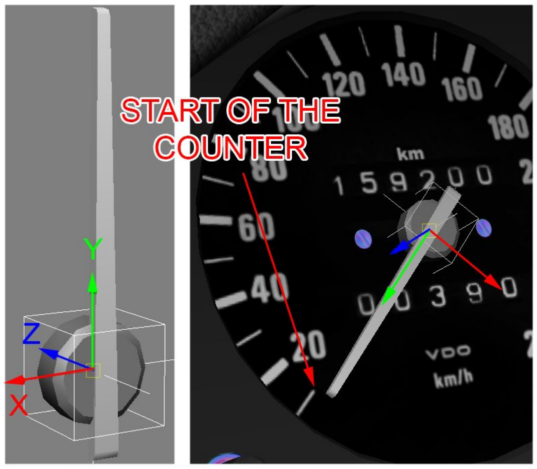
See the example on the left: the geometry of the WHEEL is centered exactly on the NULL.

This will allow the rotation to be correct. In the car example file you'll be able to explore how we placed all the NULLs and the relating mesh objects. The **BRAKE DISK** must have the center in the same EXACT position of the wheel and the rims.

4. FUNCTIONAL MESH ELEMENTS

ANALOG INSTRUMENTS

To animate a needle on the dashboard, the ARROW_ mesh needs to be placed under a proper null. Just as other objects, ARROW_ nulls must follow specific conventions:
ARROW_SPEED
ARROW_RPM
ARROW_TURBO
ARROW_FUEL
ARROW_WATER_TEMP
ARROW_TIME
*ARROW_LIMITER



Each ARROW_ null must be linked to the COCKPIT_HR null!
Each instrument is controlled by specific values in the analog_instruments.ini script, located inside "content/cars/car_name/data" folder.
The needle mesh must be created in neutral position and linked as child of a specific null.
The arrow then must be rotated to the 0 position as shown in the image above.

The Y axis determines the arrow position on the gauge and must be placed at 0 (ZERO) or the start of the gauge at hand.

Note: Z axis must always point FORWARD

*The **ARROW_LIMITER** is a tell-tale found for example in the Lotus 49 and Lotus 72D cars, showing the maximum RPM in a given stint. The rotation of the ARROW_LIMITER null must be the **same** as the ARROW_RPM null, and it requires **NO** script, as it is controlled from within the core engine.

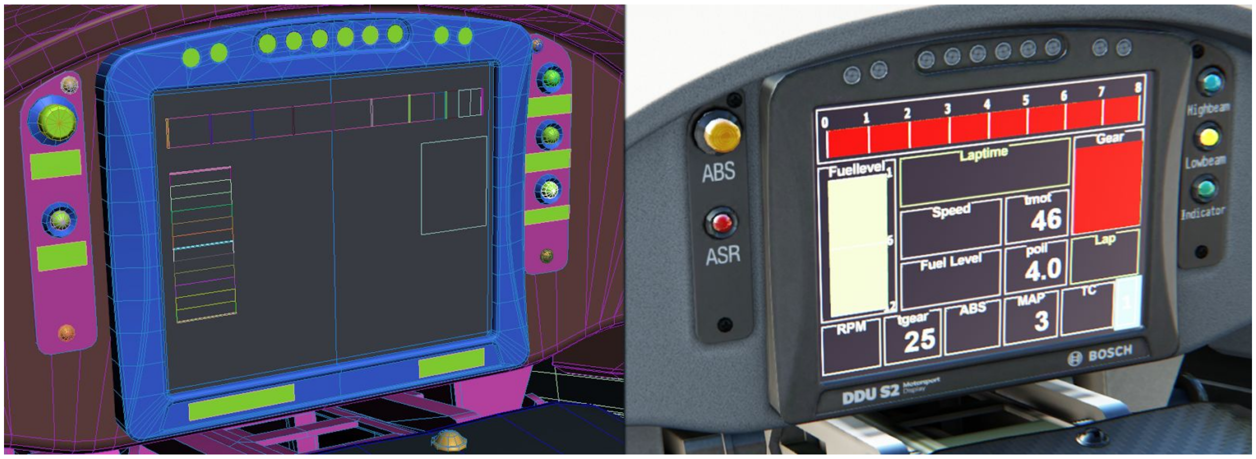
You can use the Data Scripts tab in the KS editor to set up the analogue scripts (see **EDITOR** section).

To ensure compatibility in the future, if present, set up gauges that are currently not supported by the engine in a similar fashion, with name conventions that are consistent with the existing rules and the function of the instrument:

ARROW_VOLTAGE
ARROW_OIL_TEMP
ARROW_OIL_PRES
ARROW_FUEL_PRES
ARROW_WATER_PRES

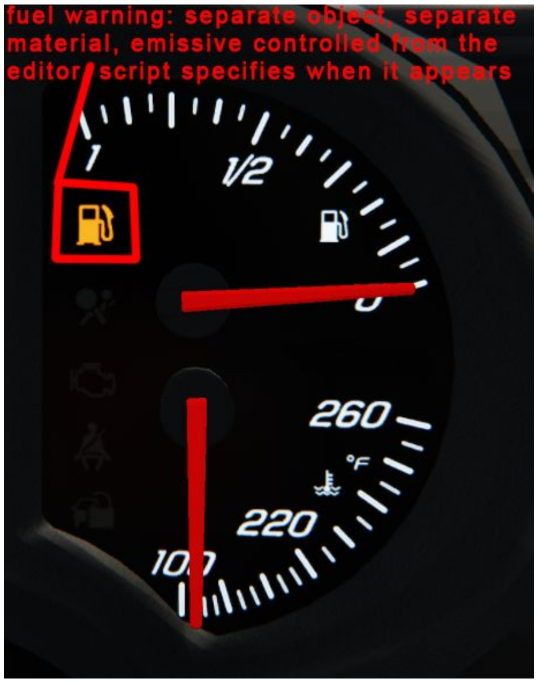
LEDs and DIGITAL DISPLAYS

It is possible to have various LEDs and indicator lights light up in the cockpit during driving. It is recommended to make the cockpit dynamic with as many functional items as possible.



Each individual LED (such as for RPM, boost or KERS) or bar TAG must be a separate object and numbered in a series:

LED_RPM_# where the “#” is the number of each specific item in a series
TAG_RPM_#
KERS_CHARGE_#
KERS_INPUT_#
TURBO_#

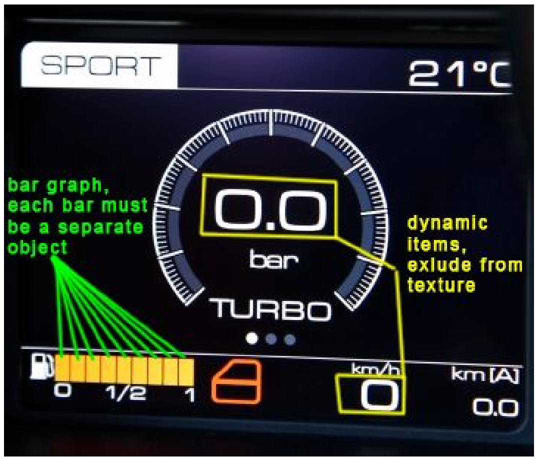

fuel warning: separate object, separate material, emissive controlled from the editor, script specifies when it appears

Some cars have multiple displays and it is possible that certain items (such as RPM) are shown in more than one screen. In this case, make sure that you differentiate between the two readouts: LED_RPM_1_# and LED_RPM_2_# etc.
For functional LEDs or warning lights, each item must be a separate object, using the following naming conventions:

LED_LIGHT	headlight indicator light
LED_FUEL	fuel warning light
LED_KERS	KERS status light
LED_IGNITION	ignition status light

There are **two** ways to control specific items on the dashboard. The first one is where the mesh is always present and the script controls the emissive value on a per object basis. This is the method used for RPM LED series, headlight indicators and ignition status lights.

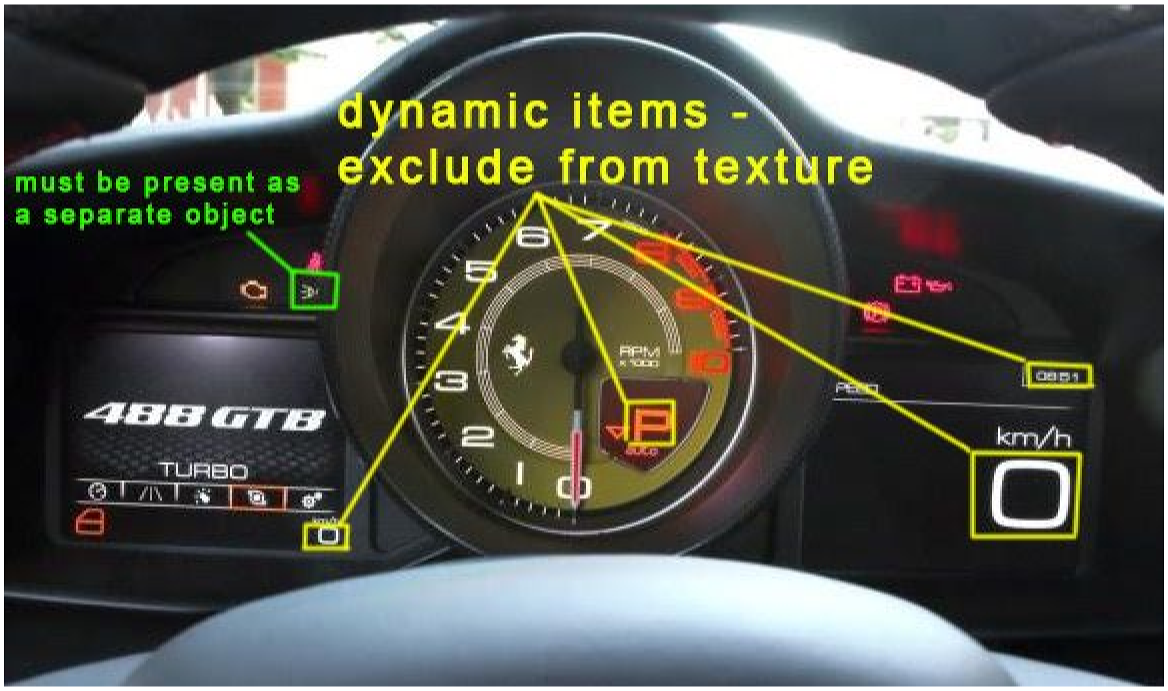
With the second method the mesh is disabled by default and the script controls how and when it should appear with the shader and object properties set up in the editor. It should be noted that the meshes appear in the editor and the showroom. This is used for dynamic RPM bar graphs, boost bar graphs, fuel level bars, shift indicators, fuel warning lights and KERS bar graphs.


SPORT
21°C
bar graph, each bar must be a separate object
0.0
bar
dynamic items, exlude from texture
TURBO
0 1/2 1
km/h
0
km [A]
0.0

NOTE: If in the second case the item is a light with emissive values, it must also use an individual material with the desired emissive value set in the editor (e.g. fuel warning light).

NOTE: It is very important that you do extensive research about dashboard functionality and digital screens (if present) and prepare the model for dynamic displays.

When creating the texture for the digital display, you must take into consideration the dynamic readouts that are supported by the engine and must not include them in the static diffuse texture.


dynamic items - exclude from texture
must be present as a separate object
488 GTB
TURBO
RPM x1000
km/h
0

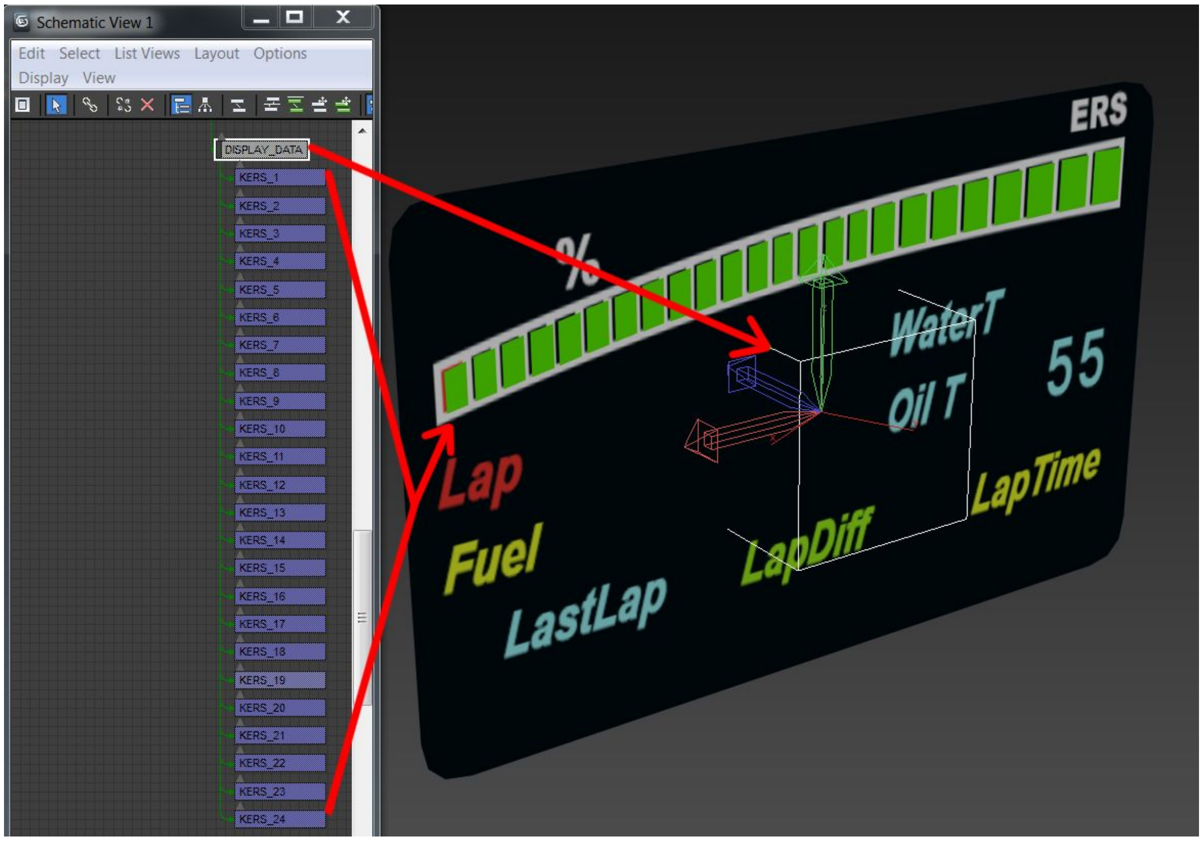
NOTE: Do NOT use the convention _01, _02, _03 etc. for single digits for the suffix of tag object names, always use _0, _1, _2 etc. E.g. LED_RPM_0, LED_RPM_1 etc.
The following items are currently supported by the game engine:

Time
Gear
Speed
RPM
Water temperature
Fuel level (bar graph)
Fuel level (litres)
KERS charge (bar graph)
KERS input (bar graph)
Turbo boost (bar graph)
Turbo boost (pressure)
Lap time
Previous lap time
Difference from previous lap
TC setting
ABS setting
Headlight indicator
Fuel warning light
Current lap
Total laps
Ambient temperature
Km with current fuel left
KERS charge readout
Estimated fuel
Any RPM-dependent status indicator
Tyre pressure
G-meter
Placeholder script for any static text or numbers

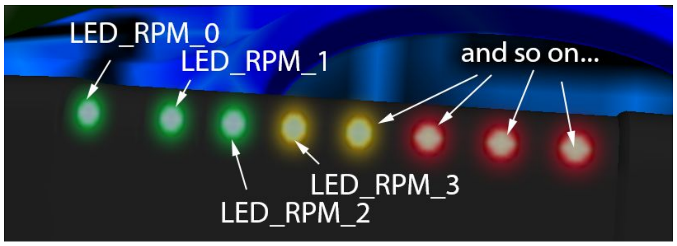
IMPORTANT: If there is a digital display, create a NULL called DISPLAY_DATA with the orientation shown in the image below. If there are more displays, use a serial number (DISPLAY_DATA_1 etc.) to specify each individual screen. The DISPLAY_DATA null is the reference for the items in the digital_instruments.ini, it serves as a reference point and makes sure the text appears on the same surface as the display. For this reason if the display is rotated/tilted, the null must follow the same orientation. To avoid clipping, place the null so that its pivot point is in front of the mesh by a few millimetres and not directly on it.

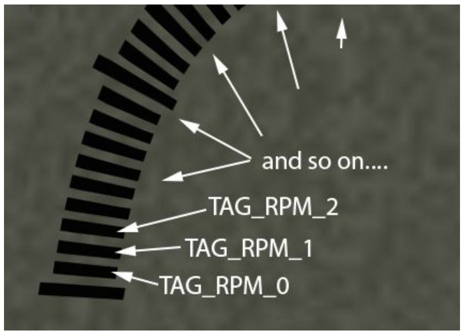
NOTE that the above list is not complete. Please visit the following link to see a community and dev-assisted thread on the official support forum for a complete list of digital scripts and their implementation:
http://www.assettocorsa.net/forum/index.php?threads/analog-digital-instruments-lights-q-a-request-official-support-here-check-first-post.12249/

DISPLAY_DATA null orientation and example for KERS bar graph:



NOTE: When the display is located on the steering wheel, the DISPLAY_DATA null and all the TAG/RPM mesh objects must be a **child** of the STEER_HR null to make sure they rotate along with the steering wheel.





DIGITAL PANELS

Digital panels can be used for two functions: **Push-to-Pass** status and on-track **Position**.
This feature requires a digital_panels.ini in the car’s data folder and pre-drawn numbers in the your_car/texture/display_panel folder. As an example, take a look at the content/cars/ks_audi_tt_cup/texture folder in your game install folder.

You will also need a parent NULL (e.g. DISPLAY_PANEL), with the same orientation rules that exist for the digital instruments (see above).

Position

Use the following script to activate the function:

```
[FULLPOSITION_SERIES_0]
PREFIX=textName_          prefix of texture names located in car_folder/texture/display_panel
POSITION=
PARENT=DUMMY              parent dummy name
START=0                   postfix start
END=9                     postfix end
DIGIT=1                   set 1 for second digit, 10 for first digit
WIDTH=30
HEIGHT=40
COLOR=255,255,255,255
INTENSITY=2
```

Push-to-Pass

Use the following script to activate the function:

```
[PUSH2PASS_SERIES_0]
PARENT=PANEL_P2P                    name of parent dummy
POSITION=0.0615,-0.068,0
WIDTH=0.124
HEIGHT=0.137
TRIGGER=0
PREFIX=num_                         prefix of texture names
COLOR=255,255,255,255
INTENSITY=40.0
START=0                             name postfix to start from
END=9                               name postfix to end at
DIGIT=1                             (=1 for second digit, =10 for first digit)
BLINK_HZ=5                          blink rate when activated (=0 for no flashing)
```

P2P status led

```
[PUSH2PASS_LED_0]
OBJECT_NAME=LED_P2P
EMISSIVE=0,0,800
DIFFUSE=0.35
INVERTED=0                          for inverse function
BLINK_HZ=0                          if higher than 0, it blinks
```

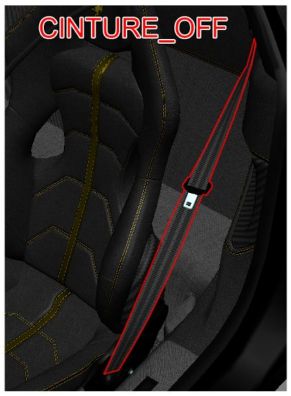
Known limitation: in replays, the P2P and displayed position status is not communicated, which is why the panels will show incorrect or placeholder values.

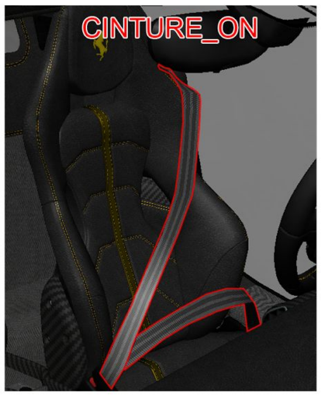
SEATBELTS

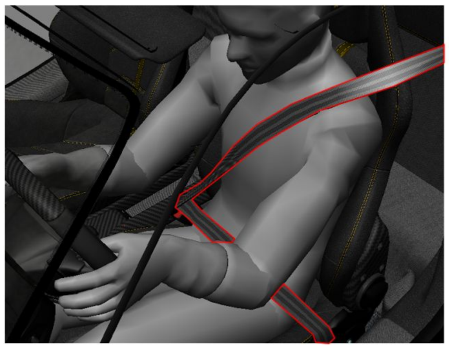
The cockpit contains two different mesh objects for the belts: One for the belt ON and another for the belt OFF. These two meshes must be linked as a child of the null COCKPIT_HR and must be named as follows:

CINTURE_ON	for the belt on the driver when is driving
CINTURE_OFF	for the belt on the seat, without driver (showroom view)

NOTE: The names are in ITALIAN (CINTURE = SEATBELT)….
To create the proper mesh of the belt on a driver, place the driver first, then animate it and verify how the arms move in order to avoid compenetration with the belt mesh.



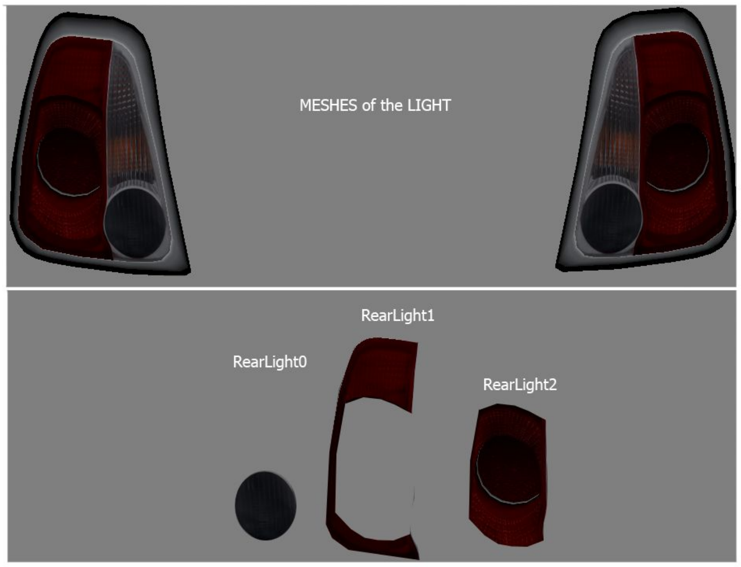




The seatbelt mesh must be modelled also in the cockpit LR but only the CINTURE_ON mesh. When you see the cockpit LR it means that you are in game, not in the showroom, so a driver is in the car and you have to show the CINTURE_ON configuration only, without the belt being separated from the rest of the cockpit mesh.

LIGHT MESH AND SCRIPTS

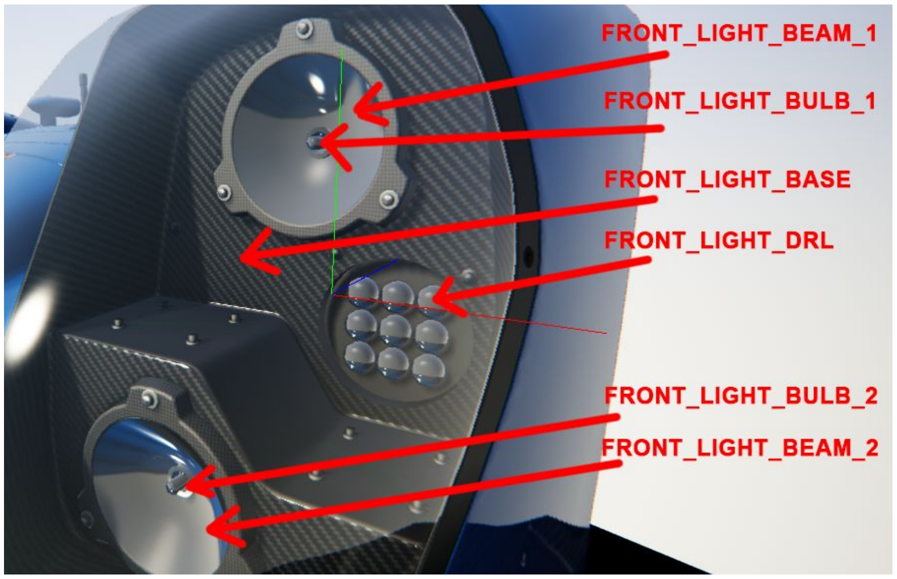
Each car must have individual objects. The light mesh objects must be separated and detached from the body of the car and use specific naming conventions. The mesh name must be controlled from the lights.ini script. The same scripts include the instructions for the ON/OFF conditions, as well as the light emission colour.



Example image on the right: The mesh of the light is made from different parts, which are divided according to their function.
Some examples:
Position lights, brake, rear, standard front lights, high beams etc.

Note: There is no need to split the lights up as "right" and "left". They can be one mesh because they turn on together.

IMPORTANT: similarly to the dashboard, please do an extensive research about light functionality, and strive to implement as many functions as possible. Each light source must be **detached** as a separate object, avoid keeping all the difference reflectors and bulbs in one object. This way, each element can be controlled individually to achieve realistic results. A good example is shown in the image below:



Open the lights.ini script located in the “data” folder
The script contains the following values:

```
[HEADER]
VERSION=3                          Script version. Keep this value like is.

[BRAKE_0]
NAME=REAR_LIGHT                    name of the mesh to light up
COLOR=500,60,40                    RGB value when you press the brake pedal
OFF_COLOR=50,12,8                  RGB value for position light when brakes are off

[LIGHT_0]
NAME=FRONT_LIGHT                   name of the mesh to light up
COLOR=240,195,180                  RGB emissive value when the front light is on
OFF_COLOR=50,50,70           RGB emissie value for day-light (optional)
```

In the above example: The NAME= value affects a mesh called REAR_LIGHT (as seen in the image above). The COLOR= value assigns a colour when the brakes are on (you are pressing brake pedal). The line below OFF_COLOR= is the emissive value of the brake light (in some cars, the same mesh is lit up when you turn the lights on and when you brake).
Different functions and colours can be assigned to different meshes.
As an example, the value COLOR=3,0,0 assigns a specific colour to the light. The mesh is lit using HDR and there is no maximum limit of intensity.
The values for the COLOR= parameter are in RGB 0 to 1 range, so a value of 1 means the maximum value of the RGB scale (256). The values can go over 1 if more intensity is needed. As an example, a value of 240 is given to the [LIGHT_0] section, in order to produce a strong glow.

NOTE: for a glowing brake light we recommend an R (red) value between 150 and 850. For day running lights, we recommend values between 40 and 100, while for high beams, we recommend values ranging from 250 to 800.

NOTE: the dashboard headlight indicator lights and the dashboard lighting are also controlled by the lights.ini. The dashboard objects and any other objects that light up in the interior must be detached and named according to the following convention:

LIGHT_GAUGE_#
LIGHT_INTERIOR_#

Don’t forget to detach the objects and link them to their respective dummies if they are located on moving objects (e.g. the steering wheel)

Modern Formula cars can use the following script for **ERS status flash light** at the rear:

```
[HEADER]
VERSION=3
FLASHING_BLINK_TIME=0.35            Blink length in seconds
FLASHING_REPEAT=1                   Number of flashes on activation
KERS_BLINKING=1                     Line means KERS blinking is enabled
NO_LIGHT_SWITCH=1                   Line means it will not work as headlight

[LIGHT_0]
NAME=g_Rain_Lights
COLOR=180,0,0
PITLINE=1                           1 means it flashes in pitlane
KERS=1                              1 means it flashes when KERS harvest is active
SPECIAL=1                           1 means light toggle is disabled
```

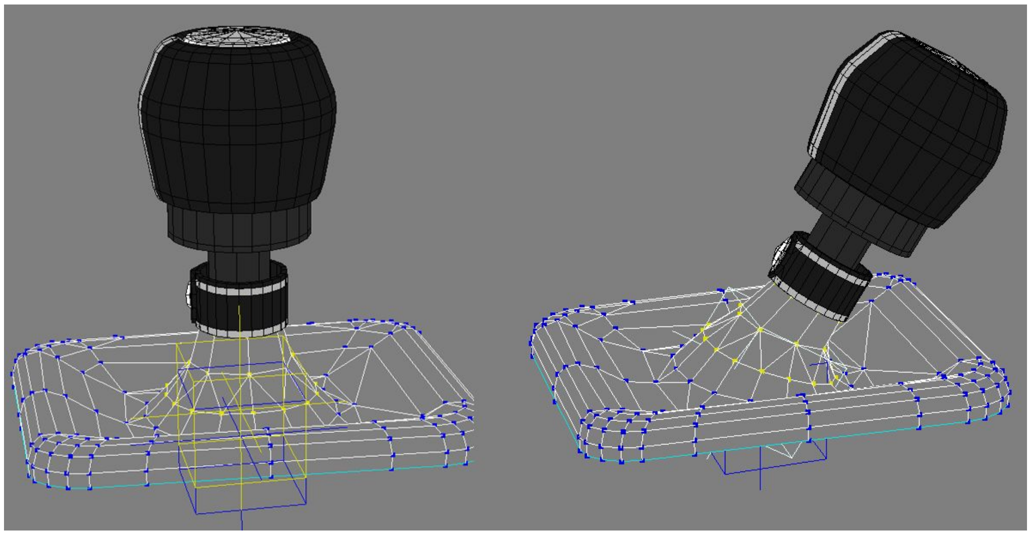
Use the following script to use **flash function** for headlights and **flashing pitlane** lights:

```
[HEADER]
VERSION=3
FLASHING_BLINK_TIME=0.15
FLASHING_REPEAT=8

[LIGHT_0]
NAME=LIGHT_FRONT
COLOR=530,420,50
FLASH=1                             1 means will flash when flash toggle is pressed

[LIGHT_1]
NAME=LIGHT_RAIN
COLOR=95,0,0
PITLINE=1                           1 means it will flash in pitlane
SPECIAL=1                           1 means light toggle is disabled
```

SKINNED MESH



FBX skinned mesh objects are supported by the game engine. Skinned mesh objects can have as many bones as necessary but no more than 4 bones influencing a single vertex.

A good example of skinned mesh is the driver (explained later) or the gearshift lever with a fabric skirt at the base of the lever.

The example image up here show kind of usage that you can do.

Rules for creating a skinned mesh:

1) All vertices must be influenced by at least one bone. If a vertex is not influenced by a bone, its world coordinates will be 0,0,0, resulting in a long polygon that spawns from the center of the 3D world until your space position.
A non-skinned object can be linked with a skinned mesh. Connect the non skinned object as a child of the skinned mesh. In the above image, the skin has 2 bones but the handle is a parent of the non-skinned yellow null.

2) Every material with a skin rig must be unique. A material cannot be used on a standard mesh and at the same time on a skinned mesh. Two different materials must be created, one for the standard mesh and another one for the skinned mesh.

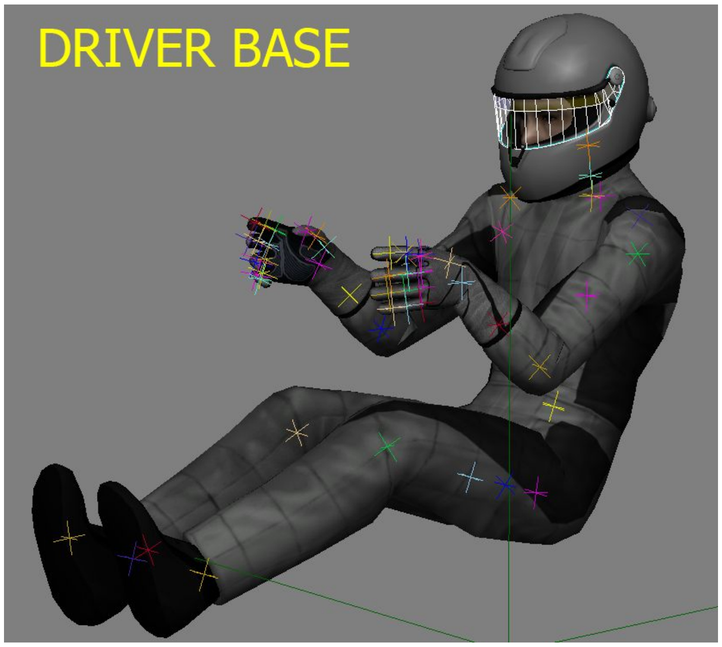
3) The skinned-mesh dedicated material, must be KsSkinnedMesh or the KsSkinnedMesh_NMDetail. The animation works only when the skinned material is assigned to the skinned mesh.

4) The skinned mesh must have the pivot in the 0.0.0 coordinates of the world. It can be child of another dummy, but this dummy must also have the coordinates of 0.0.0.
We usually put the skinned mesh of the gearshift or something else in the cockpit as the child of the cockpit dummy/null. And the cockpit Dummy/null is usually in the 0.0.0 coordinates. Or you can simply leave the mesh free without linking it to any node.

NOTE: Do not use this material on a standard mesh without bones. **Avoid using skinned mesh on suspension parts!** For springs and rubber parts use scale animation.

DRIVER POSITION AND MESH

A copy of AC driver with the bones skinned, basic animation of the steer rotation, helmet and some textures, is provided as an example template. It can be placed inside any custom car.



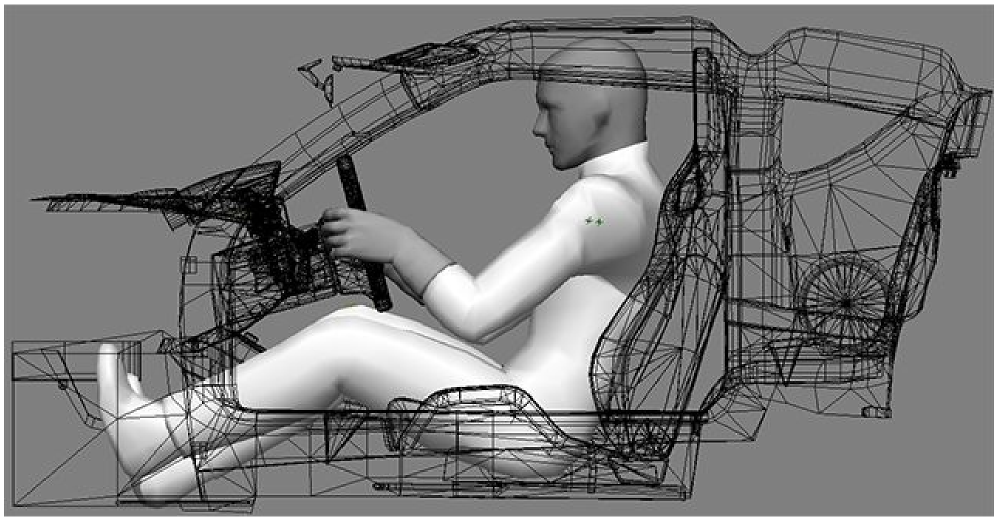
For a proper placement follow these steps:

If you want to use a custom driver mesh go to the section **CUSTOM DRIVER**, otherwise follow these steps:

1) Import the template file DRIVER_BASE.fbx in your 3D application. You should see the driver as in picture. Inside the template, a basic steering wheel rotation animation is provided as an example.
The animation consists of 200 frames.
The neutral position is on frame 100. From neutral (100) to 0, the steering wheel rotates to the left. From neutral (100) to 200, it rotates to the right.

2) place your driver on the seat, with his hands on the steering wheel. Probably some modifications of our animation template will be necessary.

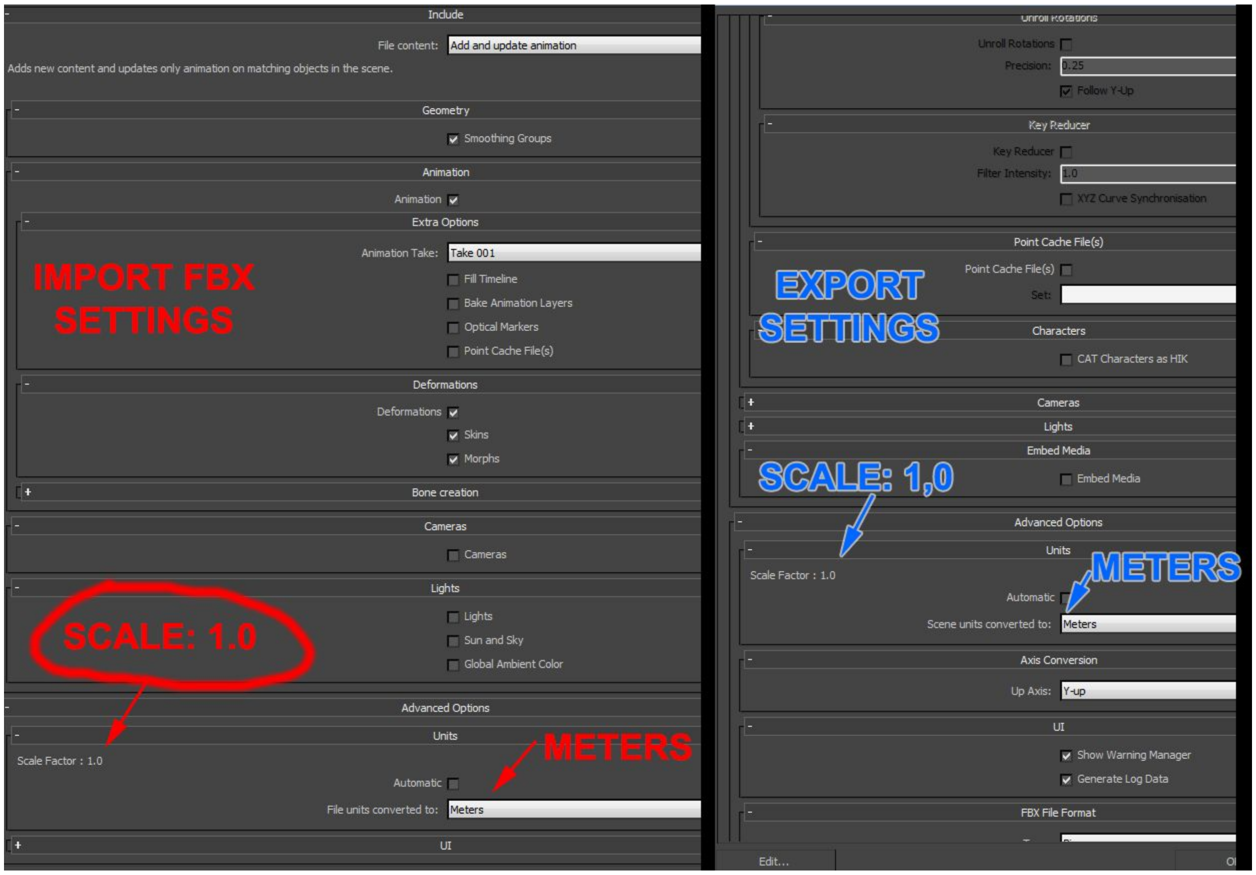
The image here shows an example placement:



The driver mesh and position can now be exported and it will contain the correct hierarchy, and the correct names for the bones and various objects.

IMPORTANT:

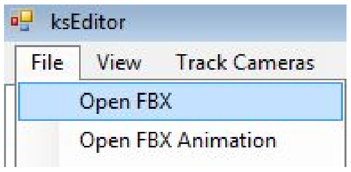
Remember to set the unit in EXPORT (for the provided pilot) to Meters. If not, the editor will produce a weird position of the bones and a wrong result. Keep the same GENERIC UNITs in your 3D software. This is needed because of the original scale of our pilot is 1 and must remain 1 even when exported. For a bone created with a scale of 1 inside 3dsMAX or MAYA, this problem should be not present.



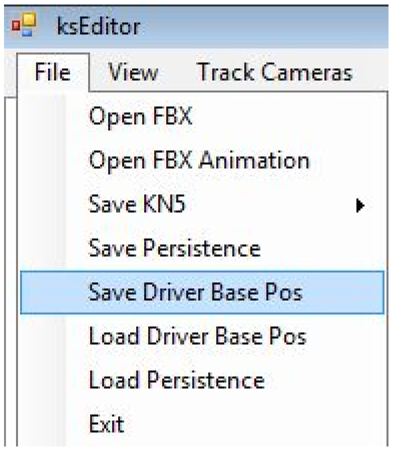
Include
File content: Add and update animation
Adds new content and updates only animation on matching objects in the scene.
Geometry
[✓] Smoothing Groups
Animation
Animation [✓]
Extra Options
Animation Take: Take 001
[ ] Fill Timeline
[ ] Bake Animation Layers
[ ] Optical Markers
[ ] Point Cache File(s)
Deformations
Deformations [✓]
[✓] Skins
[✓] Morphs
Bone creation
Cameras
[ ] Cameras
Lights
[ ] Lights
[ ] Sun and Sky
[ ] Global Ambient Color
Advanced Options
Units
Scale Factor : 1.0
Automatic [ ]
File units converted to: Meters
UI

Unroll Rotations
Unroll Rotations [ ]
Precision: 0.25
[✓] Follow Y-Up
Key Reducer
Key Reducer [ ]
Filter Intensity: 1.0
[ ] XYZ Curve Synchronisation
Point Cache File(s)
Point Cache File(s) [ ]
Set:
Characters
[ ] CAT Characters as HIK
Cameras
Lights
Embed Media
[ ] Embed Media
Advanced Options
Units
Scale Factor : 1.0
Automatic [ ]
Scene units converted to: Meters
Axis Conversion
Up Axis: Y-up
UI
[✓] Show Warning Manager
[✓] Generate Log Data
FBX File Format
Edit...

How to export the driver base position from the editor:



1) Open in the editor the FBX file with the driver placed in the correct base position.

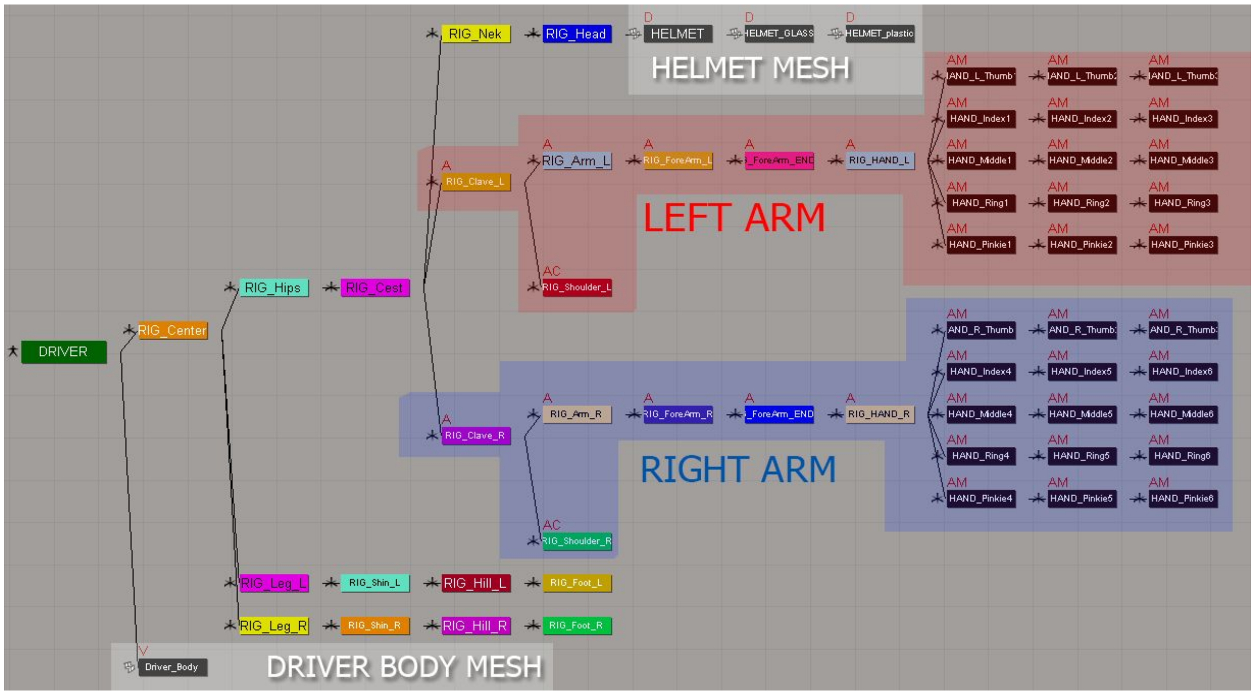


2) Save Driver Base Pos

A file named driver_base_pos.knh is created and stored in the same folder where the source FBX is located.
This file must be placed in the following path: AssettoCorsa/content/cars/CAR-NAME/ where car-name is the car's folder.
The game engine will load the driver and place it using the correct position information stored in the driver_base_pos.knh file.

DRIVER ANIMATIONS

The provided template file DRIVER_BASE.fbx contains a basic example of a 360° steer rotation loop animation.
This animation will probably not match the steering wheel of your car's design. The animation must be modified to match your custom steering wheel dimensions and placement.
Note: The animation must be 200 frames where frame 0, frame 100 and frame 200 match to allow a LOOP animation. For 3DS Max users we have prepared an animation rig that can be downloaded from the Driver animation folder in the Dropbox link.
After editing the animation, save the keyframes of the arms NULLs only and export the FBX with ONLY the animated parts. Animating the pedals is not supported yet. The image below shows the hierarchy:



The bones of the arms are highlighted in the blue and red area in the image, and every bone is parent of the RIG_Clave_L and RIG_Clave_R bones.

To animate the hand that does the shifting, animate the arm bones from RIG_Clave_L/R up to the fingers.
To animate the paddle gear change, animate the fingers only.
For every animation you must export a copy of the driver.fbx with ONLY the animated parts needed for the desired clip. Example: Export driver.fbx with the steering wheel animation only, then another one with gear animation only etc.
Store the driver animations with the names indicated below in the animation folder of your car project folder with all the fbx files and textures.

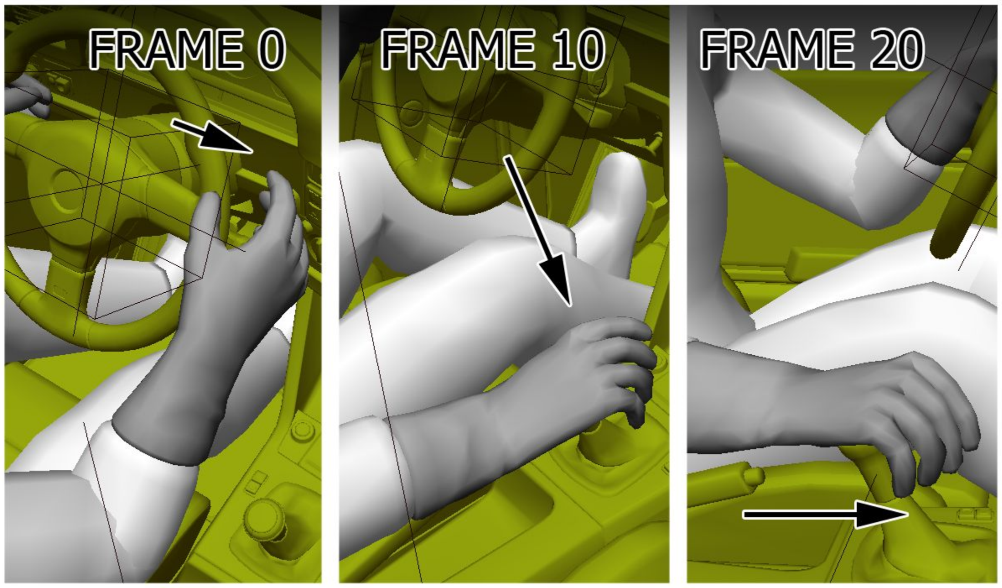
Steer.fbx	for the 360° steer rotation
Shift.fbx	for the gearshift animation

Shift_up.fbx	for the paddle shift up
Shift_dw.fbx	for the paddle shift down

See the section **EXPORT ANIMATIONS FROM THE EDITOR** for instructions on how to create a clip.
Note: Always verify that the car shift animation and the driver shift animation have the same number of frames so that the animation is perfectly synchronized in the game.

Warning: There is typo in the name of the "neck" bone, which is spelt as "nek" by error. Albeit being incorrect, the game still works with this wrong name, so please do NOT correct the typo and keep it "nek".

Example: When the driver changes gear, his arm starts the animation with the hand slightly distant from the steering wheel. (see image below)



On frame 0 the hand is slightly away from the steering wheel. On frame 10 the hand is on the gear lever. On frame 20 the hand moves the gear lever. Be sure that the gear lever animation is synchronized with the hand.
For example, if the hand needs 10 frames to reach the gear level, the gear lever must have 10 frames in the static position before it starts to move.

NOTE: See the following forum thread for a driver rig and explanation to be used in 3DS Max (many thanks for the_meco):
http://www.assettocorsa.net/forum/index.php?threads/custom-steering-animation-rig-1-7.18201/

DRIVER SCRIPTS

The driver is managed by driver3d.ini script in “AssettoCorsa/content/cars/CAR-NAME/data”.
The file structure is the following:

```
[MODEL]
NAME=driver
POSITION=0,0,0
```
- This section determines the model of the driver that will be used (there are different models available)

```
[STEER_ANIMATION]
NAME=steer.ksanim
LOCK=360
```
- This section determines the clip to use for the steering wheel animation and its rotation limit (in this case 360 degrees)

```
[SHIFT_ANIMATION]
BLEND_TIME=200             ; (MS) Time used to move the driver’s hand
                           from the steer position to the first frame of the
                           animation.
POSITIVE_TIME=400          ; (MS) Time needed to move the driver’s hand from the
                           first frame of the animation to the gear lever (forward
                           animation).
STATIC_TIME=10             ; (MS) Interval of time were the driver hand is still
                            on the gear lever (Wait Time between forward and
                           reverse animation)
 NEGATIVE_TIME=400         ; (MS) Time needed to move the driver’s hand from the
                           gear lever back to the first frame of the animation
                            (reverse animation).
PRELOAD_RPM=6000           ; (MS) when the engine reaches this RPM value the
                            forward animation is automatically played

INVERT_SHIFTING_HANDS=0  ; Set to 1 if the driver shifts with the left hand.
[HIDE_OBJECT_0]
NAME=DRIVER:HELMET ; Hide the specific mesh (copy the correct name
                           from the editor) when in cockpit camera. In this case,
                           the helmet is hidden.
[HIDE_OBJECT_1]
NAME=DRIVER:GEO_Driver_FACE    ; Here the driver’s head mesh is hidden.
```

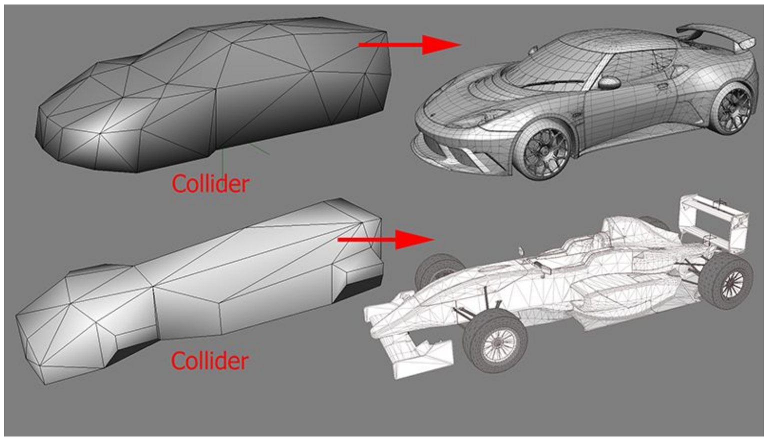
Note: The driver face mesh can have different name if you use a custom driver mesh.

COLLIDER

Collisions between vehicles are one of the most resource-demanding activities of any game, especially if 20 cars collide in a turn at the same time. To optimize such scenarios, a simple collider shape is used to calculate collisions between car bodies and track objects.

The collider shape must be a simple solid object with as low polygon count as possible, without any UV or texture.

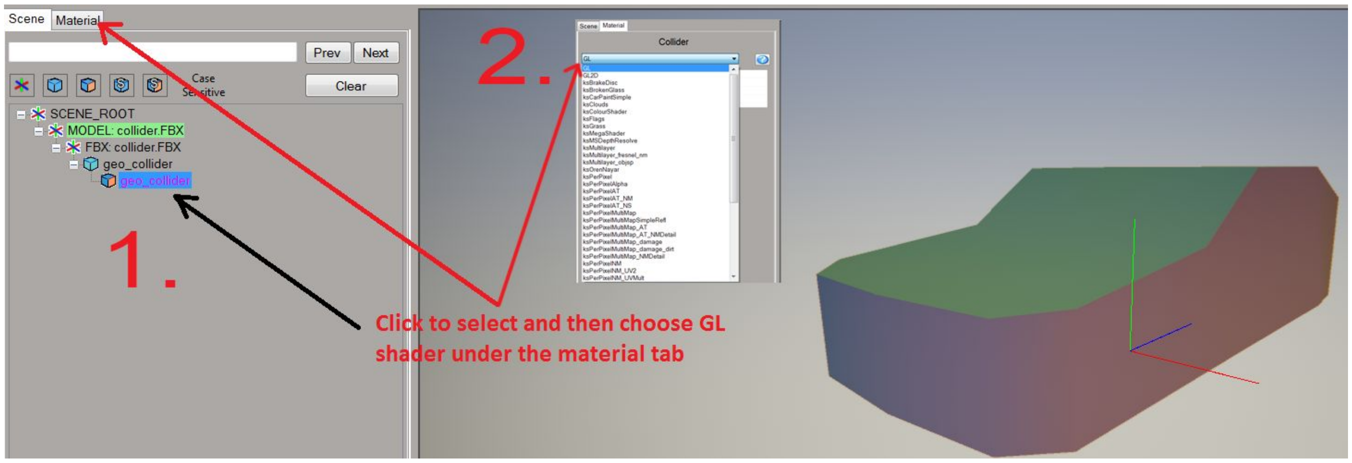
The collider's pivot must be in the 0,0,0 coordinates and have the same orientation as the wheel dummies.



Rules for collider objects:

1) The collider should have no more than 40/60 triangles.

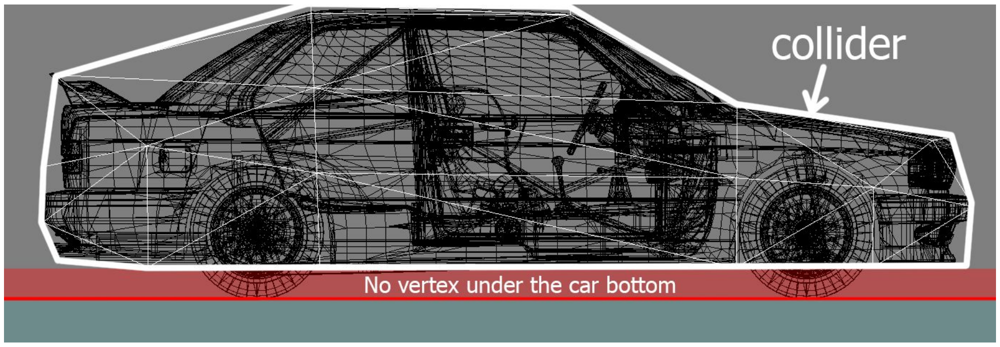
2) A material called "GL" must be assigned to the collider inside the editor. This is a special material specifically made for a mesh that is not rendered. Meshes with this material are used only for collisions.



3) The collider must not extend below the floor of the car.

4) The collider must have no holes. The mesh must be completely closed.

Once the collision mesh is done, simply export the kn5 from the editor, using name "collider.kn5".

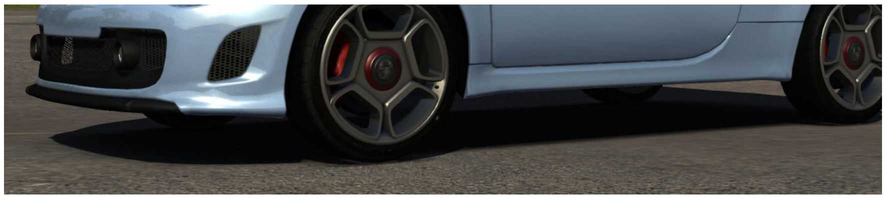


Make sure you save with NO textures! The file must be placed in the same folder as the car LODs with the name collider.kn5.

5. FUNCTIONAL TEXTURES

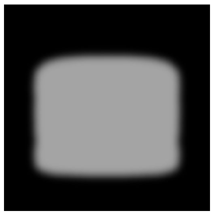
CAR SHADOWS

The car ground shadows are not generated in real time, such as the sun shadows, but they are very important in order to improve the visual effect of the ground position of the car and emulate an ambient occlusion effect on the ground.

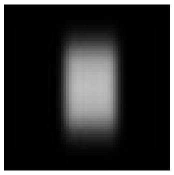


For each car there are five shadow textures. Four textures dedicated to each wheel and another one for the car body.

If not present, the car body texture is automatically generated once in game, otherwise an existing one is used. A pre-made texture is used for the wheels shadows. All shadows must be placed in the root custom car folder (see “Asset Organization”)



Examples:
This is the BODY shadow of the FIAT 500 car. In game it looks as in the image above: The shadow is blended with the dynamic shadow.

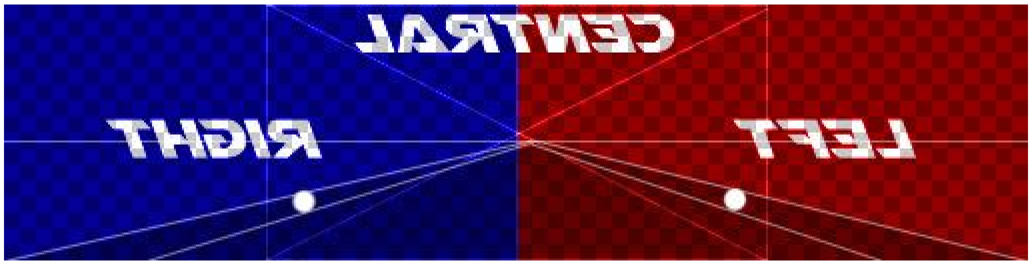


Remember to place in the folder also the 4 wheel textures (see example on the left image). Those work in the same way as the body shadow, but they are attached to the wheels. You can take the automatically generated car body shadow and further refine it in photoshop or your favourite application.

NOTE: The auto-generated shadow of the car is very ROUGH. They must be edited in order to obtain a smoother result.

CAR MIRRORS

In order to make car mirrors work, a material must be created (the name is not important), and assigned to the mirror mesh objects. The mesh must be mapped with the texture called MIRROR_PLACEMENT.

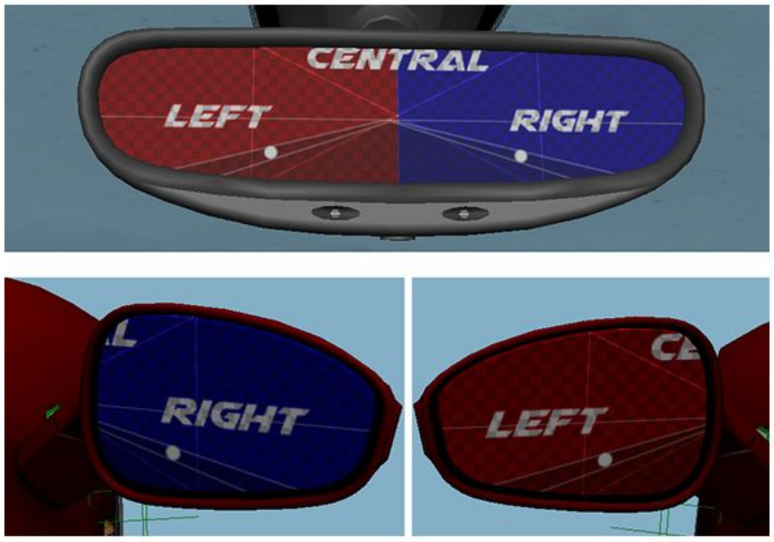


This texture is mirrored and the UV must be mirrored as well to make sure it appears correctly. The texture is divided in three areas. CENTRAL must fit the central internal mirror of the car. The red shows the left, while the blue shows the right hand side mirror.

The 2 points at the center of the lines indicate a point that must be placed in the center of the mirror to make sure that the cars behind can be clearly seen.

Note: remember to keep the correct aspect ratio of mirror UV, otherwise the image of the reflection will be distorted. The image ratio of the MIRROR PLACEMENT template is **4:1**.

Example of the MIRROR_PLACEMENT file, applied to the various mirror mesh objects:



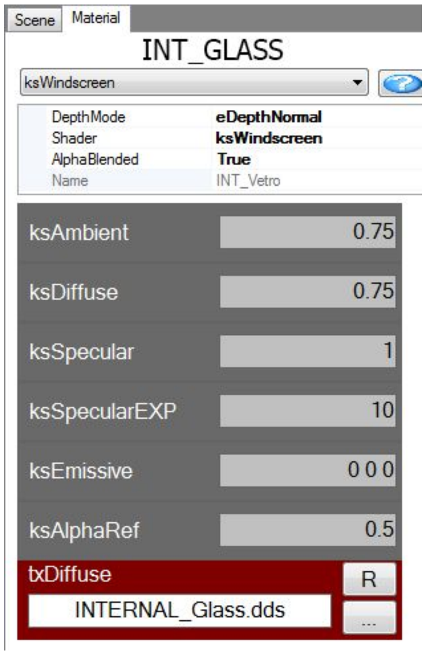
NOTE: Only use the MIRROR_PLACEMENT texture for UV mapping, the actual texture in the editor should be a flat texture named mirror.dds.

INTERNAL WINDSCREEN AND DOOR GLASS REFLECTION

A specific shader is used for the internal of the cars, made specifically to emulate the sun reflection effect when the surface is not completely clear. This effect can be increased or reduced by editing the glass texture provided.

The shader to apply at the internal glass is KsWindscreen

NOTE: the internal GLASS must be not part of the cockpit HR. Internal GLASS objects on the doors must be linked to the appropriate EXTERIOR door nulls. When the cockpit LR switches, the internal glass must remain visible on the LOD A.

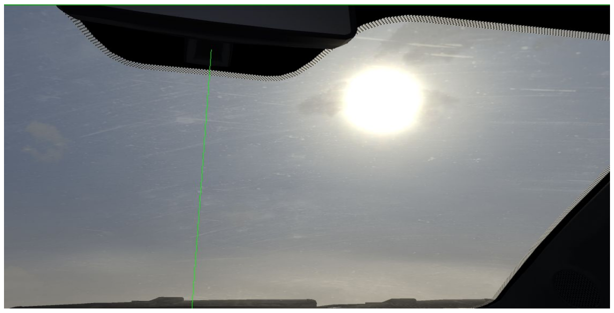


INT_GLASS

ksWindscreen

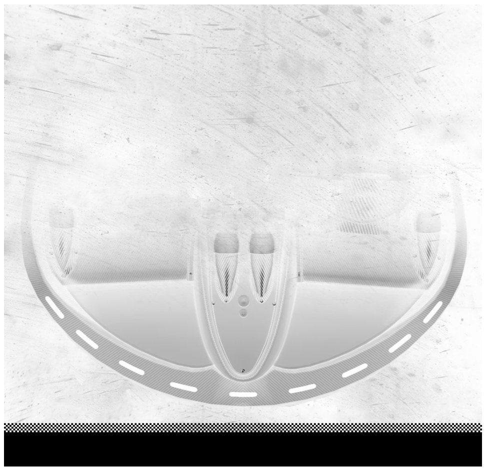
DepthMode	eDepthNormal
Shader	ksWindscreen
AlphaBlended	True
Name	INT_Vetro

ksAmbient	0.75
ksDiffuse	0.75
ksSpecular	1
ksSpecularEXP	10
ksEmissive	0 0 0
ksAlphaRef	0.5
txDiffuse	INTERNAL_Glass.dds

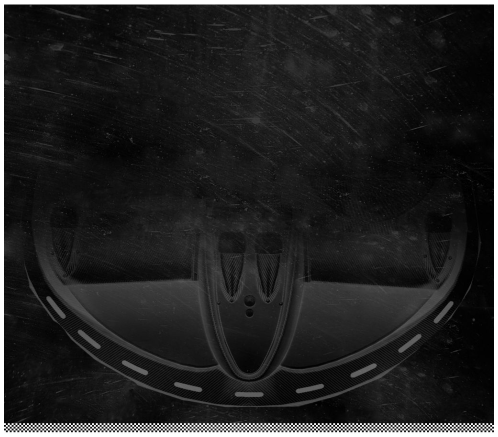


Shader parameters are shown in the image on the right.
An example of the shader effect is shown here to the left.

The texture for the INTERNAL GLASS must be saved in DDS and it must have an alpha channel and the following layout.

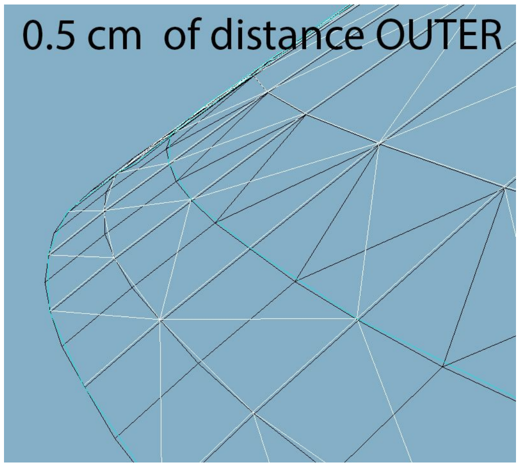


The left image shows the diffuse of the glass texture.
The right image shows the alpha channel.
A soft shadow of the cockpit dashboard is painted on top of the texture.



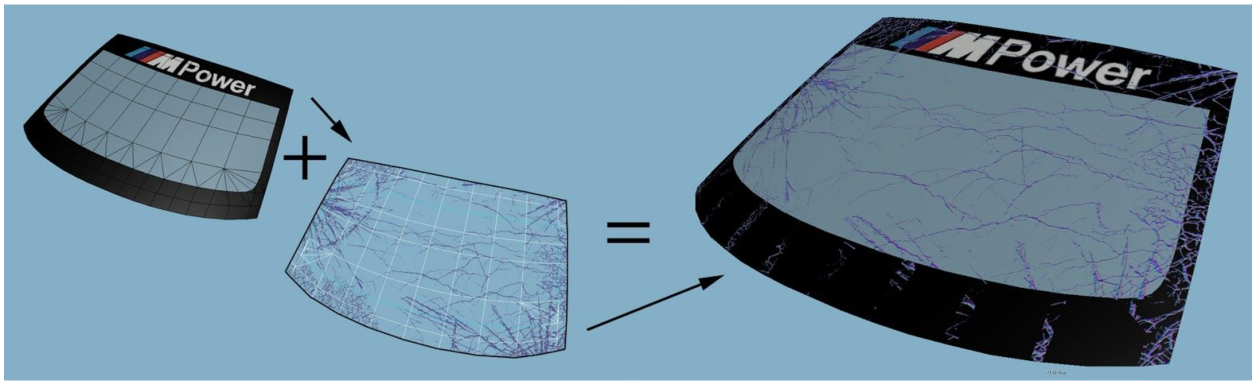
This trick allows an emulation of the internal reflection of the dashboard on the glass when the sun is in front of the car.
Note: The internal glass mesh is just a copy ot the external polygons of the glass, but it should have the normals pointing to the interior. Do likewise for all the internal windows.

DAMAGE GLASS

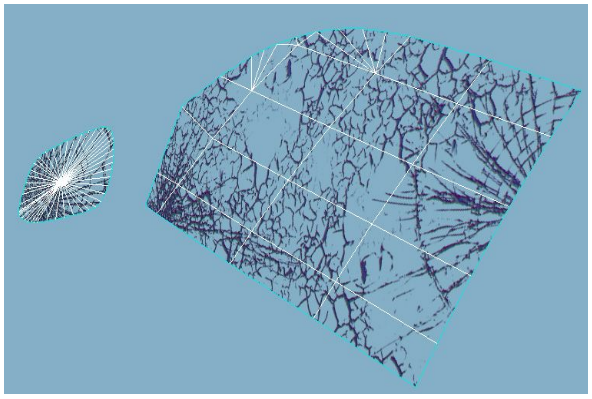


The car can have 2 kinds of damage: damage to glass objects and the body.
For the glass we have to do the following:
Duplicate the glass object, assign to it a new material and map it using the texture
you can find in the **Texture common** folder called **Glass_Crack_00.psd.**

Then, move it away (0.5mm or less) from the original glass to avoid clipping. See here:1
Try to map the glass approximately as shown here (at



least for the front windscreen), because the broken glass must allow the driver to see the road in the game. The cracks must be more visible in the corners and less so in the center (see image above).

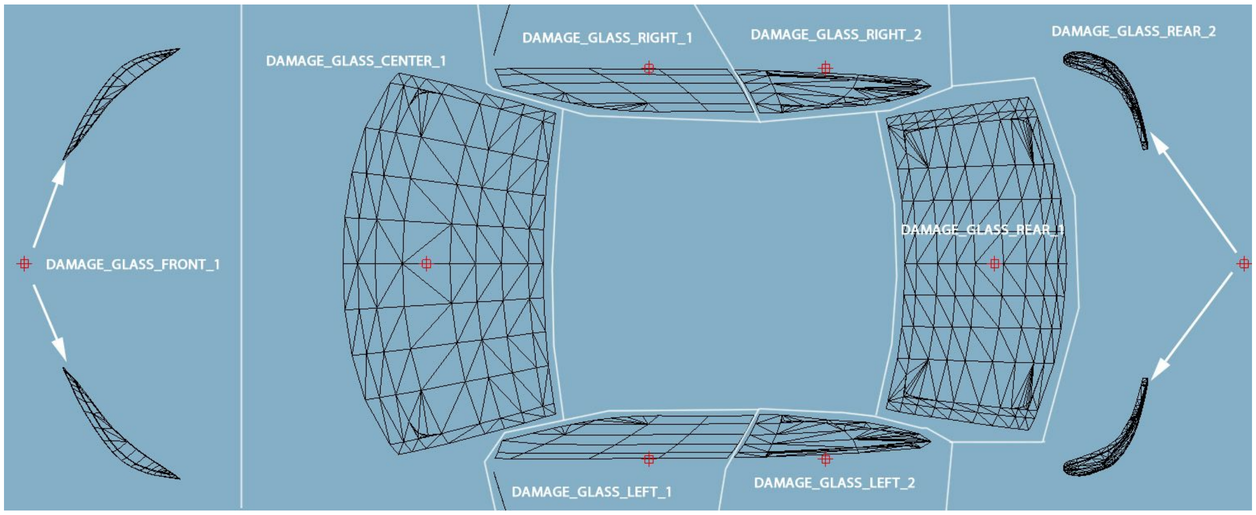


You can map other glass objects on a different area in UV, such as the bottom part. Use the radial or fragmented cracks depending on the shape of the object.

Radial is good for rounded headlight glass, while the fragmented pattern is usually used for square-shaped headlights or side windows. You can see an example for the side glass: Note that we taken also the mirror glass, because it part of the glass objects that can be broken during side impacts.

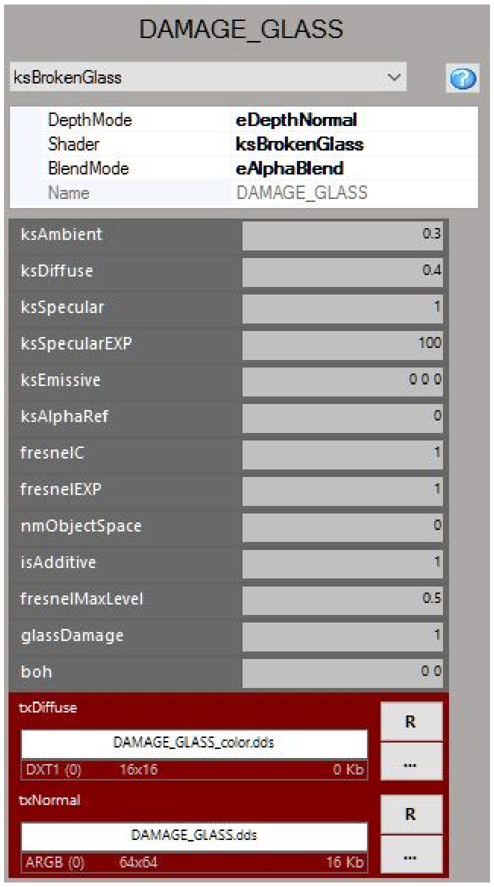
Once you have extracted your glass damage mesh you must place them under the appropriate nulls that use the following naming conventions. **NOTE: the numbers must always be present.**

DAMAGE_GLASS_CENTER_1	central glass, usually windscreen
DAMAGE_GLASS_FRONT_1	front headlight glass or similar
DAMAGE_GLASS_REAR_1	rear/brake light glass or similar
DAMAGE_GLASS_LEFT_1	left side windows of the door and near
DAMAGE_GLASS_RIGHT_1	right side windows of the door and near



Above you can see an example for how to separate damage glass parts.

For the windows you usually have to create more than one object. The same can happen when there are glass objects on the main body as well as on the front bumper. In this case, you can create a new dummy/null and call it **DAMAGE_GLASS_FRONT_2.** Place this dummy/null as the child of the FRONT_BUMPER null to force the broken front bumper light glass to move along with the FRONT_BUMPER object. Do the same for other glass objects located on various moving parts.
You can create as many damage_glass nulls as objects as you need. (for better optimization, use as few as possible….)



DAMAGE_GLASS

ksBrokenGlass

DepthMode	eDepthNormal
Shader	ksBrokenGlass
BlendMode	eAlphaBlend
Name	DAMAGE_GLASS

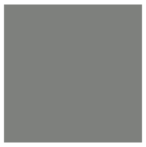
ksAmbient	0.3
ksDiffuse	0.4
ksSpecular	1
ksSpecularEXP	100
ksEmissive	0 0 0
ksAlphaRef	0
fresnelC	1
fresnelEXP	1
nmObjectSpace	0
isAdditive	1
fresnelMaxLevel	0.5
glassDamage	1
boh	0 0

txDiffuse
DAMAGE_GLASS_color.dds
DXT1 (0) 16x16 0 Kb

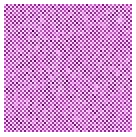
txNormal
DAMAGE_GLASS.dds
ARGB (0) 64x64 16 Kb

Once you have created, UV mapped and linked the objects, you must assign the proper material with the parameter indicated in the image on the left.
Every object of damage glass must be set **TRANSPARENT** under the object settings and must not cast shadows.
As the diffuse and normal map we must apply a PROXY TEXTURE. The Proxy texture is a placeholder texture that substitutes the texture that the game loads automatically from a common folder.

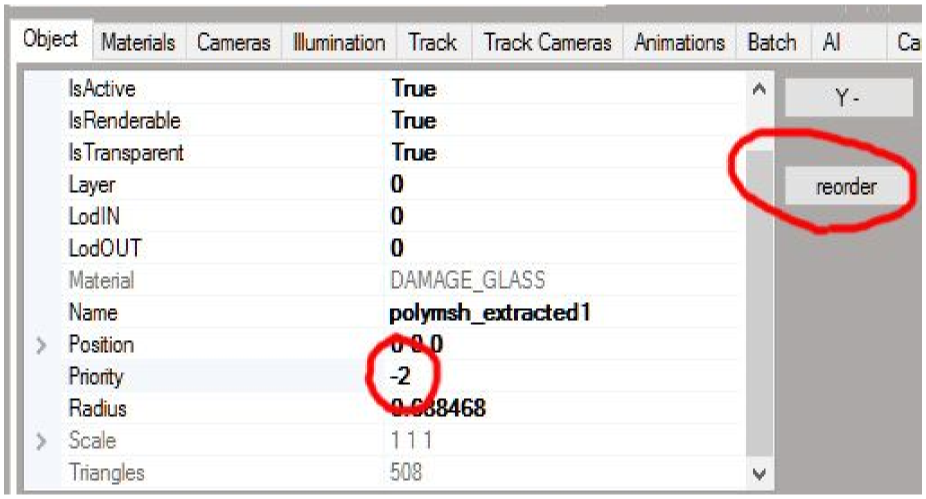
For txDIFFUSE use the DDS named DAMAGE_GLASS_color.dds in the **Common Texture** folder



For txNORMAL use the DDS named DAMAGE_GLASS.dds in the **Common Texture** folder



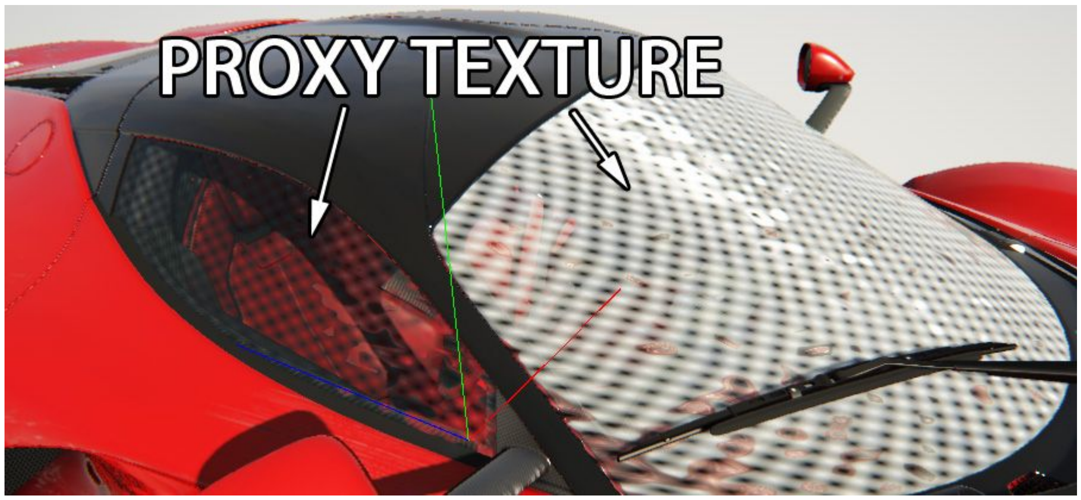
After you have set up the material, you must change the **draw priority**. Select every DAMAGE_GLASS **dummy/null** (not the object!) and set the priority to **-2** and press **REORDER.**



IsActive	True
IsRenderable	True
IsTransparent	True
Layer	0
LodIN	0
LodOUT	0
Material	DAMAGE_GLASS
Name	polymsh_extracted1
Position	0 0 0
Priority	-2
Radius	0.[illegible]88468
Scale	1 1 1
Triangles	508

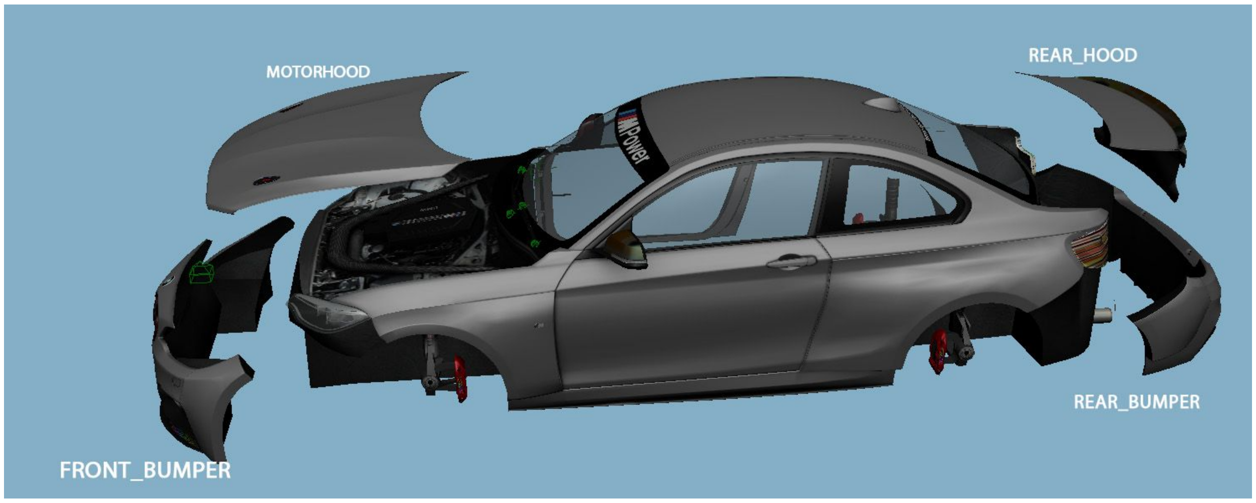
You can check if the damage glass works correctly by pressing the appropriate button in the editor. The shortcut to see DAMAGE GLASS in the editor is **F4.**

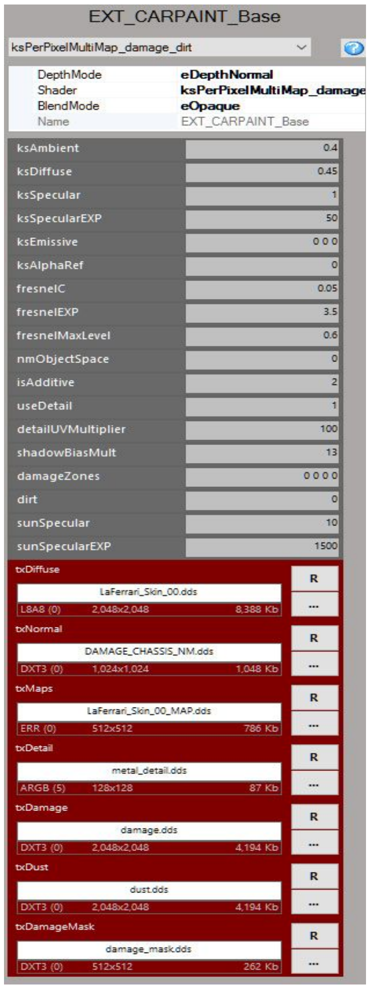
In the editor you should see the DAMAGE glass as in the image below:



For naming the mesh objects, use a naming convention that is easy to follow, such as MESH_DAMAGE_GLASS_FRONT_1 etc.

CAR DAMAGE




EXT_CARPAINT_Base

ksPerPixelMultiMap_damage_dirt

DepthMode	eDepthNormal
Shader	ksPerPixelMultiMap_damage
BlendMode	eOpaque
Name	EXT_CARPAINT_Base

ksAmbient	0.4
ksDiffuse	0.45
ksSpecular	1
ksSpecularEXP	50
ksEmissive	0 0 0
ksAlphaRef	0
fresnelC	0.05
fresnelEXP	3.5
fresnelMaxLevel	0.6
nmObjectSpace	0
isAdditive	2
useDetail	1
detailUVMultiplier	100
shadowBiasMult	13
damageZones	0 0 0 0
dirt	0
sunSpecular	10
sunSpecularEXP	1500

txDiffuse	LaFerrari_Skin_00.dds	L8A8 (0)	2,048x2,048	8,388 Kb
txNormal	DAMAGE_CHASSIS_NM.dds	DXT3 (0)	1,024x1,024	1,048 Kb
txMaps	LaFerrari_Skin_00_MAP.dds	ERR (0)	512x512	786 Kb
txDetail	metal_detail.dds	ARGB (5)	128x128	87 Kb
txDamage	damage.dds	DXT3 (0)	2,048x2,048	4,194 Kb
txDust	dust.dds	DXT3 (0)	2,048x2,048	4,194 Kb
txDamageMask	damage_mask.dds	DXT3 (0)	512x512	262 Kb

Certain body parts must be detached and placed under a Dummy/null that acts like a pivot/center of rotation for the element when it receives damage. Upon impact, a script is activated that makes the parts vibrate/rotate on the basis of the location and pivot of these nulls.

Keep the nulls and also these parts separate in LOD B. In LOD C, the elements can be attached to the main body and they do not have to be movable. If the damaged parts are significant in size (massive front and rear wings on formula cars, you can keep the most important items on the LOD C to make sure there is no visible LOD switch when the car is damaged.

Use the following guidelines:

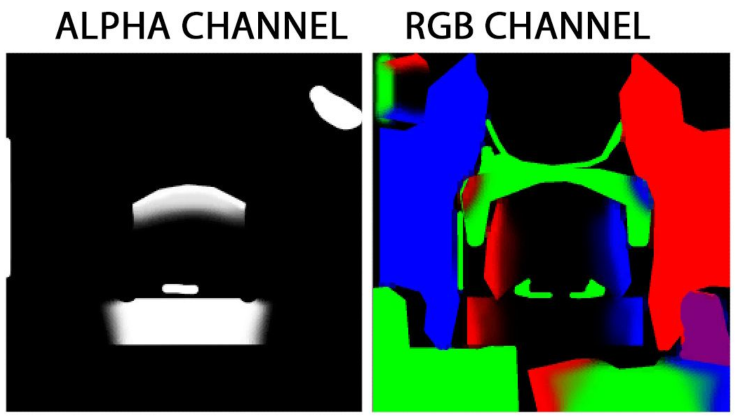
1) Make sure that you have closed the mesh in the interior. You put a black texture or something very dark to ensure that there is no gaping hole behind the moving objects.
2) Place the dummy/null in the rotation point that is logical for the part. For the MOTORHOOD it can be the hinges, for a FRONT_BUMPER it can be a point that allows rotation but avoiding any intersection with the main body mesh.
3) Detach parts only that don't leave holes in the car when moving, or carefully cap the holes.

Parts that take damage may be:
Front Bumper, Rear Bumper, Front and Rear Hood, Exhaust, Wing and the Extractor (diffuser) on various GT cars etc. It depends on the model at hand.

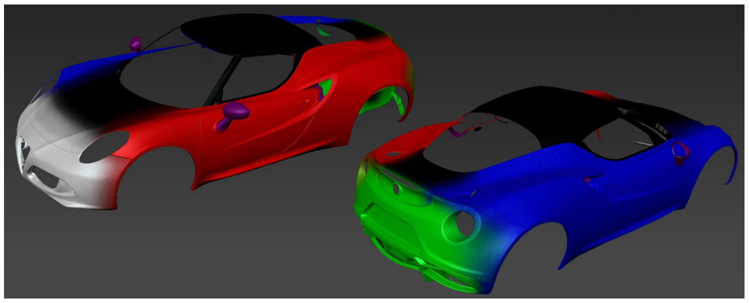
After the parts are **done**, you must set up the material properly and edit a script.

Damage needs 4 different textures to work properly:
The damage feature to work properly you need the following textures: DAMAGE_NORMAL map, DAMAGE_SCRATCHES map, a DUST map and a DAMAGE_MASK map.

The **DAMAGE MASK** must be called **DAMAGE_Mask.dds** and must be created in the following way:

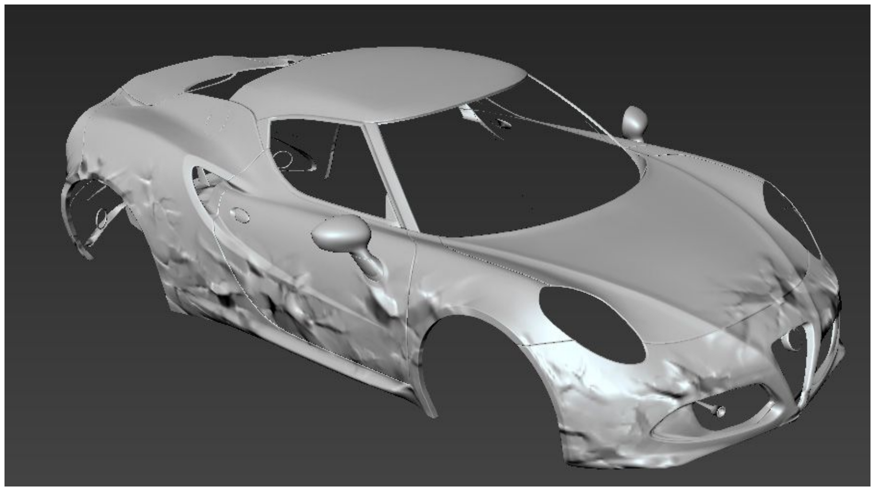


The **WHITE** part indicates the front of the car (painted in the alpha channel), the **RED** the left-hand side, the **BLUE** the right-hand side, and the **GREEN** the rear of the car. We use this mask to control which areas are affected by the damage.
The mask must be painted as shown here:



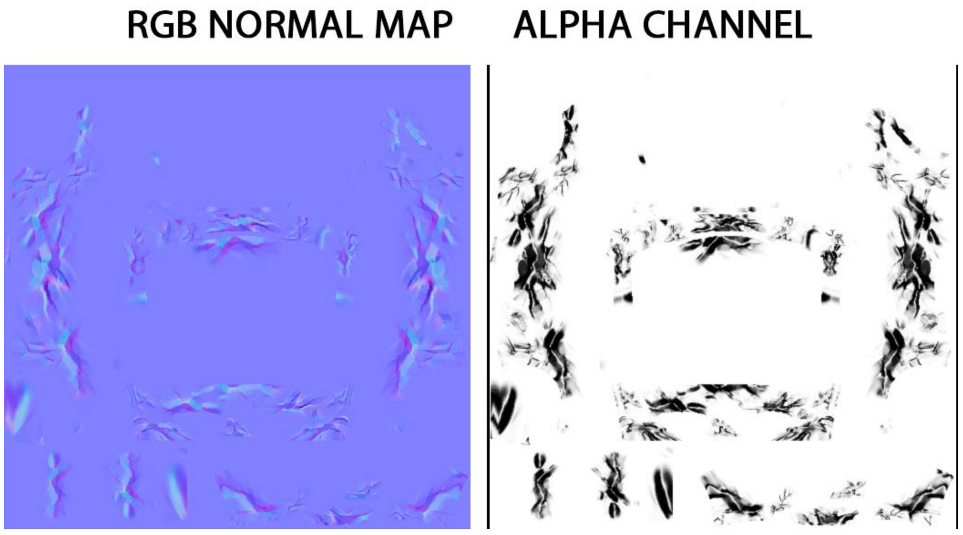
Remember to blend the colors, do **NOT** create sharp transitions. Never paint the roof and the top of the bonnet.

Assign this texture to the slot **txDamageMask**
Resolution must be **512x512** pixels and the texture can be exported as DXT5.



For the **NORMAL MAP** texture we must create a normal map with the metal deformation as shown in the example.

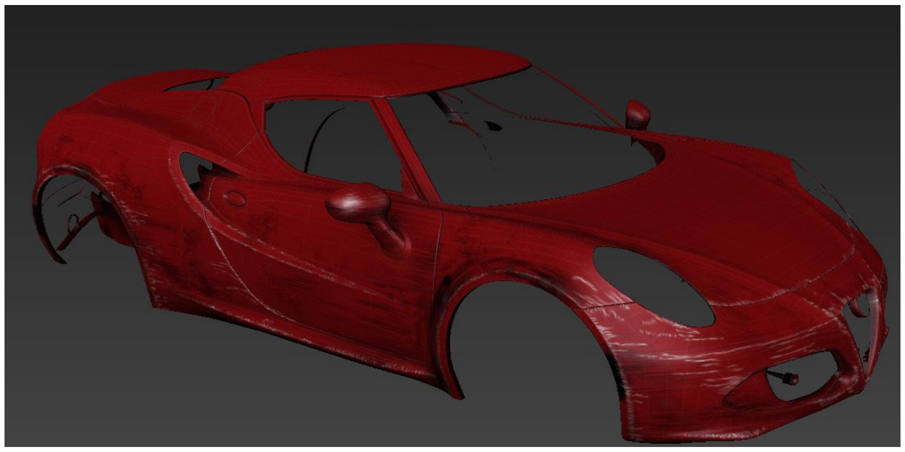
You can use your preferred tool, such as Mudbox, Zbrush, or anything else you're familiar with.



Look at the example: in the alpha channel you must paint the part of the chassis with scratches, which becomes less reflective, to visualize better the wrecked appearance.

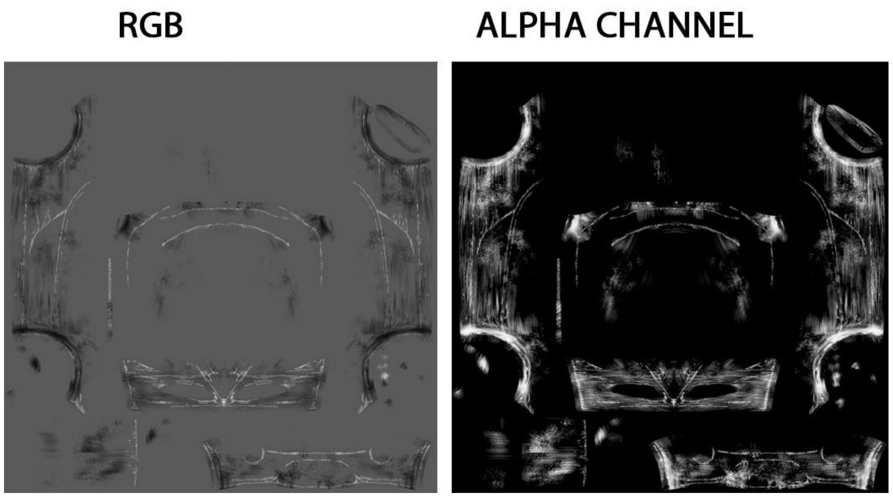
This texture must be assigned to the slot called **txNormal.**

Texture must be done in DXT5 and size can be **512X512** pixels.



For the **DAMAGE SCRATCHES** texture you must paint a texture that, working along with the normal map, shows scratches and damage on the surface. Have a look at example image. The scratches appear in front of a red background here but in the texture use a grey background shown in the texture example below.

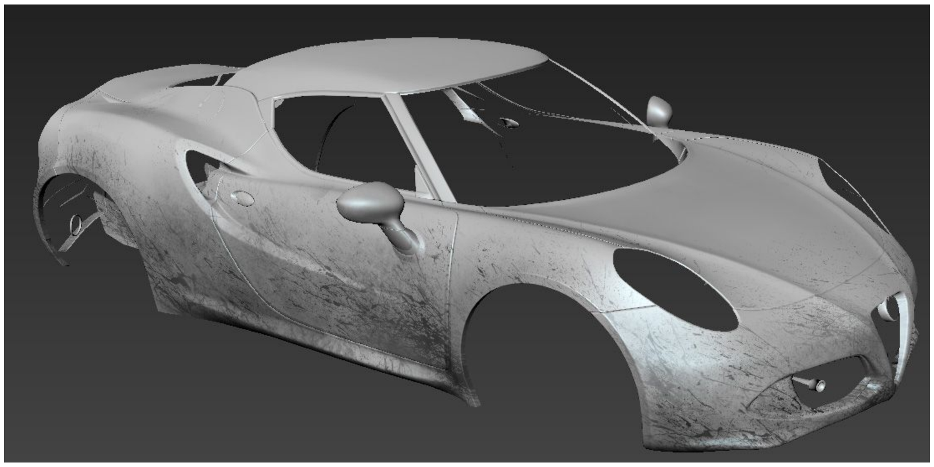
The scratches on the edges must have a highlight and they must be visible on the sides, too. Scratches appear over the car paint texture so the texture needs an alpha channel to use as a mask for opacity.



This texture must be assigned to the slot called **txDamage.**

The texture must be exported as DXT5 and the size must be **2048x2048** pixels.

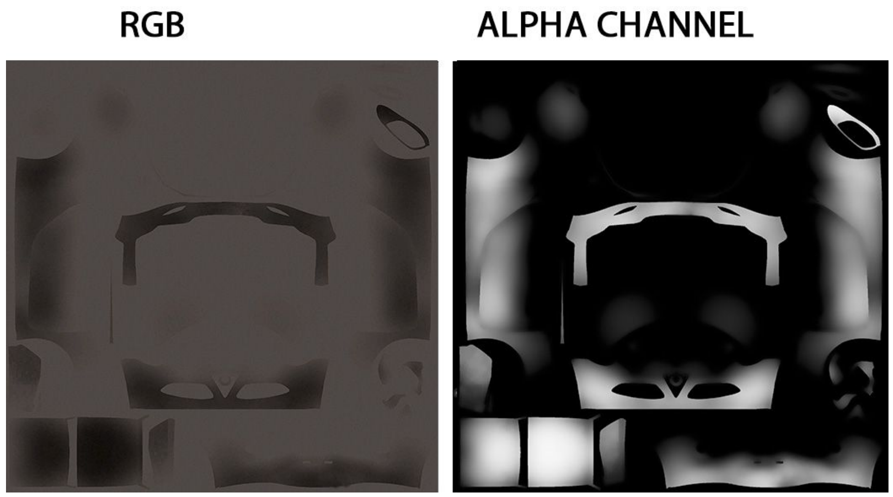
The last texture is the **DUST** texture that shows dirt on the car after driving off-road.



We must draw a dust layer to visualize dry dust.

Texture must be done in DXT5 and size must be **2048x2048** pixels or 1024x1024 if the car is heavy on textures.

This texture must be placed in the slot called **txDust.**

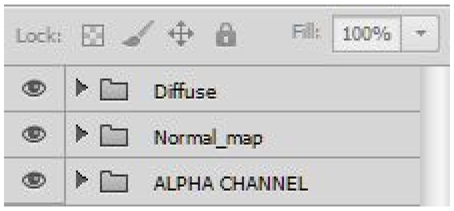


To visualize the global effect of the damage you can use the show damage button in editor:

6. TEXTURING GUIDELINES

The supported texture format is: **directX DDS**
This format can be outputted from Photoshop (for example) using the specific nVidia plugin, available here: https://developer.nvidia.com/legacy-texture-tools

As a general rule, we recommend using the DXT5 format with high-resolution textures (with or without Alpha) and 8.8.8.8 format for textures with sensitive gradient information (RGB maps, detailed normal maps) or small-size detail textures and 8.8.8.8 where Alpha information is included.
We need for every texture a PSD source with layers inside. The layers must be placed inside layer folders with consistent and user-friendly names.



Inside every folder we need a base layer that allows us to change important features of the texture. Follow these rules:

a) If the texture has an **ALPHA CHANNEL,** do not collapse transparent features, keep the transparent features in a specific layer.
b) If there is a normal map, provide in the layer also the greyscale texture so that it can be re-generated with the nVidia tool
c) ALWAYS work with **DOUBLE resolution** (no more no less) of the target image and shrink it to the right size only when you export the DDS. Test your results to be sure that the reduction does not spoil the image too much (this could happen with tiny texts or symbols).
d) All PSD files must be in **RGB Color 8 bit for channel** mode.
e) Name them correctly following our naming **conventions**.

TEXTURE NAMING CONVENTIONS FOR PSD SOURCE FILES

Skin.PSD contains the main body textures
Ambient occlusion
Wireframe (UV)
RGB Map (material specular-gloss-ref map)
Material IDs and zones
Alpha channel

Ext_Details.PSD contains rivets, bolts, logos, and decals on the exterior
Diffuse
Normal map
Ambient occlusion
RGB Map (material specular-gloss-ref map)
Alpha channel

Rims.PSD contains rim base and rim blur texture plus the blurred spokes
Diffuse
Normal map
Ambient occlusion

RGB Map (material specular-gloss-ref map)
Alpha channel

Calipers.psd contains the brake caliper texture
Diffuse
Normal map
Ambient occlusion
RGB Map (material specular-gloss-ref map)
Alpha channel

Lights.psd contains the light texture
Diffuse
Normal map
Ambient occlusion
RGB Map (material specular-gloss-ref map)
Alpha channel

Mechanics.psd contains the underside, engine and all the parts that are not included in the skin
Wireframe (UV)
Diffuse
Normal map
Ambient occlusion
RGB Map (material specular-gloss-ref map)
Alpha channel

Glass.psd contains the glass texture and all similar parts such as black frame
Diffuse
Normal map
Alpha channel

Grids.psd contains tileable grids and similar textures (use more if needed)
Diffuse
Normal map
Ambient occlusion
RGB Map (material specular-gloss-ref map)
Alpha channel

Tyre.psd contains tyre textures with blur and dirt
Wireframe (UV)
Diffuse
Normal map
Ambient occlusion
Alpha channel

Disc.psd contains the brake disc texture and the glow texture
Wireframe (UV)
Diffuse
RGB Normal map

Glow map

Windscreen.psd contains the fake internal glass reflection
Diffuse
Alpha channel

INT_Decals.psd contains dials, dashboard symbols, cockpit details and logos, plates and interior bolts and stickers
Diffuse
Normal map
Ambient occlusion
RGB Map (material specular-gloss-ref map)
Alpha channel

INT_Details.psd contains coloured gradients and other details to use for smaller objects
Diffuse
Normal map
Ambient occlusion
Alpha channel
INT_Occlusion.psd contains the cockpit ambient occlusion texture
Wire frame stamp
Diffuse
Normal map
Ambient occlusion
RGB Map (material specular-gloss-ref map)
Alpha channel

Belts.PSD contains cockpit belts
Diffuse
Normal map

Seams.psd contains stitching, seams, and similar textures in **tileable** form
Diffuse
Normal map

INT_cockpit_LR.psd contains cockpit LOW RESOLUTION texture
Ambient occlusion
Wire frame stamp
RGB Map (material specular-gloss-ref map)
Material ID and zones
Alpha channel

All the extra textures that can occur and are not mentioned here can have a name that explains in brief what they contain. To see how to manage textures you can see examples in the example folder in the SDK.

EXPORTING TEXTURES AND OPTIMISATION

The carname_lod_A.kn5 file of an official car must stay below 44MB, including all textures and mesh. The textures have to be very well optimized. When the PC runs out of memory, the game engine starts to reduce texture size automatically, however, this is done in a way that does not ensure high quality, so we have to avoid it in all cases and stay below the 44MB limit for the LOD A.kn5 file.

NOTE: As a general guideline, you can use the DXT5 compression for high-resolution textures. Use AL (8.8, alpha luminance) mode for grayscale textures with sensitive gradients. Use RGB (8.8.8) mode for map textures and ARGB (8.8.8.8) for NM textures with fine details. Remember to keep a complete set of ALL of your textures **without compression** as a **backup** so that if they have to be outputted again, there is no quality loss due to the added compression.

NEVER use the DXT1 compression mode. Keep the PSD files organized and updated so that they can be used to re-output textures if a change is necessary after the delivery of the model. Do not work on compressed DDS textures, always make your changes in the PSD and keep it updated along with the exported textures so that t**he latest version of each PSD file corresponds to the latest DDS output**.

Below you can see some examples for texture size. Taking into consideration priorities to maintain a high-quality look, you can use larger textures provided that you optimize other textures better and you do not go over the limit:

Skin_00.dds (the main body) must be 2048x2048 when it have sponsor and livery on. If is flat, can be 1024x1024 and saved as 8.8. **Skin_00_map.dss** 512x512 ARGB

Rim.dds 512x512 It can contain a base material for rim blur non-transparent parts. **Rim_map.dds** is half of the rim size and saved as ARGB. **Rim_Spokes.dds** 256x256

INT_Occlusion.dds 512x512 and **INT_Occlusion_map .dds** 512x512 saved as 8.8.

INT_Cockpit_LR.dds 512x512 or 1024x1024 depending from the car roof if open or close. DXT5 is enough.

INT_Decals.dds 1024x512 DXT5 - **INT_Decals_NM.dds** 1024x512 in DXT5 or ARGB.

Lights.dds 512x512 ARGB - **Lights_NM.dds** 512x512 ARGB - **Lights_Map.dds** 256x256 ARGB

Grids tileable and various similar 256x256 or also half, depending on the image detail, export as ARGB.

Tyre_D.dds and **Tyre_NM.dds** 1024x1024 DXT5
Tyre_blur_D.dds and **Tyre_blur_NM.dds** can be 512x512 or 256x256, save NM as ARGB.

Disc_D.dds and **Disc_NM.dds** can be 512x512 DXT5 (D) ARGB (NM) when very visible and half when it is small and not very visible or when there are no details. **Disc_Blur_NM.dds** and **Disc_Blur_NM.dds** are half of the non-blurred disc textures. **Disc_warm.dds** is always 128x128.

INT_Materials_D.dds and **INT_Materials_NM.dds** 512x512 or less, depending from image content. **INT_Materials_map .dds** is half of base texture, save all as ARGB (especially NM and RGB map) to keep the quality of the gradients.

Damage.dds 2048x2048 - **Damage_NM.dds** is 512x512 - **Damage_Mask.dd**s is 256x256 - **Dust.dds** is 1024x1024 DXT5.

Stiching_D.dds and **Stiching_NM.dds** can be 256x128 vertically tileable, save as ARGB.

Belt_D.dds and **Belt_NM.dds** can be 128x256 ARGB and must be vertically tileable.

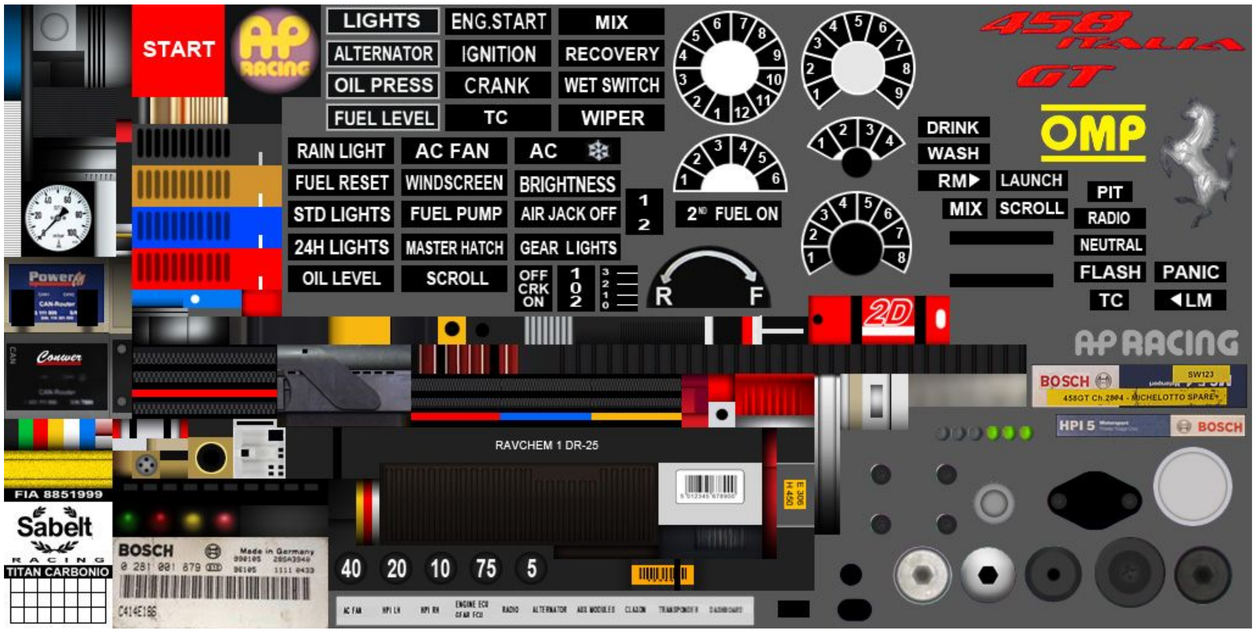
Mechanics_D.dds and **Mechanics_NM.dds** can be 1024x1024 if contain a visible engine. If not, it can be half. **Mechanics_map.dds** is always half of the diffuse one. Diffuse DXT5, NM ARGB, map RGB.

Calipers.dds and **Calipers_NM.dds** can be 256x256 or in some cases can be part of the Mechanics textures if your car is a ‘60s open seater racing car. **Calipers_map.dds** is always half the size of the diffuse. DXT5 for diffuse, ARGB for NM and RGB for map texture.

OPTIMAL USE OF TEXTURE SPACE

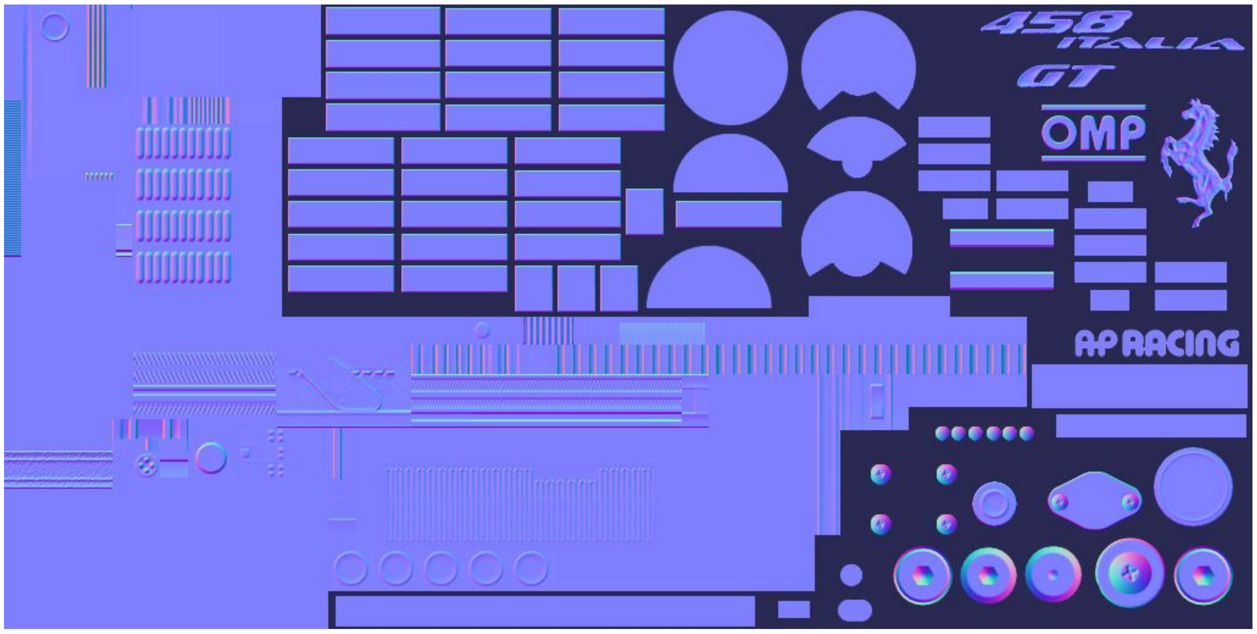
When you use your space in the texture you must make sure to optimize everything the best you can. Maximum means that all available space must be used. You have to plan before you start to make sure that you use your texture space in the most efficient way.

An example for the Decals_D texture with good use of space:

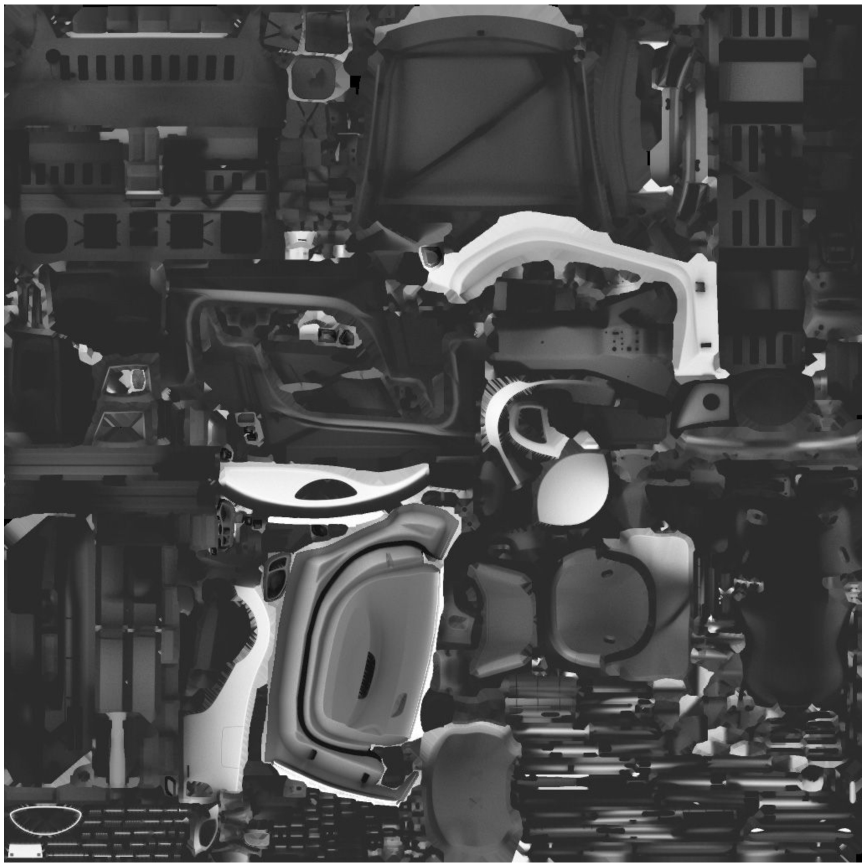


Include the alpha channel both in the diffuse and the NORMAL MAP to make sure it suits every shader type.

In the following texture you can see the normal map texture with the alpha channel visible. The uncompressed alpha channel defines the outline of the details.



Look at the following examples for the occlusion or the car skin textures to see how to optimize the available texture space:



The parts use the maximum space available and and the padding (extension borders) fill up the remaining space.

This arrangement allows us to reduce the texture to as low as 512x512 (uncompressed) but keep the occlusion gradients at an acceptable level of quality.

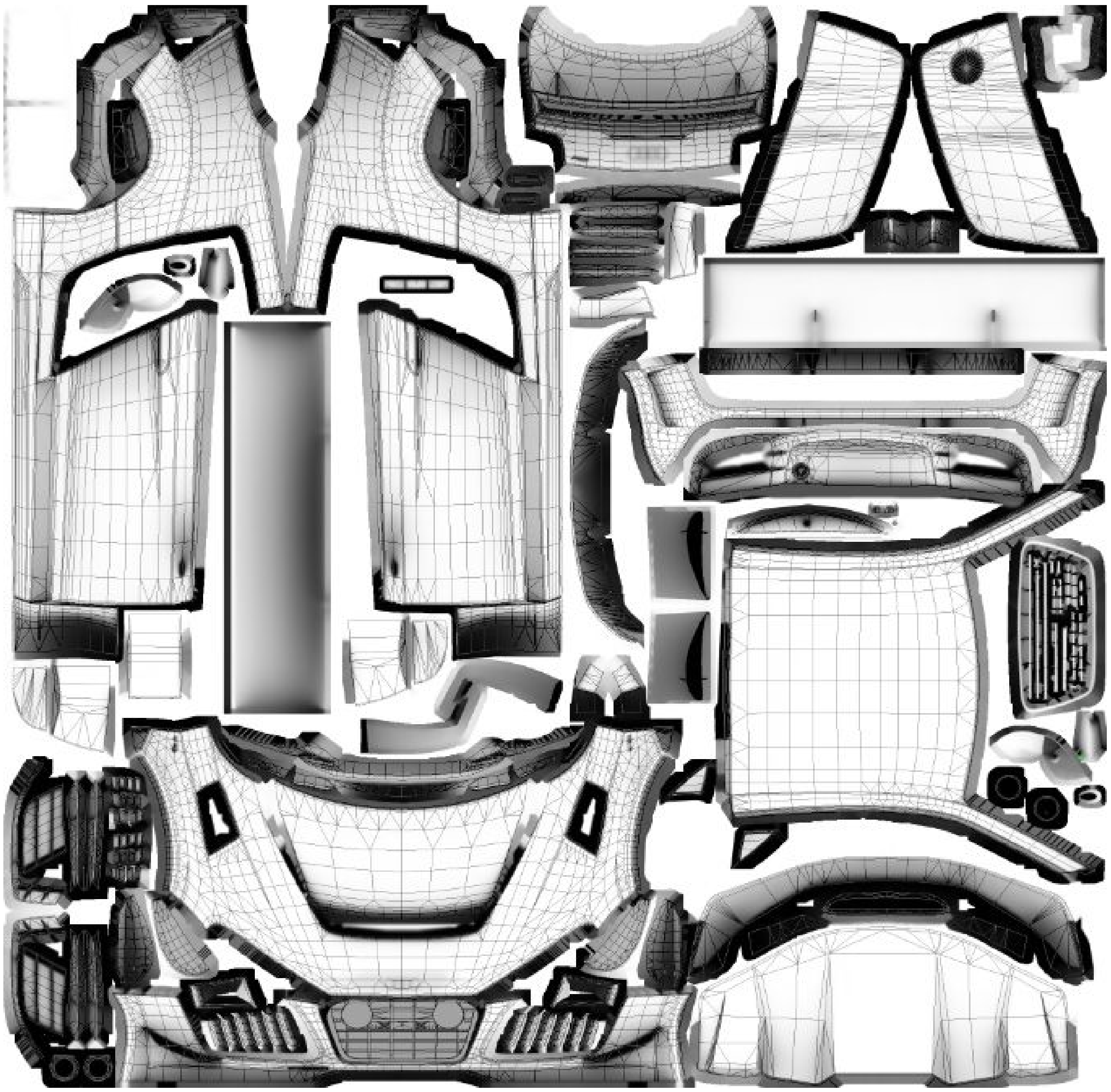
It is recommended that all interior objects with an AO map be mapped on a single texture.

The same material groups must use the same **scaling** to make sure the detail textures appear correctly.

NOTE: You also have to make sure that in the UV map the different UV parts are using the same scale to make sure that any detail texture (metal flakes or carbon) appear correctly without any stretching and distortion!

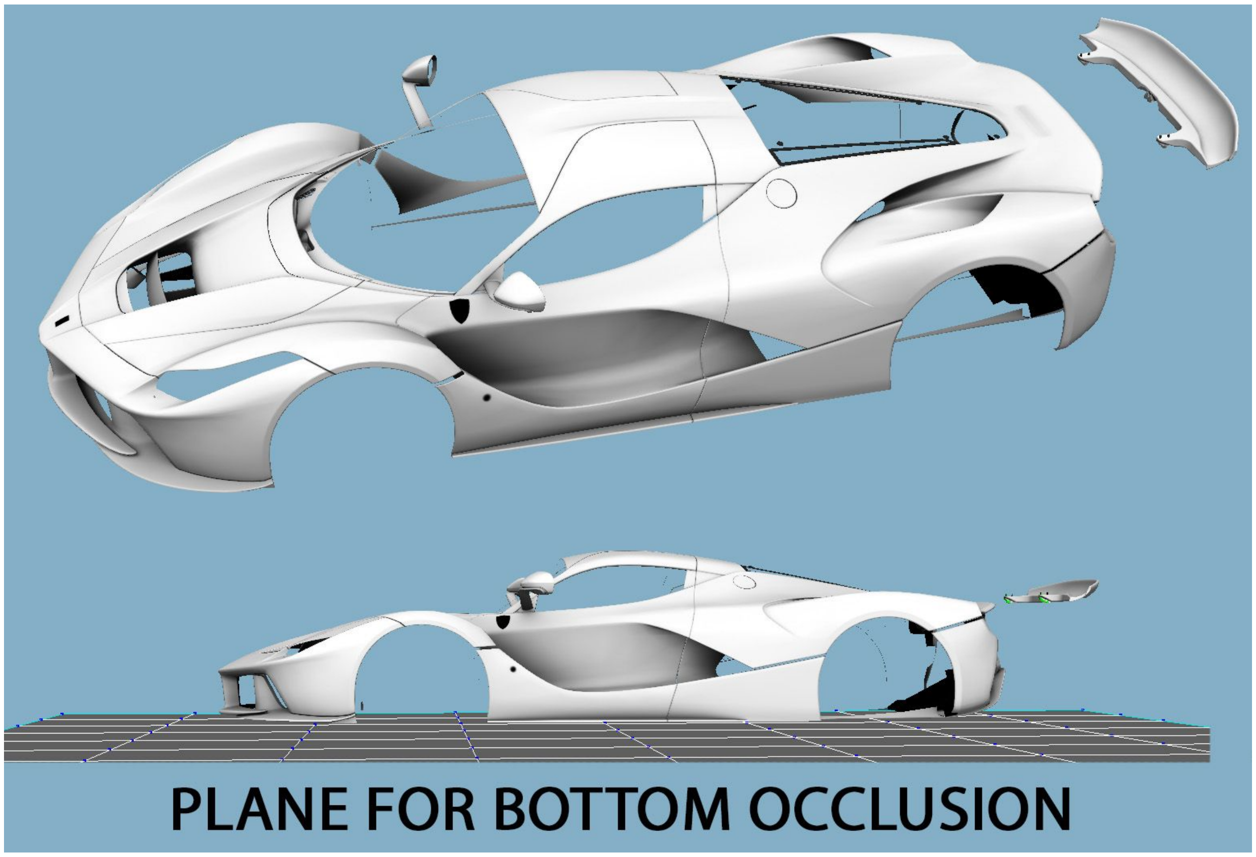
It is recommended that you use a checkered detail texture for mapping the body and interior textures that use detail textures.

Also, and especially on the exterior, textures must be well organized. Look the second example:



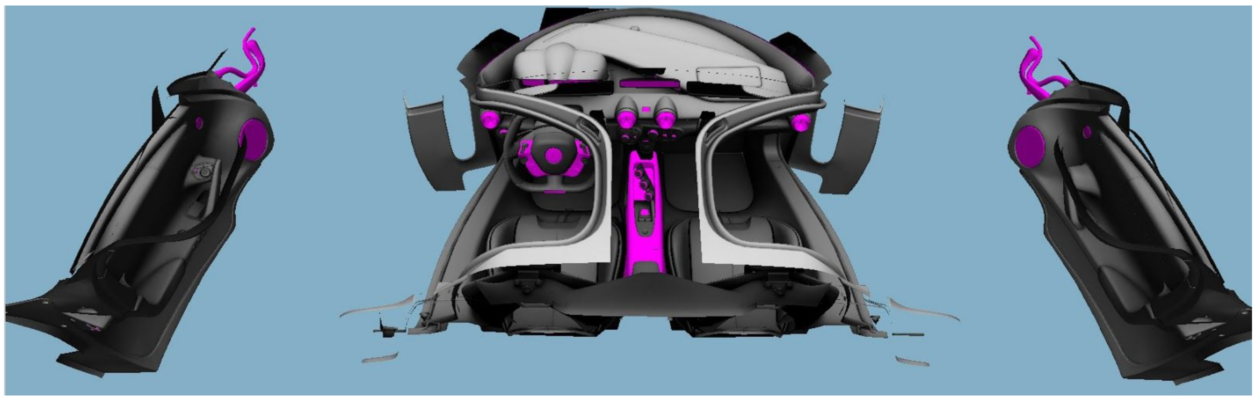
BAKING THE AMBIENT OCCLUSION

To have a more realistic illumination effect, we need to bake the ambient occlusion map for the exterior of the car, the rims, the lights and the interior of the cockpit.

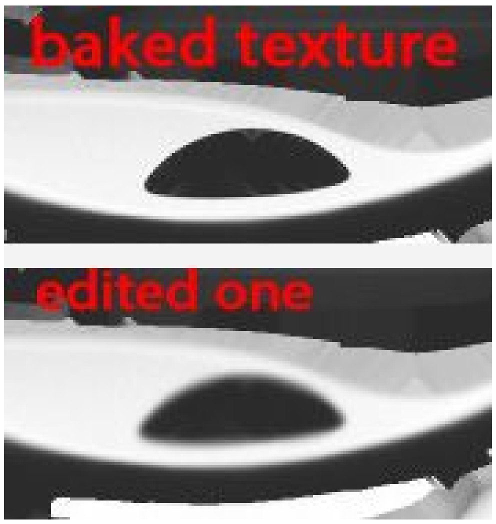


Take the exterior, and **remove all the DECALS objects**. If you have a movable wing move it a bit far from the body. Bake the Ambient Occlusion at double resolution (4096x4096).

For the cockpit: remove all the DECALS for logos and the stitching. For baking the interior place the doors like in the image. For baking the steering wheel, remove everything else and bake it facing **UP**.
NOTE: look the pink pieces, they don't take occlusion, but they influence the cockpit for them. They will be placed under a different material.



Keep the doors far enough to avoid a dark occlusion on the borders and the doorsill.
A baked texture is never how we want it in the end. We suggest to edit it in Photoshop and create softer intersections with objects. Random pixels can create a bad effect when they are in a visible place.



Use Photoshop to make the transitions smoother where necessary. The AO textures are globally a kind of soft gradients. Avoid sharp, pixelated and unclean transitions.

IMPORTANT: when baking make sure you use a wide-enough padding to avoid bleeding black artifacts around the edges with low-resolution textures.

NOTE: With high-quality occlusion maps, such as those baked using V-Ray, will require less retouch in PS later on so it is worth spending more time on how to bake the textures at the best possible quality.

7. ANIMATIONS

SUSPENSION ANIMATION

The 3D suspensions of a car can be animated if needed.
In order to enable suspension animations, you have to edit the the script "car.ini" with the following values:

`"USE_ANIMATED_SUSPENSIONS=1"`	to enable use of the animations
`"USE_ANIMATED_SUSPENSIONS=0"`	to leave the animations as by default (disabled)

Usually this means the animations are disabled, but on some occasions they are always on like the steering wheel which is automatically animated.

NOTE: Animating suspensions do have some disadvantages. Animating suspensions follow predetermined arcs and movements, so the wheels do not represent visually the setup values chosen by the player in game. i.e. Camber angles might differ visibly from the values selected in setup screen.

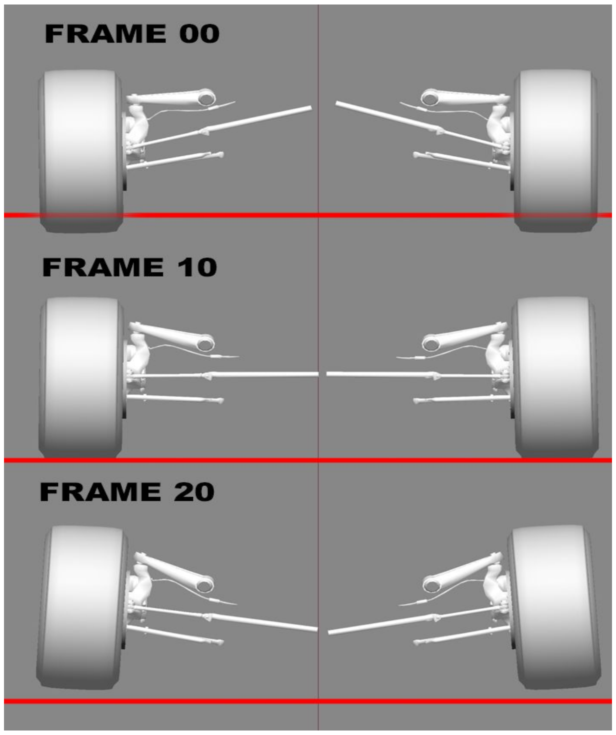
We use a NULL hierarchy to animate the suspension geometry. Here's an example below

1. Set your timeline frames to (for example) 20 frames.
2. The frame 0 will be lower position.
3. The frame 10 will be the neutral position.
4. The frame 20 the higher position.

The engine works as follows: It verifies the position in the y axis of the suspension and finds the right frame to match the animation to the position of the physical suspension. It will interpolate the frames to generate a smooth movement. Assetto Corsa will search for the following NULL/DUMMY objects, named as follows:

SUSP_LF
SUSP_LR
SUSP_RF
SUSP_RR

The NULL/DUMMY must be designed to move the suspension on the Y axis. Example of how to animate a suspension correctly:



NOTE: Before exporting the FBX, timeline needs to be set with the same number of total frames as the number of animated frames created.
Example: If you animate 20 frames, do not export with a timeline of 30. This will cause a crash. Remember to set the timeline to 20 if you have animated 20 frames. Empty frames will cause a crash and are not supported.

Suspension Hierarchy

The suspension must have the hierarchy identical to one of the two FBX examples provided:

TEMPLATE_Suspension_EASY.fbx and TEMPLATE_Suspension_COMPLEX.fbx

The first scene contains a simple suspension hierarchy, made for a car with simple suspension system. The second is prepared for complex suspension hierarchy, like 60's Formula 1 cars, with more complex arms and particular suspensions. Those examples contains more DUMMY/NULLs

Some names can be customized and we have named the customizable DUMMY/NULL in an appropriate way, inside the template.fbx
"Custom_name##". (where # is a number)
All the DUMMY/NULL are used for animated parts. Their use is optional. You can create the necessary number of DUMMY/NULL as you desire.

The following DUMMY/NULLs are mandatory:

Suspension Nulls:

SUSP_LF	Left Front
SUSP_LR	Left Rear
SUSP_RF	Right Front
SUSP_RR	Right Rear

Hub Nulls:

HUB_LF	Left Front
HUB_LR	Left Rear
HUB_RF	Right Front
HUB_RR	Right Rear

Wheels Nulls:

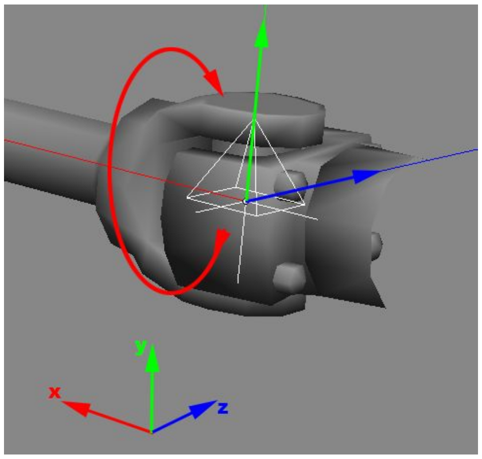
WHEEL_LF	parent of the TYRE_LF for the tyre mesh, RIM_LF for the Rim mesh, and RIM_BLUR_LF for the Rim Blurred mesh
WHEEL_LR	parent of the TYRE_LR for the tyre mesh, RIM_LR for the Rim mesh, and RIM_BLUR_LR for the Rim Blurred mesh
WHEEL_LR	parent of the TYRE_LR for the tyre mesh, RIM_LR for the Rim mesh, and RIM_BLUR_LR for the Rim Blurred mesh
WHEEL_RR	parent of the TYRE_RR for the tyre mesh, RIM_RR for the Rim mesh, and RIM_BLUR_RR for the Rim Blurred mesh

In some cars, the transmission shafts might be visible. There is a convention name to animate these objects automatically.

Transmission DUMMY/NULL names:
TRANSMISSION_L_1 for the Left shaft
TRANSMISSION_R_1 for the Right shaft

If you have more transmission pieces to animate, you can use sequential of numbering. For example: TRANSMISSION_L_2 , TRANSMISSION_L_3 and so on. The same applies to TRANSMISSION_R_2 and so on.
The engine recognizes the prefix "TRANSMISSION_L_" and looks for a sequential number after it. There is no hard-coded limit on the number of transmission parts that can be animated.

The transmission nulls rotate on the X axis and needs to be oriented like the image below:

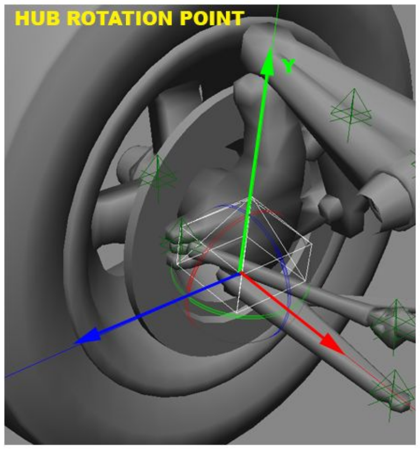


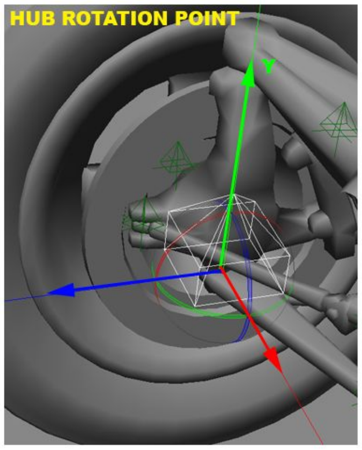
This node is useful for animating the joint on the Y axis according to the suspension animation, and the engine automatically rotates the transmission according to the wheel on X axis. Avoid animating on the Z axis, as it is not used.

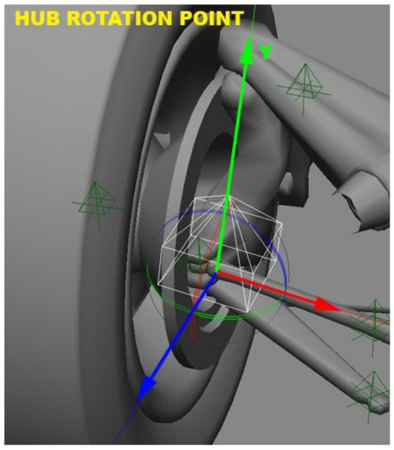
NOTE: In order to have a correct direction of rotation, the Z axis of the transmission nulls always need to point forward.

The engine recognizes 4 other nodes for the the hubs of every wheel. These nodes are designed to allow the hub to rotate in accordance with the camber of the wheel.

HUB_LF
HUB_LR
HUB_RF
HUB_RR







In the above image example, the hub is the parent of the steer arms.

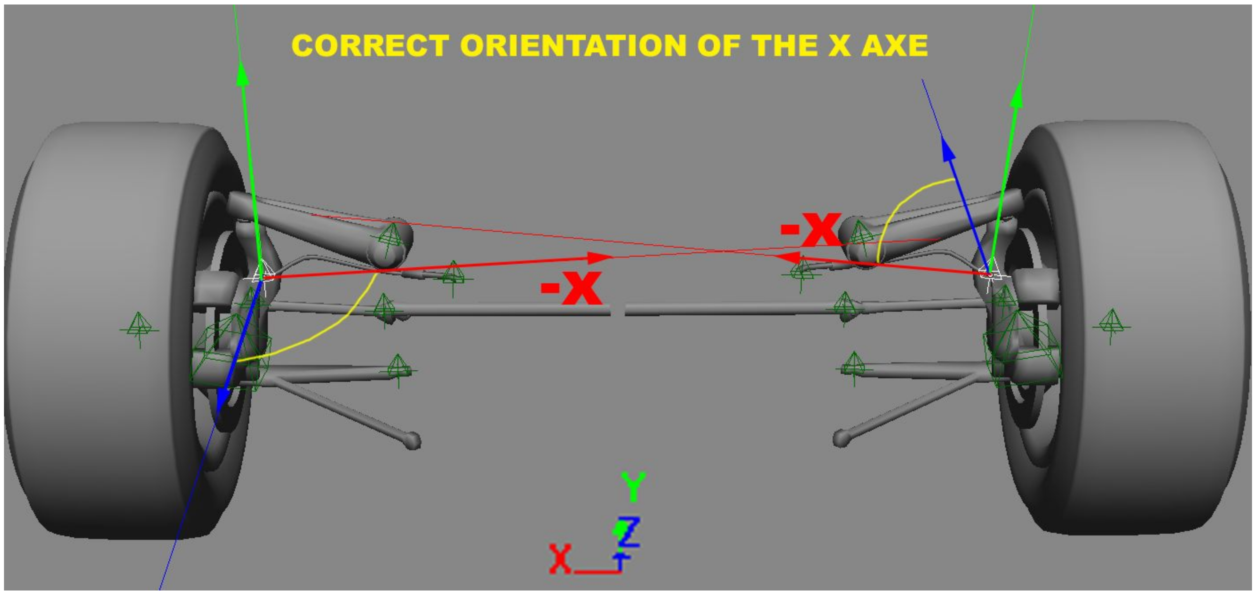
This way you can animate up and down movements of the hub or the suspension and during the animation you can change the camber of the hub as required by the actual physical suspension layout. Inside the TEMPLATE_Suspension_COMPLEX.fbx you can find an example for the correct hierarchy.
Note: The SUSP_ node must always match the position of the Wheel_ node. The AC engine verifies the position of the suspension in 3D space by checking the SUSP_ node. The wheel bounding box is recognized between the SUSP_ node and Wheel_ node. Those 2 positions must be the same.
Again the FBX file TEMPLATE_Suspension_COMPLEX.fbx is a perfect example.

STEER ARMS and DIRECTION CONSTRAINTS

We can animate many different parts and just import the animation to the editor and from that export to the game, but the STEER arm cannot be animated. Its position changes in the 3D space according to the HUB rotation.

In order to constrain the movement of the steer arm to the HUB's position and rotation, the convention name with a prefix "DIR_customName" must be used. This indicates the direction of the X negative axis of this mesh, and the null called "customName" will point the X axis to the correct direction.

Example: a null called "SteerArm_L" will point the negative X axis in direction of a null named "DIR_SteerArm_L"
Pay attention to the rotation of the null which the animated mesh is linked to. In the image below the right-hand side Null point has a positive Z axis. The left Null point has a negative Z axis. This allows the -X axis to point to the center of the car or any other direction required by the mesh.
Inside the TEMPLATE_Suspension_COMPLEX.fbx file, you can find a proper hierarchy example.



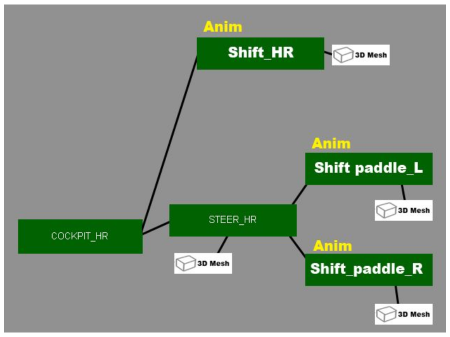
Note: You can create more constraints, if you have more objects to constraint to the HUB by simply giving them different names. Nevertheless, it is always good in terms of optimization to use the lowest possible number of constraints.
You can animate your custom nulls in the following vectors: Rotation - Translation - Scale.
Inside the TEMPLATE_Suspension_COMPLEX.fbx example file, you can see the animation of the suspension spring, on SCALE Y .
Note: Never animate the mesh. Always animate the NULLs only! With this approach you can change and update your mesh every time you want without re-exporting the animations. Use the same technique to create animations for any NULL that has to be animated. For example, doors, gearbox levers, or any other parts.

CONSTRAINT FULL ANIMATION SETUP

If you don't want animate the suspension manually you can create a full CONSTRAN setup.
You can use the DIRECTION CONSTRAINT logic to force you suspension to work automatically without animate them
A example of this kind of suspension setup can be found in the EXAMPLE FBX provided with the SDK.
Costraint_suspension_Only.fbx for generic FBX 2014 version
Costraint_suspension_Only_XSI_2014.scn for XSI 2014 version
Costraint_suspension_Only_MAX_2013.scn for 3ds MAX 2013 version

ANIMATION EXPORT

Once you have animated your NULL/DUMMY inside your scene, using the AC Editor you need to export the animation to an AC-specific clip format, called name.ksanim.



A couple of rules must be followed:
1) To generate a clip, you only need to animate the nulls. Animating the mesh itself is not needed. You can also export the mesh, but the editor will only export the objects/nulls that have animation keyframes. As we mentioned before, it is a good technique to animate only the NULL/DUMMY so that you can change your animation independently from the actual 3D mesh.

2) When you export an animated null, you must also export the the hierarchy tree above it, as the name of an object inside the editor is determined by its position in the hierarchy tree.

3) Always export using the FBX format, as it is the only format that supports animation. Do not use any other formats.

In the example shown in the image above, we have two null hierarchies that contain meshes as their children. We have animated the shift and steer paddle and we need to export the animation.

We cannot export the SHIFT PADDLE_L null only. We must take the entire hierarchy from COCKPIT_HR, including STEER_HR and SHIFT paddle_L.

This way the editor will define the null related to the position of SHIFT PADDLE_L in the hierarchy.

OPTIMISING ANIMATIONS

1) The frames are interpolated in the game. You don't need to export an animation with all the keyframes. For simple animations, like doors, gear levers and so on, you can export just the important frames only. For a simple door animation, the “close” and “open” frames are enough, the game will interpolate the rest. When you have more complex doors, with pistons, vertical openings, like those in a Mclaren P1, you can add more frames to animate the door in a more precise way. But always keep in mind that the less frame you use, the more optimized the result will be, because the engine interpolates smoothly between the keyframes.

2) Identical frames are optimized. If you create multiple identical frames, and the variation between them is 0, the frames will be optimized.

Example: A gear lever starts animation at frame 15 of the complete animation, because during previous frames it stays fixed to its position, waiting for the driver’s hand to first reach it. A keyframe must be placed to frame 0 and another one at frame 14, both in static position. The animation of the gear lever starts at frame 15.There is no need to place more keyframes between 0 and 14.

CLIPS AND NAMING CONVENTIONS

There are two types of pre-programmed playbacks:

1) Ping-pong: The animation played reaches the end is then played in reverse from back to start.
2) Loop: When the first frame match the last frame and the animation restarts the loop.

The specific naming conventions have a pre-programmed playback in-game, so the engine knows whether the playback should be LOOP or PING-PONG..

DRIVER ANIMATION CLIP NAMES:

steer.ksanim	Loop for the rotation of the driver’s arms on the steering wheel
shift.ksanim	PingPong for the animation of the driver’s arm to the gearshift lever (we usually do a simple animation, check the fbx example)
shift_dw.ksanim	PingPong for the animation of the fingers that operate the paddle to shift down (usually left)
shift_up.ksanim	PingPong for the animation of the fingers that operate the paddle to shift up (usually right)

CAR ANIMATION CLIP NAMES

car_shift.ksanim	PingPong to animate the car gear lever

Important: this clip must have the same number of frames as the shift.ksanim to match the arm movement with the shift animation. If in the driver animation, the shifting movement begins at frame 10, the shifter must also start to move at the exact same frame!

car_susp_LF.ksanim	Controlled by engine for the animation of the Left Front suspension
car_susp_LR.ksanim	Controlled by engine for the animation of the Left Rear suspension
car_susp_RF.ksanim	Controlled by engine for the animation of the Right Front suspension
car_susp_RR.ksanim	Controlled by engine for the animation of the Right Rear suspension
car_door_R.ksanim	PingPong for the animation to open the right-hand side door (the closing animation should be the open animation in reverse. Do not animate the closing sequence!)
car_door_L.ksanim	PingPong for the animation to open the right-hand side door (the closing animation should be the open animation in reverse. Do not animate the closing sequence!)
car_wiper.ksanim	Loop for the animation of the wiper (here you must animate the full animation back and forth)
lights.ksanim	PingPong for the animation of the car lights that are dynamic (e.g. Ferrari F40. Animate the opening sequence ONLY).
car_shift_up.ksanim	PingPong for animating the paddle shift up
car_shift_dw.ksanim	PingPong for animating the paddle shift down

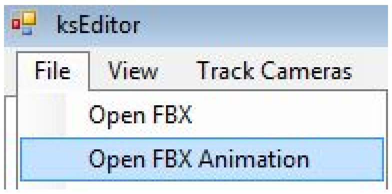
Note: You can create all the animations needed. You can also use new names and then engage them from an .ini script (such as active DRS, wings etc.). The names listed above are recognized automatically and managed by the game engine.

For example: the animation car_wing.ksnim is a custom name. On certain cars we created animations called wing_rear.ksanim or wing_side.ksanim. These optional animations are managed from .ini scripts.

car_wing.ksanim	PingPong for the animation of dynamic wings (animate only opening sequence)

EXPORTING ANIMATIONS FROM THE EDITOR

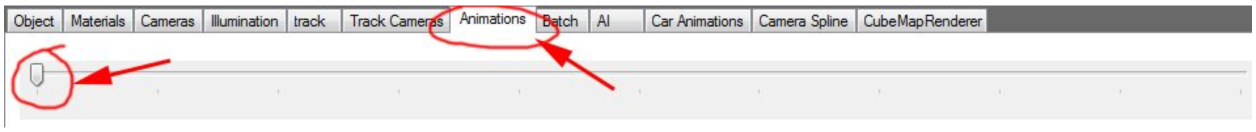
Follow these steps to export animations:
- Open the car FBX in the editor.
- Open the ANIMATION with the Open FBX Animation option under the File tab (see below):



- Find and load the FBX animation that you had previously exported to the “animations” folder.

When an animation is loaded it automatically saves a clip_name.ksanim file in the same folder where your FBX file is located, there is no need to manually export the animation clip.

- Select the Animation tab at the bottom part of the editor UI. Drag the animation slider, and you should see the animation playback.



You can load multiple animations into the scene. Every time you load an animation, a .ksanim file is saved with the same name as the source FBX.
If your fbx is not named properly, you have to rename your clips to match our name conventions, for example steer.ksanim for the animation of the driver arms etc.

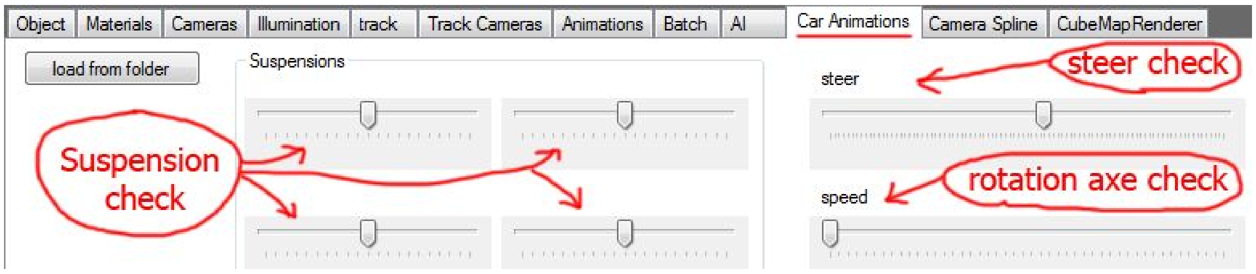
CHECKING CAR ANIMATIONS

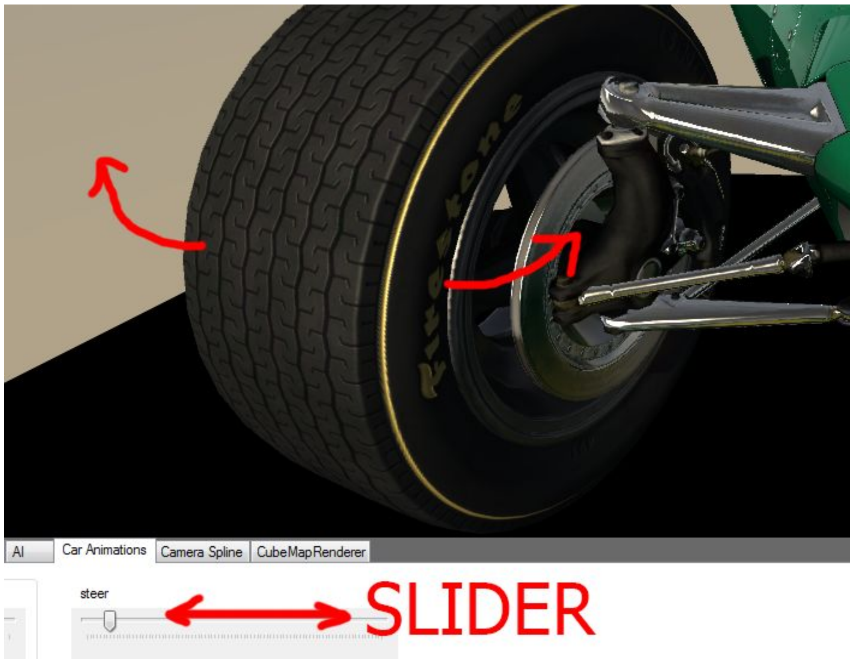
When all your suspensions clips are exported to the proper animation folder (see "**Project Structure**" section), you can load your car.fbx in the editor and check if the animations work properly.
You can do this only after you have created the animation clips and by loading them into the editor.

NOTE: LOD B must contain the same null hierarchy and names of animated nulls as LOD A for the exterior (suspension, wings, pop-up lights), but not for animations in the HR interior (paddle and shifter) and the doors nulls under the Cockpit_HR null.

Open you car_name.fbx

After the car is loaded, you can load the clip of the suspension that you want to test.
In the tab called "Car Animations" you'll find sliders. These are designed to help you test the animation of the springs, the constraint of the arms and the wheel rotation.

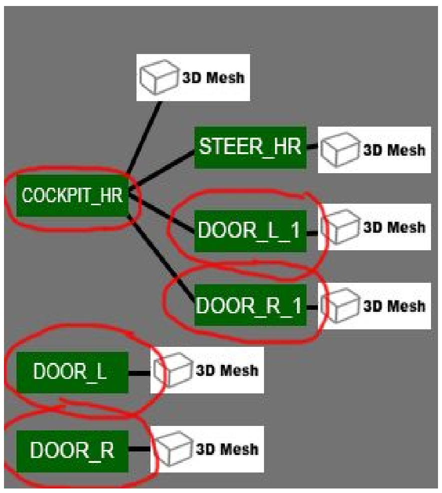


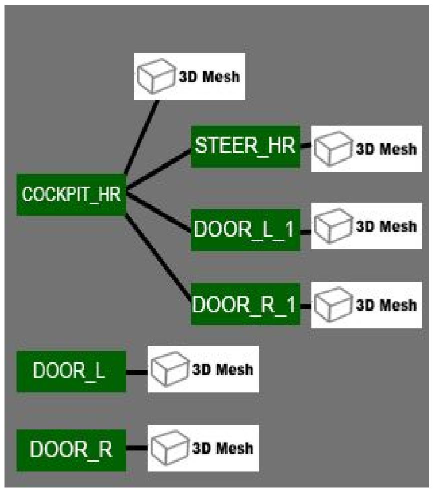


The sliders allow you to check the suspension in the editor and detect any issues, frame by frame. Here you can see an example: Moving the slider you can check the hub behaviour.

GENERIC ANIMATION EXPORTING GUIDELINES

There are some important things to keep in mind when exporting animations, especially door animation clips. When exporting a door clip, the following hierarchy should be present:





Complex door animations may include a higher number of nulls, make sure that the naming conventions are consistent and that you export every null that is animated (per side). For more information, see **ANIMATION EXPORT**.

Note that the meshes and nulls of the interior door elements are under the null, COCKPIT_HR. This is needed because the cockpit switches from high to low resolution. The LR door will be hidden. So we have duplicated the door animation nulls with animation included, and placed them outside of COCKPIT_HR. When you export, remember to include all the cockpit door nulls (left image).

NOTE: When exporting the animation of the paddle shifters (car_shift_dw and car_shift_up), make sure that you export the parent nulls as well. If the paddles are on the steering wheel, for each animation you have to export either of the paddle nulls (SHIFT_R or SHIFT_L), the null for the steering wheel (STEER_HR) and the HR cockpit null (COCKPIT_HR).

To animate wipers, you may use a number of nulls depending on the complexity of the wiper. Usually a wiper with 2, maximum 3 pivot points (and thus 2 or 3 dummies) is sufficient.
The wiper nulls must be located in the root of the scene and they must be present in **LOD A through LOD C**.

8. MATERIALS AND AC EDITOR

In the following section you will find information about how to use the AC editor and guidelines for setting up materials.

MATERIAL NAMING CONVENTIONS

Please use the following conventions for naming materials in your 3D software. Firstly, ALWAYS indicate whether it is for the interior or exterior (INT_ or EXT_), then indicate the name that is straightforward (usually indicating the texture it is using) and lastly indicate if it has to use a certain transparency property, such as AT for alpha test. When possible, use the following names:

EXT_Tyre	
EXT_Rim	
EXT_Rim_blur	
EXT_Rim_blur_Alpha	transparent blurred rim part (use **alpha blend** mode and transparency ON)
EXT_Carpaint	
EXT_Carbon	
EXT_Details_AT	for labels and logos (use alpha test mode with transparency OFF)
EXT_Details_Plastic	Details=using the exterior details texture
EXT_Details_Metal	
EXT_Details_Chrome	
EXT_Engine	
EXT_Disc	
EXT_Caliper	
EXT_Window	
EXT_Lights_Glass	
EXT_Lights_Chrome	

INT_Details_AT	for labels and logos (alpha test mode with transparency OFF)
INT_Details_Plastic	Details=using the interior details texture
INT_Details_Chrome	
INT_Details_Metal_Black	
INT_Details_Metal_Flat	
INT_Details_Gauges	
INT_OCC_Carbon	OCC=using the ambient occlusion texture
INT_OCC_Leather	
INT_OCC_Alcantara	
INT_OCC_Plastic	
INT_OCC_Metal	
INT_BELT	
INT_LCD	ALWAYS keep the digital display on a separate texture (for racing cars)
INT_FUEL_INDICATOR	material with emissive for the fuel warning light

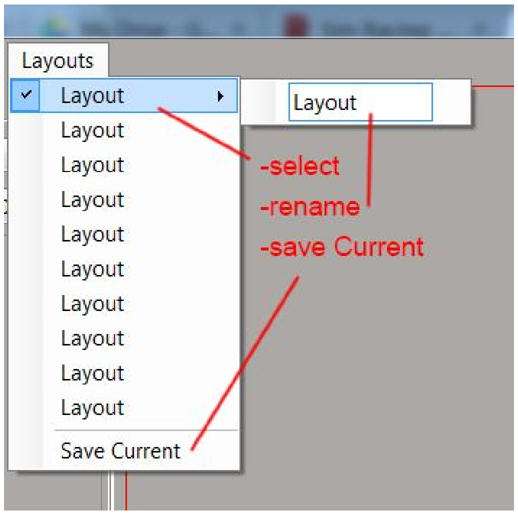
If needed, you can use multiple materials (for multiple carbon patterns), in this case, make a distinction with numbers or name (e.g. INT_OCC_Carbon_Flat and INT_OCC_Carbon_Refl). In any case, strive to differentiate exterior and interior materials.

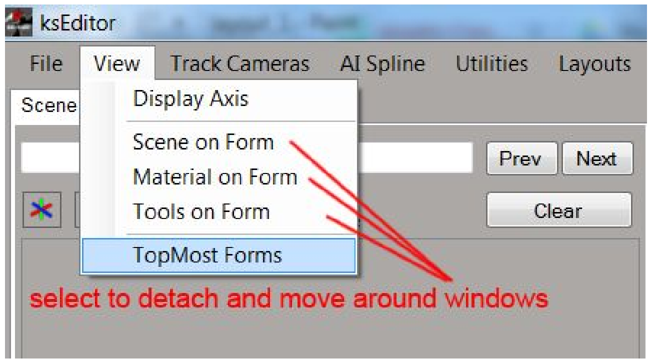
NOTE: For a more comprehensive guide and community tips to use the editor, see the following thread on the official support forum:
http://www.assettocorsa.net/forum/index.php?threads/ac-editor.10964/

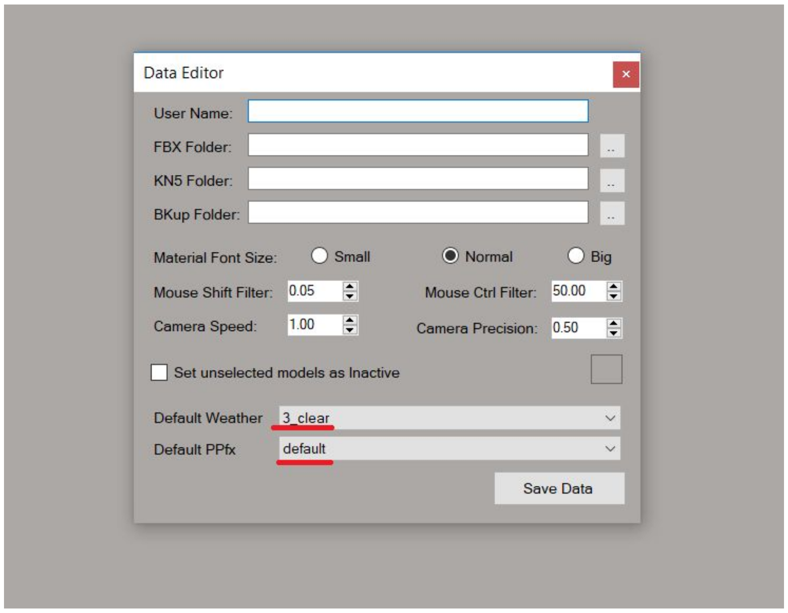
NOTE: to find useful information and request help for general editor and shader-related issues, see the following thread on the official support forum:
http://www.assettocorsa.net/forum/index.php?threads/car-materials-shaders-modelling-stuff-add-your-knowledge-here.19704/

Basic guide to the KS EDITOR:

When you first open the editor, make sure you save the layout and set your preferences under Utilities/Data Editor.







Data Editor

User Name:			
FBX Folder:			
KN5 Folder:			
BKup Folder:			
Material Font Size:	○ Small	● Normal	○ Big
Mouse Shift Filter:	0.05	Mouse Ctrl Filter:	50.00
Camera Speed:	1.00	Camera Precision:	0.50

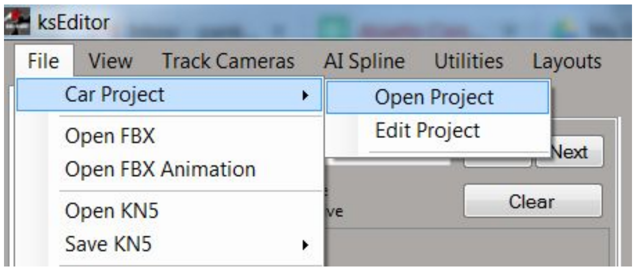
☐ Set unselected models as Inactive

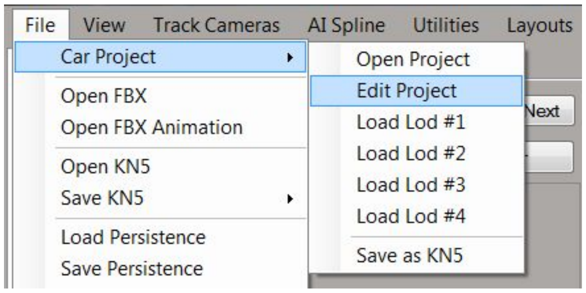
Default Weather: 3_clear

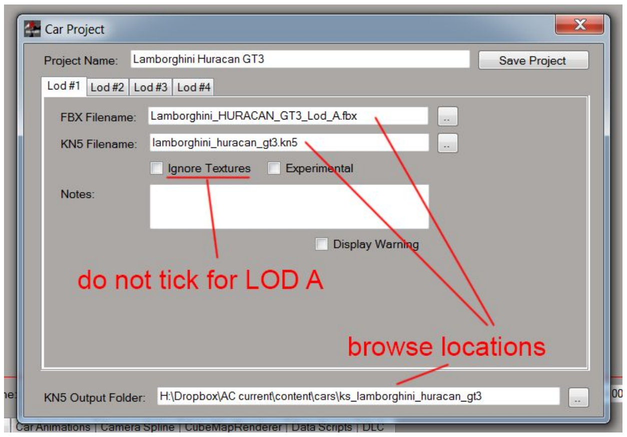
Default PPfx: default

Save Data

You can use the built-in **Project Manager** to save and manage projects:







Car Project

Project Name: Lamborghini Huracan GT3 — Save Project

Lod #1 | Lod #2 | Lod #3 | Lod #4

FBX Filename: Lamborghini_HURACAN_GT3_Lod_A.fbx

KN5 Filename: lamborghini_huracan_gt3.kn5

☐ Ignore Textures ☐ Experimental

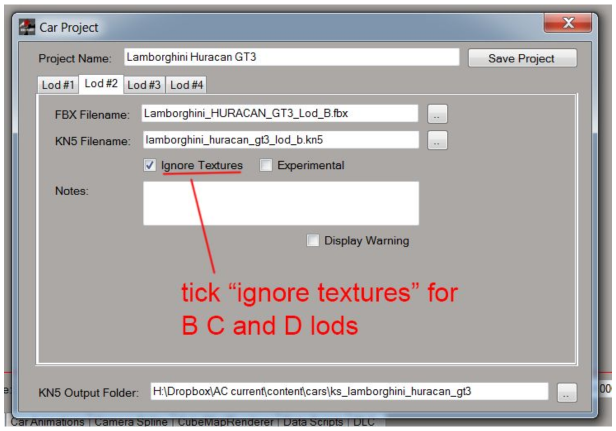
Notes:

☐ Display Warning

do not tick for LOD A

browse locations

KN5 Output Folder: H:\Dropbox\AC current\content\cars\ks_lamborghini_huracan_gt3



Car Project

Project Name: Lamborghini Huracan GT3 — Save Project

Lod #1 | Lod #2 | Lod #3 | Lod #4

FBX Filename: Lamborghini_HURACAN_GT3_Lod_B.fbx

KN5 Filename: lamborghini_huracan_gt3_lod_b.kn5

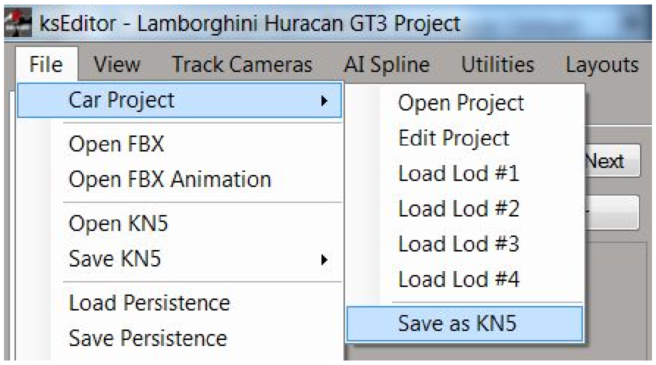
☑ Ignore Textures ☐ Experimental

Notes:

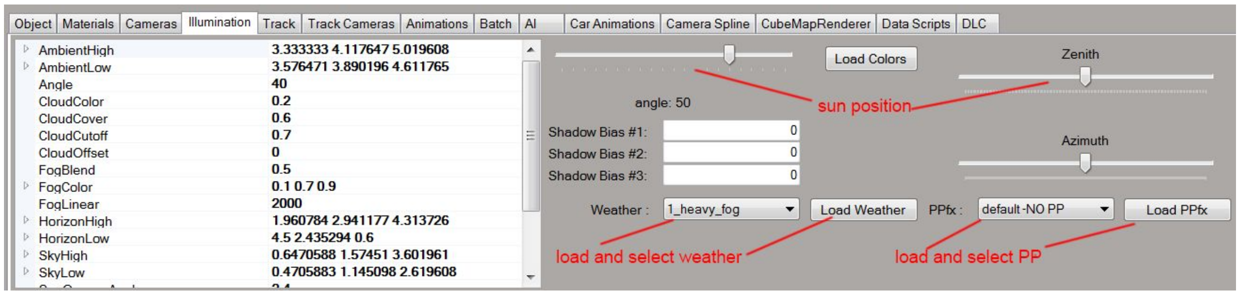
☐ Display Warning

tick "ignore textures" for B C and D lods

KN5 Output Folder: H:\Dropbox\AC current\content\cars\ks_lamborghini_huracan_gt3



Scene illumination in the editor can be changed under the **Illumination** tab:



AmbientHigh	3.333333 4.117647 5.019608
AmbientLow	3.576471 3.890196 4.611765
Angle	40
CloudColor	0.2
CloudCover	0.6
CloudCutoff	0.7
CloudOffset	0
FogBlend	0.5
FogColor	0.1 0.7 0.9
FogLinear	2000
HorizonHigh	1.960784 2.941177 4.313726
HorizonLow	4.5 2.435294 0.6
SkyHigh	0.6470588 1.57451 3.601961
SkyLow	0.4705883 1.145098 2.619608

angle: 50

Shadow Bias #1: 0

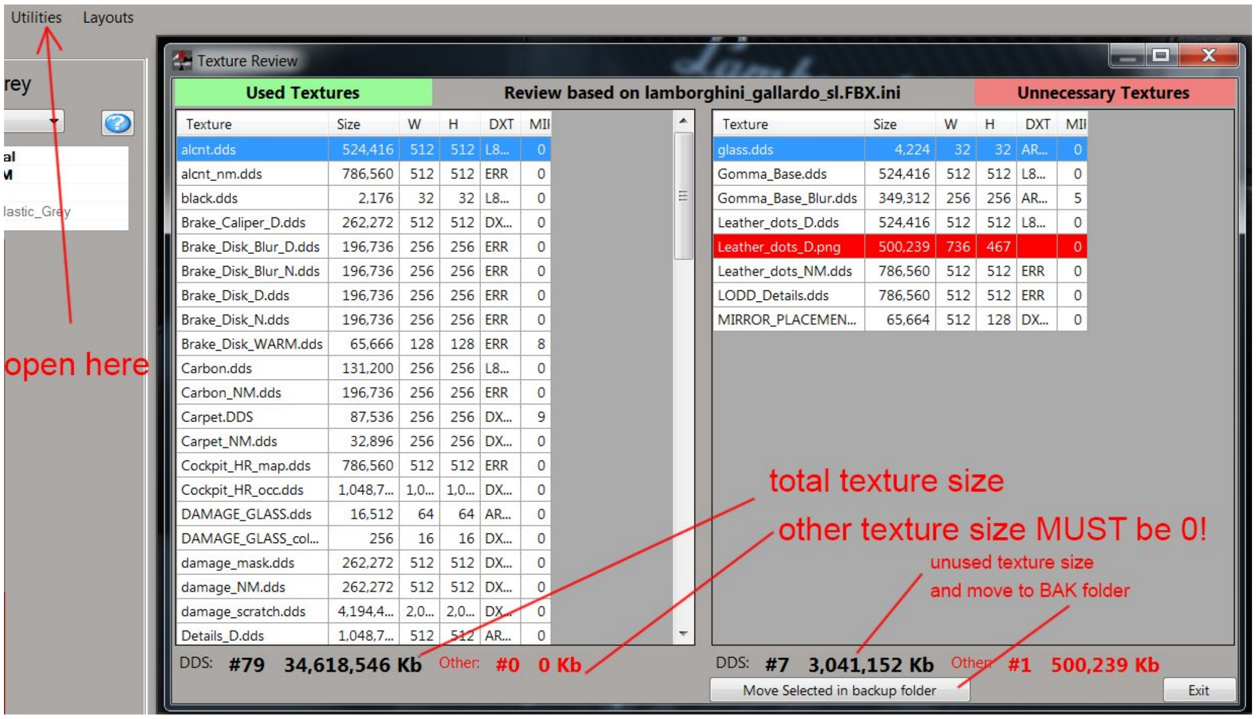
Shadow Bias #2: 0

Shadow Bias #3: 0

Weather : 1_heavy_fog

PPfx : default -NO PP

You can review your textures using the **Texture Review** tool under Utilities/Texture Review (option only visible when a model is loaded).



Used Textures — Review based on lamborghini_gallardo_sl.FBX.ini

Texture	Size	W	H	DXT	MII
alcnt.dds	524,416	512	512	L8...	0
alcnt_nm.dds	786,560	512	512	ERR	0
black.dds	2,176	32	32	L8...	0
Brake_Caliper_D.dds	262,272	512	512	DX...	0
Brake_Disk_Blur_D.dds	196,736	256	256	ERR	0
Brake_Disk_Blur_N.dds	196,736	256	256	ERR	0
Brake_Disk_D.dds	196,736	256	256	ERR	0
Brake_Disk_N.dds	196,736	256	256	ERR	0
Brake_Disk_WARM.dds	65,666	128	128	ERR	8
Carbon.dds	131,200	256	256	L8...	0
Carbon_NM.dds	196,736	256	256	ERR	0
Carpet.DDS	87,536	256	256	DX...	9
Carpet_NM.dds	32,896	256	256	DX...	0
Cockpit_HR_map.dds	786,560	512	512	ERR	0
Cockpit_HR_occ.dds	1,048,7...	1,0...	1,0...	DX...	0
DAMAGE_GLASS.dds	16,512	64	64	AR...	0
DAMAGE_GLASS_col...	256	16	16	DX...	0
damage_mask.dds	262,272	512	512	DX...	0
damage_NM.dds	262,272	512	512	DX...	0
damage_scratch.dds	4,194,4...	2,0...	2,0...	DX...	0
Details_D.dds	1,048,7...	512	512	AR...	0

DDS: **#79 34,618,546 Kb** Other: **#0 0 Kb**

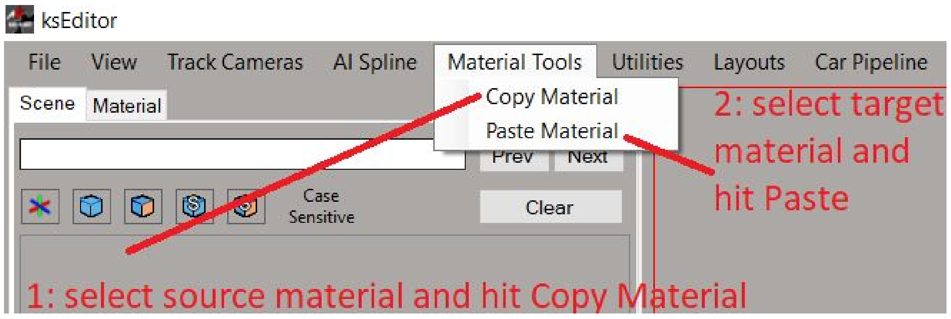
Unnecessary Textures

Texture	Size	W	H	DXT	MII
glass.dds	4,224	32	32	AR...	0
Gomma_Base.dds	524,416	512	512	L8...	0
Gomma_Base_Blur.dds	349,312	256	256	AR...	5
Leather_dots_D.dds	524,416	512	512	L8...	0
Leather_dots_D.png	500,239	736	467		0
Leather_dots_NM.dds	786,560	512	512	ERR	0
LODD_Details.dds	786,560	512	512	ERR	0
MIRROR_PLACEMEN...	65,664	512	128	DX...	0

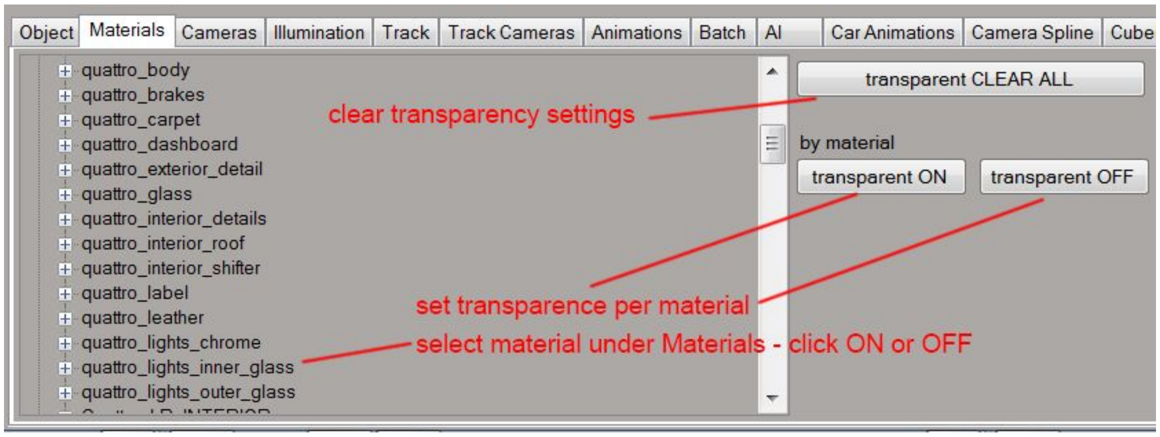
DDS: **#7 3,041,152 Kb** Other: **#1 500,239 Kb**

It is recommended you keep your texture folder organised. You can back up your unused textures with the "Move Selected in backup folder" button.

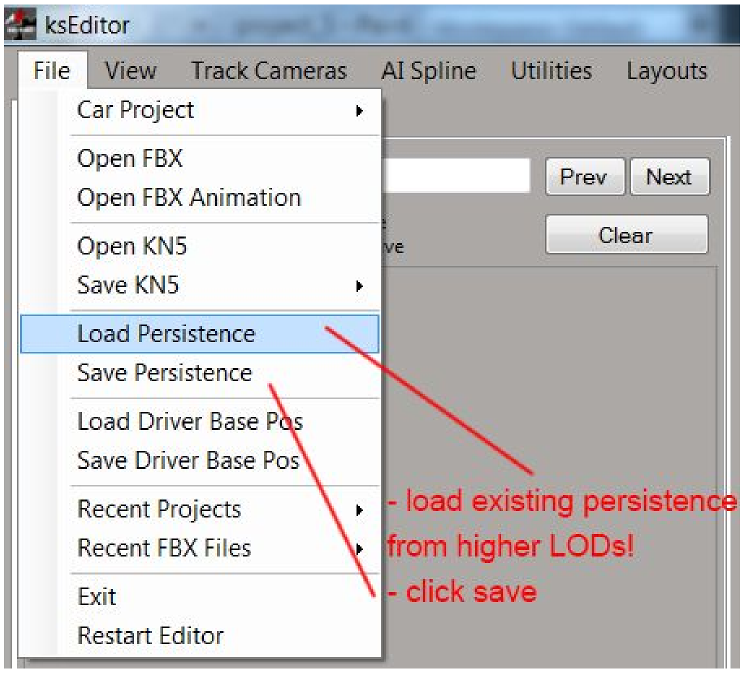
You can use a **Copy&Paste** tool to copy existing shader properties to a second material.
Note that first you need to select the correct shader (if source is ksPerpixelMultimap, the target needs to be a multimap material as well and so on), then fill in the shader slots manually! After these steps are done, you can use the tool under **Material Tools** to copy and paste the shader values, as follows:



Transparencies and **cast shadow** settings can be applied globally to materials under the **Materials** tab:



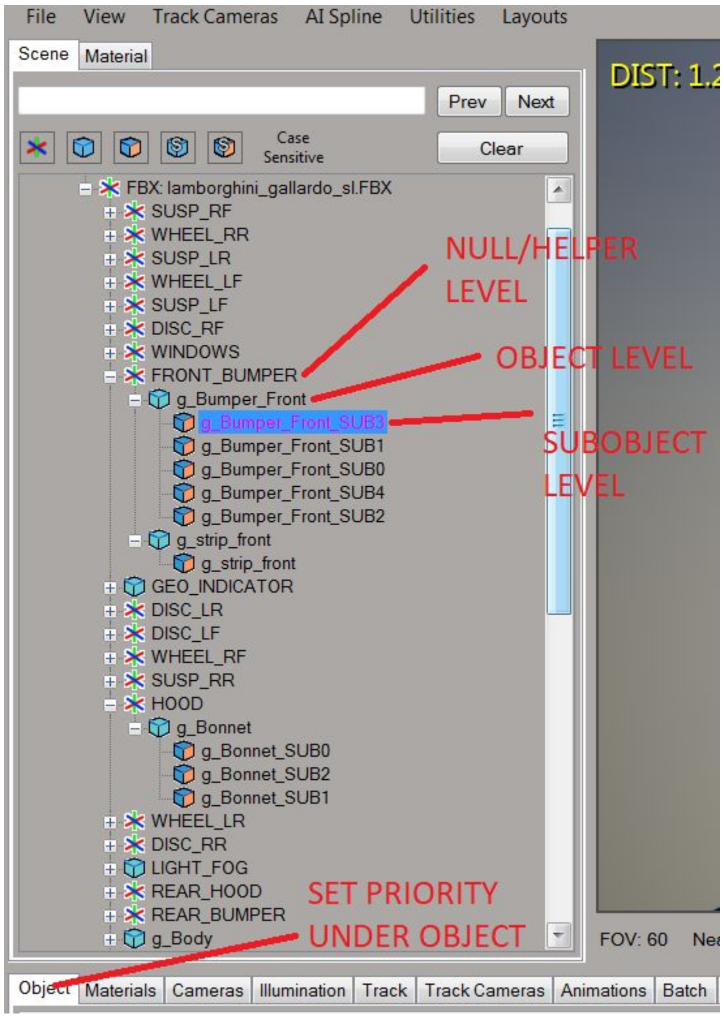
Persistence files (containing shader and object settings) can be saved under File. You can also load existing persistence files from higher LODs. Note that loading a persistence file on a new export will only transfer shader settings, transparencies and cast shadow settings will have to be set manually. However, for later persistence updates using the load function (once the transparencies are set), the transparencies will not have to be set again.



General guidelines to using different alpha modes:

Where possible try to avoid using **Alpha Blend** mode. Blend requires transparency, which can cause issues with draw priorities, because some objects can be viewed from two directions.
A common issue is the interior: interior objects sometimes (such as the transparent interior windshield banner) need a priority set to 1 to avoid the object being drawn before the external glass objects when viewed from outside.
However, from cockpit view, this can cause issues with the blurred rim on opponent cars, because the blurred spokes object with a priority of 0 will draw before the interior banner, if for example it goes around the windshield.
Of course, Alpha Blend mode is still required for glass objects and the blurred rim spokes.

When using priorities, make sure the priority is applied on the **object** level, not the sub-object level.
Additionally, if the transparent object is linked to a helper, you have to assign the priority on the highest level in the hierarchy, thus the helper itself (see the section about **Damage Glass**).



It is recommended to detach all transparent object as separate objects in your 3D software before exporting. **AVOID** including transparent objects in a group object with multiple material IDs. This is very important because otherwise adding a new material ID later on could cause the transparency and cast shadow settings to “migrate” to another subobject, incorrectly assigning transparency to otherwise opaque pieces of mesh.

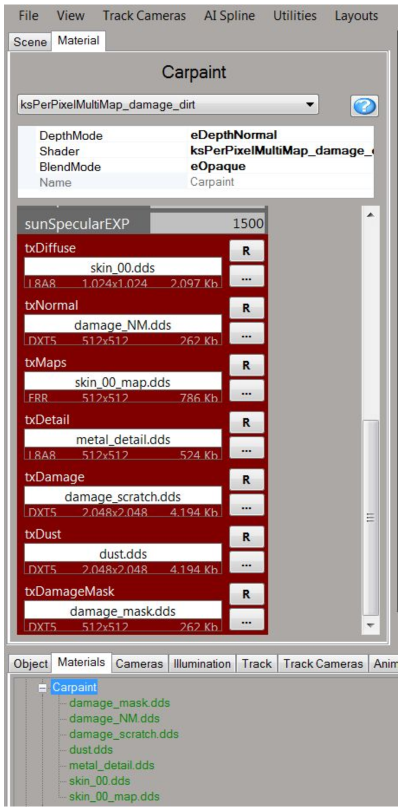
Alpha Test mode usually works to a satisfying level when the alpha has no gradient. Alpha Test requires no transparency, which is why no issues will arise if more layers are in front of one another. In Alpha Test materials, transparency is defined by the Alpha channel in the **Normal Map**.

Alpha Test mode can also be used to **hide certain objects** using a simple texture (make sure you disable shadow casting for those objects). Remember that you control transparency with the Normal Map alpha channel.

Opaque mode is required for non-transparent objects, or where the shape of objects is defined by the mesh. Make sure you don’t group objects with different properties in this respect under the same material. If you have objects that require the alpha channel to define their border, group them under a new material. As a general rule, keep alpha and non-alpha objects in separate materials.

Guide to shader types

Below you find a few shader types used for specific parts of the vehicle, showing the recommended shader and blending mode (note that other properties are merely representative):



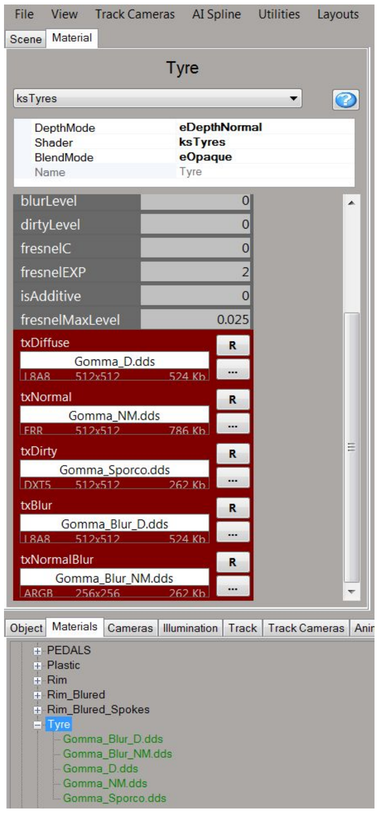
Carpaint

ksPerPixelMultiMap_damage_dirt

DepthMode	**eDepthNormal**
Shader	**ksPerPixelMultiMap_damage_**
BlendMode	**eOpaque**
Name	Carpaint

sunSpecularEXP	1500			
txDiffuse	skin_00.dds	L8A8	1.024x1.024	2.097 Kb
txNormal	damage_NM.dds	DXT5	512x512	262 Kb
txMaps	skin_00_map.dds	FRR	512x512	786 Kb
txDetail	metal_detail.dds	L8A8	512x512	524 Kb
txDamage	damage_scratch.dds	DXT5	2.048x2.048	4.194 Kb
txDust	dust.dds	DXT5	2.048x2.048	4.194 Kb
txDamageMask	damage_mask.dds	DXT5	512x512	262 Kb

- Carpaint
  - damage_mask.dds
  - damage_NM.dds
  - damage_scratch.dds
  - dust.dds
  - metal_detail.dds
  - skin_00.dds
  - skin_00_map.dds



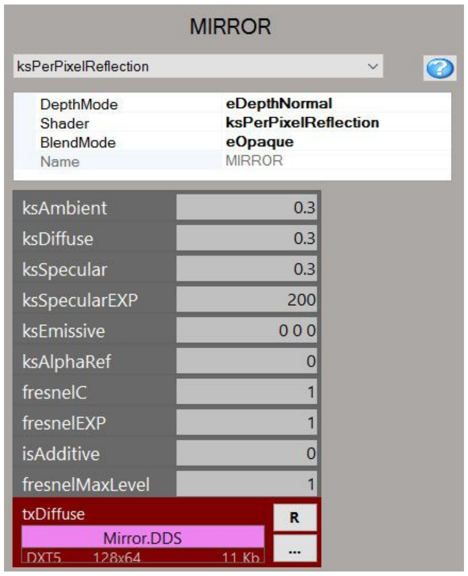
Tyre

ksTyres

DepthMode	**eDepthNormal**
Shader	**ksTyres**
BlendMode	**eOpaque**
Name	Tyre

blurLevel	0			
dirtyLevel	0			
fresnelC	0			
fresnelEXP	2			
isAdditive	0			
fresnelMaxLevel	0.025			
txDiffuse	Gomma_D.dds	L8A8	512x512	524 Kb
txNormal	Gomma_NM.dds	FRR	512x512	786 Kb
txDirty	Gomma_Sporco.dds	DXT5	512x512	262 Kb
txBlur	Gomma_Blur_D.dds	L8A8	512x512	524 Kb
txNormalBlur	Gomma_Blur_NM.dds	ARGB	256x256	262 Kb

- PEDALS
- Plastic
- Rim
- Rim_Blured
- Rim_Blured_Spokes
- Tyre
  - Gomma_Blur_D.dds
  - Gomma_Blur_NM.dds
  - Gomma_D.dds
  - Gomma_NM.dds
  - Gomma_Sporco.dds

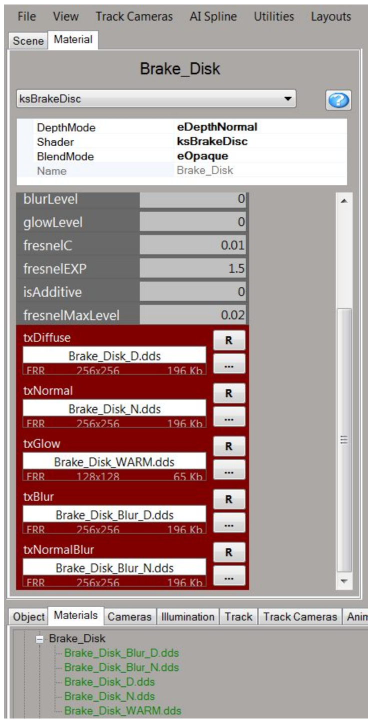


MIRROR

ksPerPixelReflection

DepthMode	**eDepthNormal**
Shader	**ksPerPixelReflection**
BlendMode	**eOpaque**
Name	MIRROR

ksAmbient	0.3			
ksDiffuse	0.3			
ksSpecular	0.3			
ksSpecularEXP	200			
ksEmissive	0 0 0			
ksAlphaRef	0			
fresnelC	1			
fresnelEXP	1			
isAdditive	0			
fresnelMaxLevel	1			
txDiffuse	Mirror.DDS	DXT5	128x64	11 Kb



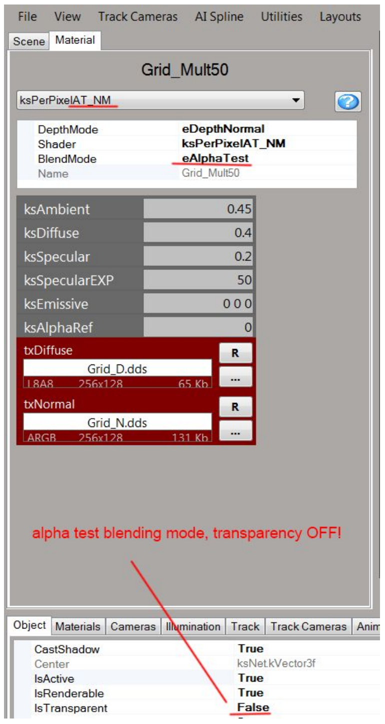
Brake_Disk

ksBrakeDisc

DepthMode	**eDepthNormal**
Shader	**ksBrakeDisc**
BlendMode	**eOpaque**
Name	Brake_Disk

blurLevel	0			
glowLevel	0			
fresnelC	0.01			
fresnelEXP	1.5			
isAdditive	0			
fresnelMaxLevel	0.02			
txDiffuse	Brake_Disk_D.dds	FRR	256x256	196 Kb
txNormal	Brake_Disk_N.dds	FRR	256x256	196 Kb
txGlow	Brake_Disk_WARM.dds	FRR	128x128	65 Kb
txBlur	Brake_Disk_Blur_D.dds	FRR	256x256	196 Kb
txNormalBlur	Brake_Disk_Blur_N.dds	FRR	256x256	196 Kb

- Brake_Disk
  - Brake_Disk_Blur_D.dds
  - Brake_Disk_Blur_N.dds
  - Brake_Disk_D.dds
  - Brake_Disk_N.dds
  - Brake_Disk_WARM.dds



Grid_Mult50

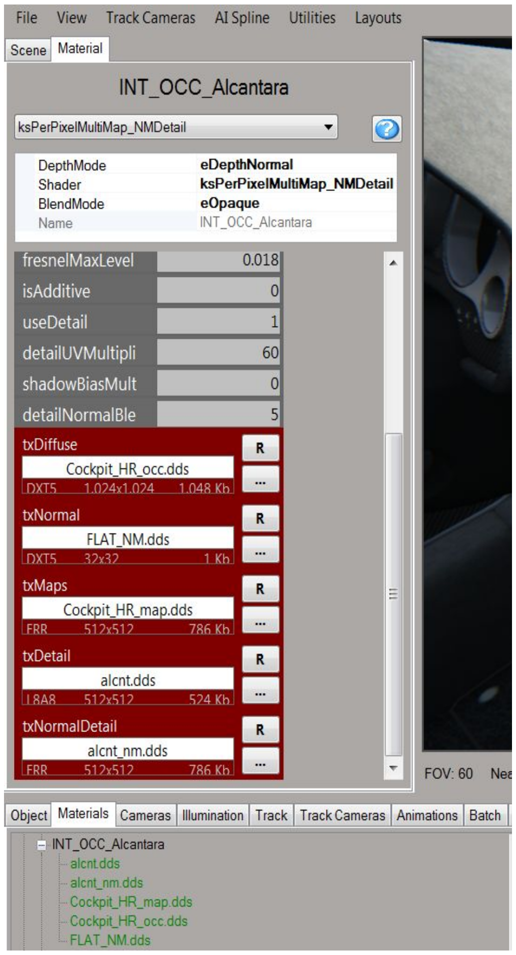
ksPerPixelAT_NM

DepthMode	**eDepthNormal**
Shader	**ksPerPixelAT_NM**
BlendMode	**eAlphaTest**
Name	Grid_Mult50

ksAmbient	0.45			
ksDiffuse	0.4			
ksSpecular	0.2			
ksSpecularEXP	50			
ksEmissive	0 0 0			
ksAlphaRef	0			
txDiffuse	Grid_D.dds	L8A8	256x128	65 Kb
txNormal	Grid_N.dds	ARGB	256x128	131 Kb

alpha test blending mode, transparency OFF!

CastShadow	**True**
Center	ksNet.kVector3f
IsActive	**True**
IsRenderable	**True**
IsTransparent	**False**



INT_OCC_Alcantara

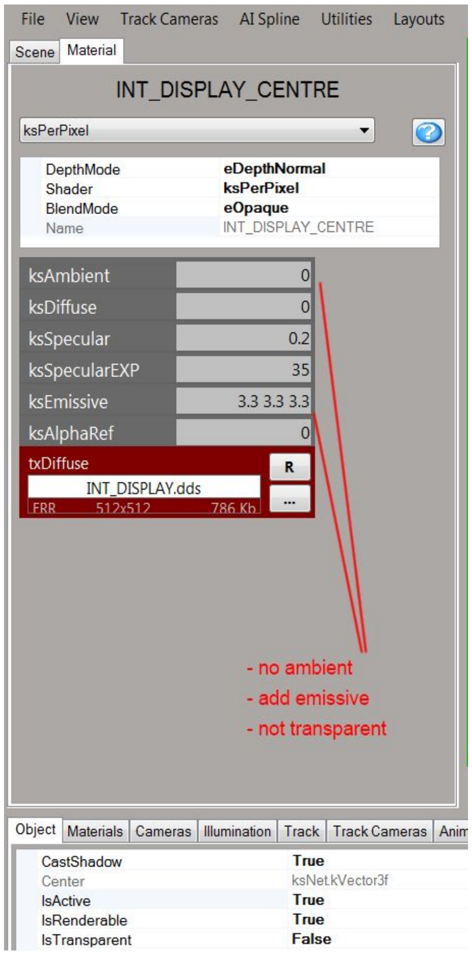
ksPerPixelMultiMap_NMDetail

DepthMode	eDepthNormal
Shader	ksPerPixelMultiMap_NMDetail
BlendMode	eOpaque
Name	INT_OCC_Alcantara

fresnelMaxLevel	0.018
isAdditive	0
useDetail	1
detailUVMultipli	60
shadowBiasMult	0
detailNormalBle	5

txDiffuse	Cockpit_HR_occ.dds	DXT5	1.024x1.024	1.048 Kb
txNormal	FLAT_NM.dds	DXT5	32x32	1 Kb
txMaps	Cockpit_HR_map.dds	FRR	512x512	786 Kb
txDetail	alcnt.dds	L8A8	512x512	524 Kb
txNormalDetail	alcnt_nm.dds	FRR	512x512	786 Kb

- INT_OCC_Alcantara
  - alcnt.dds
  - alcnt_nm.dds
  - Cockpit_HR_map.dds
  - Cockpit_HR_occ.dds
  - FLAT_NM.dds



INT_DISPLAY_CENTRE

ksPerPixel

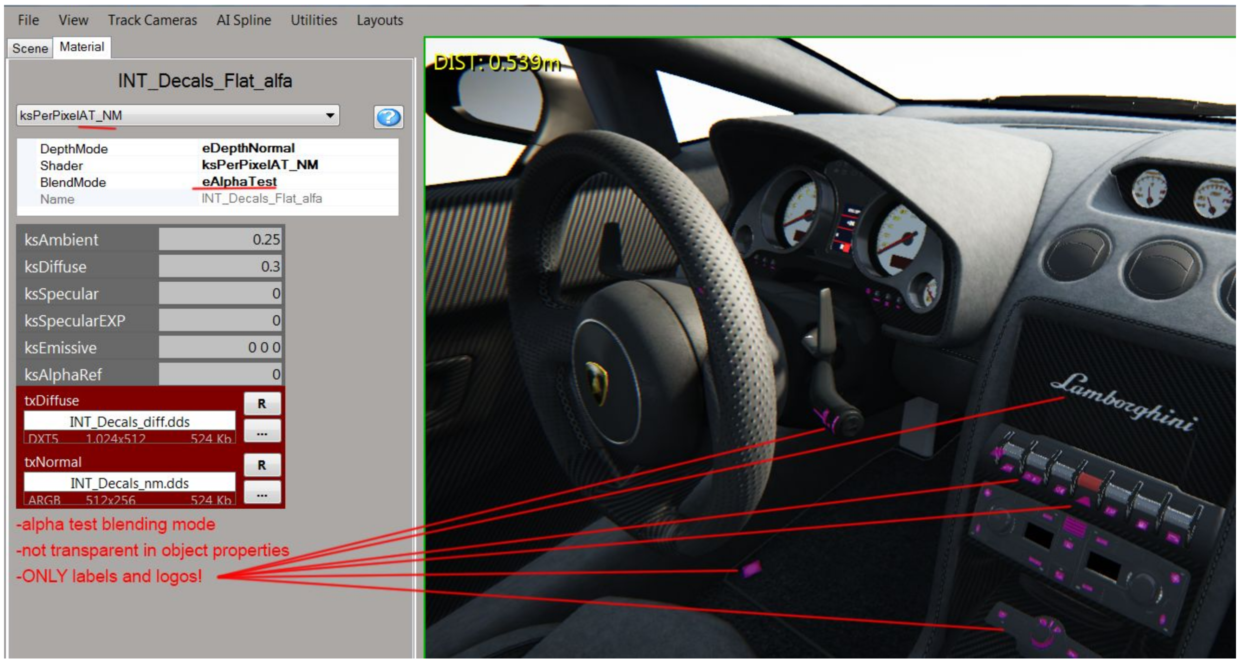
DepthMode	eDepthNormal
Shader	ksPerPixel
BlendMode	eOpaque
Name	INT_DISPLAY_CENTRE

ksAmbient	0
ksDiffuse	0
ksSpecular	0.2
ksSpecularEXP	35
ksEmissive	3.3 3.3 3.3
ksAlphaRef	0

txDiffuse	INT_DISPLAY.dds	FRR	512x512	786 Kb

- no ambient
- add emissive
- not transparent

CastShadow	True
Center	ksNet.kVector3f
IsActive	True
IsRenderable	True
IsTransparent	False



INT_Decals_Flat_alfa

ksPerPixelAT_NM

DepthMode	eDepthNormal
Shader	ksPerPixelAT_NM
BlendMode	eAlphaTest
Name	INT_Decals_Flat_alfa

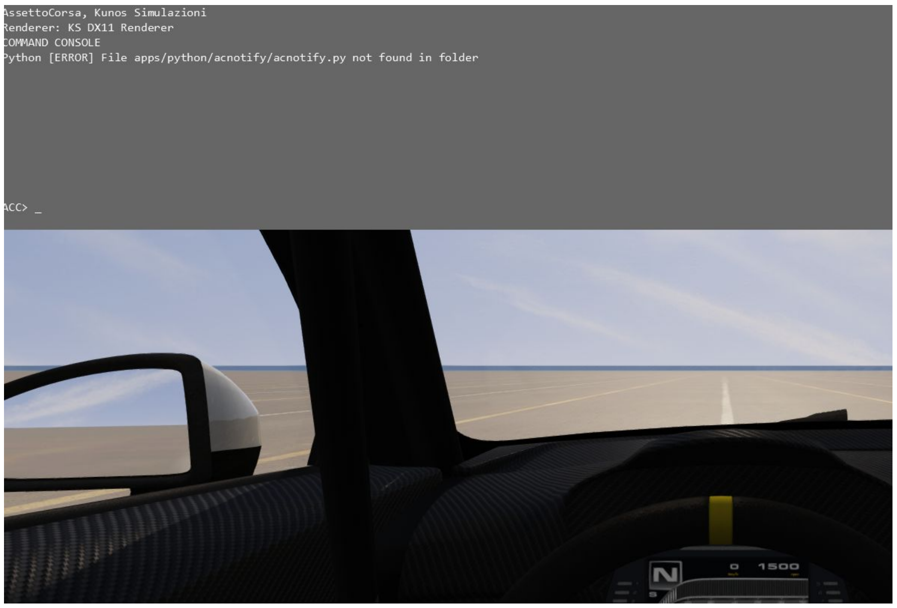
ksAmbient	0.25
ksDiffuse	0.3
ksSpecular	0
ksSpecularEXP	0
ksEmissive	0 0 0
ksAlphaRef	0

txDiffuse	INT_Decals_diff.dds	DXT5	1.024x512	524 Kb
txNormal	INT_Decals_nm.dds	ARGB	512x256	524 Kb

-alpha test blending mode
-not transparent in object properties
-ONLY labels and logos!

9. IN-GAME CONSOLE COMMANDS

Lights (front, rear, brakes), **Flames**, **Digital Instruments, Panels** and **Ground collider** can be adjusted real time in-game by using the built-in console commands.
Press **HOME** to open the in-game command window.



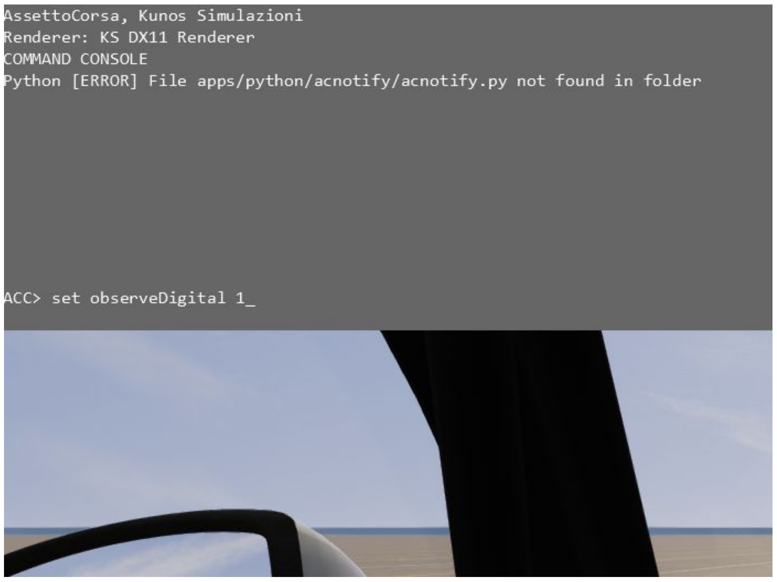
```
AssettoCorsa, Kunos Simulazioni
Renderer: KS DX11 Renderer
COMMAND CONSOLE
Python [ERROR] File apps/python/acnotify/acnotify.py not found in folder

ACC> _
```

Type one of the commands below to enter **edit** mode (case sensitive!).

- set observeLights [value]
- set observeFlames [value]
- set observeDigital [value]
- set observePanel [value]
- set editBoxes [value]

[Value] can be **1** (game will read the new values in the ini file) or **0** (game will not read the new values; this is the default value). The commands are CASE SENSITIVE. When set to 1, changes to the script in the respective text file will appear in the game real-time.



```
AssettoCorsa, Kunos Simulazioni
Renderer: KS DX11 Renderer
COMMAND CONSOLE
Python [ERROR] File apps/python/acnotify/acnotify.py not found in folder

ACC> set observeDigital 1_
```

Contributors:

Gergő Panker	- lead 3D artist, vehicle production managment @pankykapus
Gianluca Miragoli	- lead 3D artist, technical lead @yashugan
Manuel Darin	- staff programmer, beta testing management @6S.Manu
Marc Orphanos	- external 3D artist, driver rig @the_meco