This tutorial is not a modeling tutorial, see google for this. However we will build here a fictional tin-top in very low poly in order to review all aspects of rF2 car modding.
We'll see here how to split this car in various parts, and getting them to work correctly in rFactor2.

Polycount

Here is the detailled polycount of the rFactor2 Clio Cup.
Of course modders can increase these values as needed, keep in mind that rF2 cars were designed to consume as less as possible computer resources.

As measured with Polygon count Max tool :
Mx, H, M, L is for max, high, medium, low LOD.

- Tire Mx : 878
- Tire L : 470
- Rim Mx : 1379
- Rim L : 776
- Brake disc : 236
- Caliper : 216
- Susp : 187

- Body ext Mx : 5258
- Body ext H : 3397
- Body ext M : 1775
- Body ext L : 967
- Sleeves : 1080

- Windows : 503
- Windshield : 204

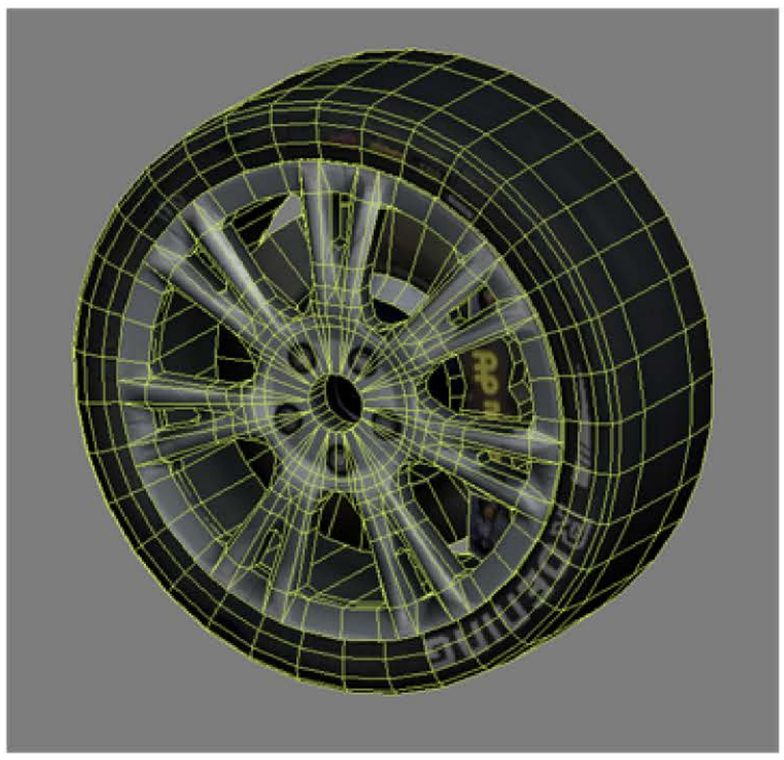
- Front ext Mx : 1954
- Front ext H : 1277
- Front ext M : 693
- Front ext L : 302

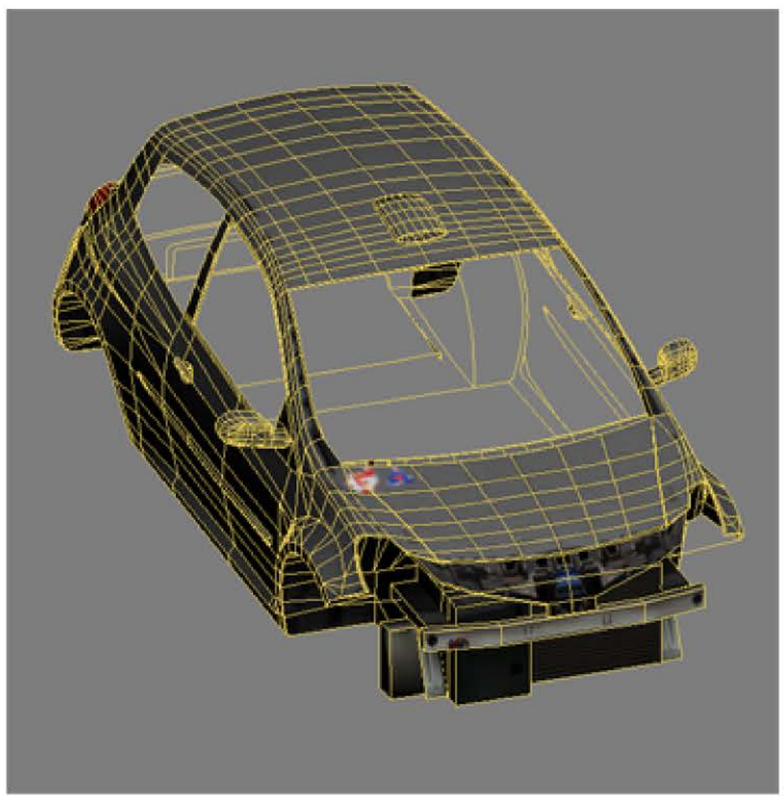
- Headlight : 313

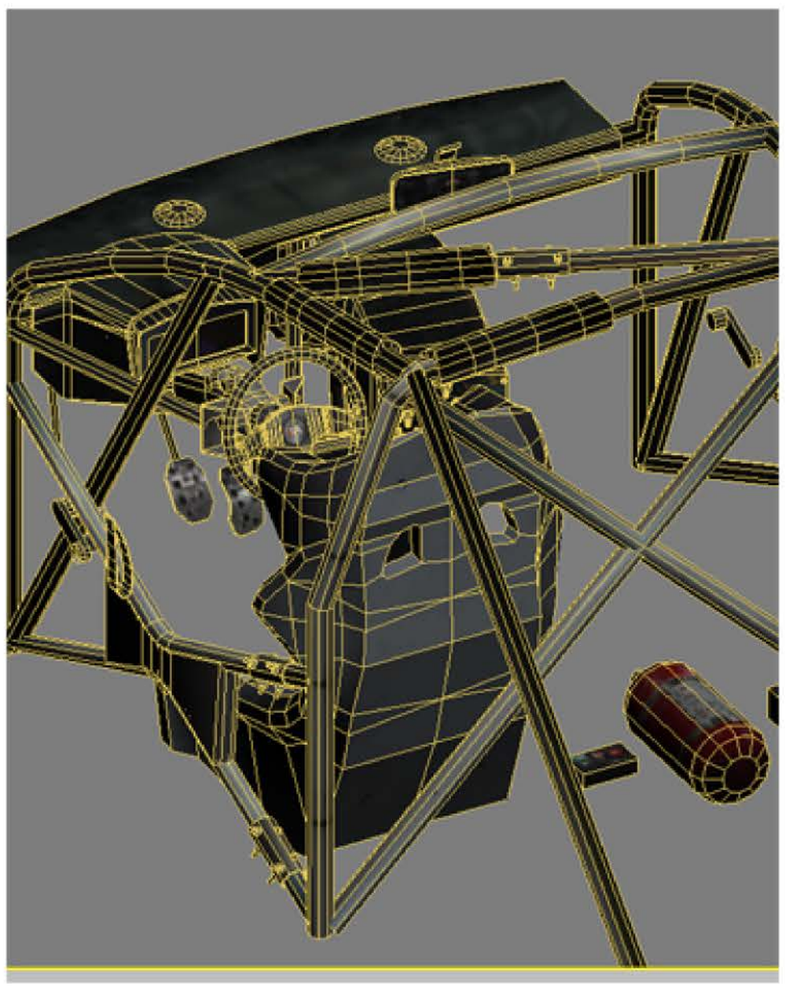
- Diffuser Mx : 1166
- Diffuser L : 697

- Rwing : 652

- Body in Mx : 1252
- Body in L : 716
- Rollcage Mx : 2394
- Rollcage H : 1268
- Rollcage L : 616
- Seat : 904
- Fishnet : 1548
- Dashboard Mx : 1074
- Dashboard L : 624
- Steering parts : 1196
- Big Cockpit parts : 1451
- Small Cockpit parts : 1146
- Cables : 1368
- Motec : 135
- Steering Wheel Mx : 2176
- Steering Wheel L : 1060







The .gen files and instances

We've seen in workflow tutorial that the .gen files are describing what meshes to load for the showroom scene (xxx_Spinner.gen) and on track scene (xxx.gen).
Here we will focus on the "on track scene", using our test_car.gen file.

An Instance is a paragraph describing what mesh (gmt format from Max gMotor exporter) is loaded, what rendering parameters apply and the distance view.
Each and every mesh you want to load ingame has to be mentioned in an Instance paragraph.
There is mandatory instances for a car as :

Instance=SLOT<ID>	car main body
Instance=COCKPIT	what shows in cockpit view
Instance=LFTIRE	left front wheel group
Instance=LRTIRE	left rear wheel group
Instance=RFTIRE	right front wheel group
Instance=RRTIRE	right rear wheel group

An instance paragraph has to be formed like this :

```
Instance=Example
{
  Moveable=True
  MeshFile=Example.gmt

}
```

All car instances have to be tagged as Moveable
Optional parameters for mesh :

CollTarget
ShadowCaster
LODIn
LODOut
Reflect
Render
Deformable

Settings for most car part :

MeshFile=xxx.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.0) LODOut=(15.0) Reflect=True

Tires are getting an extra parameter for deformation :

MeshFile=xxx_lftire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.0) LODOut=(15.0) Reflect=True Deformable=true

Collision mesh has to be tagged like this :

MeshFile=xxx_coll.gmt CollTarget=True HATTarget=False Render=False LODIn=(0.0) LODOut=(200.0)

A prefix can be added for the rendering engine graphic settings accordingly to LOD values :

<MAX> MeshFile=xxx_max.gmt LODIn=(0.0) LODOut=(10.0)
<HIGH> MeshFile=xxx_high.gmt LODIn=(10.0) LODOut=(20.0)
<MED> MeshFile=xxx_med.gmt LODIn=(20.0) LODOut=(30.0)
<LOW> MeshFile=xxx_low.gmt LODIn=(30.0) LODOut=(200.0)

In the example above, we are building 4 meshes, each one for a different level of detail. The LOD distances values are in meters and here we have :
- max detail object showing from 0 meters to 10 meters
- high detail object showing from 10 meters to 20 meters
- med detail object showing from 20 meters to 30 meters
- low detail object showing from 30 meters to 200 meters

So LOD can be set the same for all players if you don't use prefix, and depends on the graphic settings if you are using it.

An instance can gather as many as meshes you want, as long as they have the same exact pivot point. It is usually the case for car body instance, cockpit instance or tire instance.

Car body building

Remember that polycount and body shape are not the subject here, we will build a very low poly example car. The process itself is the only important thing.
For the modeling part, just google "low poly car modeling".
This tutorial is using the workspace from the workflow tutorial, don't forget to check this out.

Whatever is the car you want to build here is a few things :
- check your blueprint dimensions
- do all the modeling work first
- work with quads
- Align the bottom of the car with the Y max axis.
- start mapping and texturing the cockpit first, it is the less fun part but the most important as you will spend most of the time in the cockpit driving.
- be smart with the polycount, try to have a balanced level of detail. No need to have heavily detailed switches with a boxy dashboard or car interior.
- remember that during driving experience, 99% of the time you are looking at two car parts : cockpit (yours) and rear end (opponents).

Open Max file "test_car_complete.max".

• ***body materials***

- Open material Editor
- Rename top left mat as "test_car_body_mats"
- click "Standard" and pick "Multi/Sub-Object"
- delete all sub mats except first one
- click this material, "Standard" button, then select gMotorMaterial

We will use this first sub mat for the car body material.
- Load *Bump Cube Specular Map Add Alpha Reflect T1 lerp T2 Vertex Alpha* shader.
Five layers in this shader :
- color map (base body paint here)
- damage map (will replace body paint where it is crushed)
- specular map (shadows and lights)
- bump map
- cube map (environement reflection)

This body material has to be named "WCCARBODY"
This is one of the wild card materials that could swap its color map with a dedicated skin / map ingame. Here the color map will be replaced by the carbody skin.

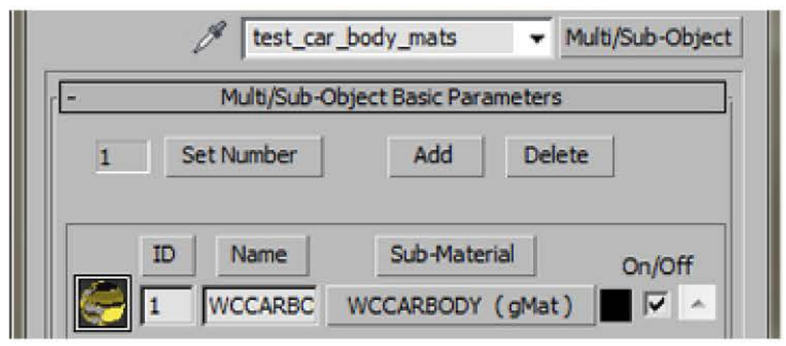
- Load maps :
color map = test_car_body.dds
damage map = test_car_body_damages.dds
specular map = test_car_s.dds
bump map = test_car_b.dds
cube map = CAR_CUBE_DX9.dds (The cube map must be tagged as Live Mapper.)

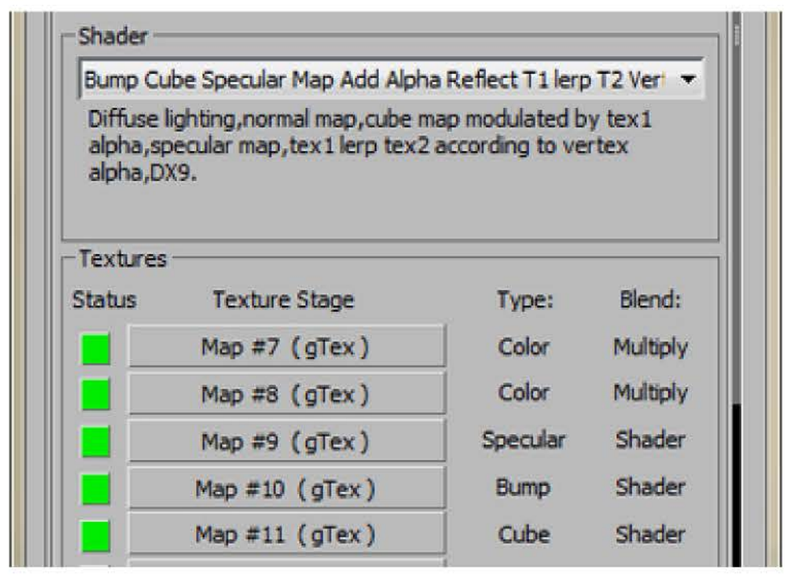
> The color map needs an alpha channel, it will modulate cube map intensity.
> The cube map will be automatically replaced ingame with live reflection.

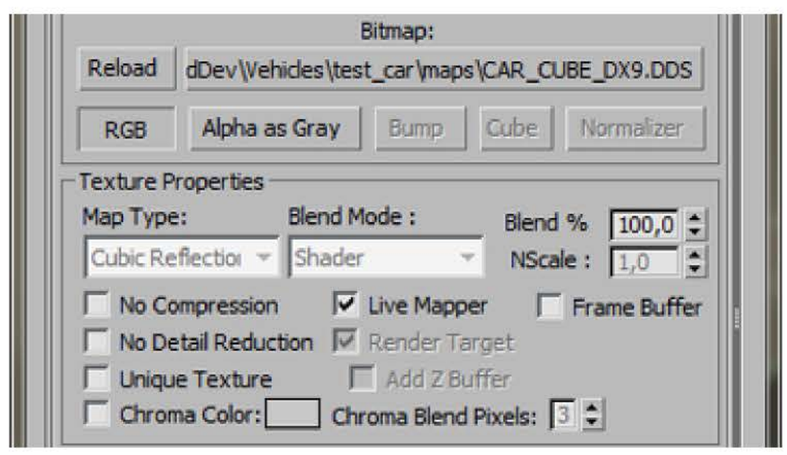
Now back to the main material setup
- enter REFMAP0 as Reflection Mapper name
- increase specular level as needed
- set specular color

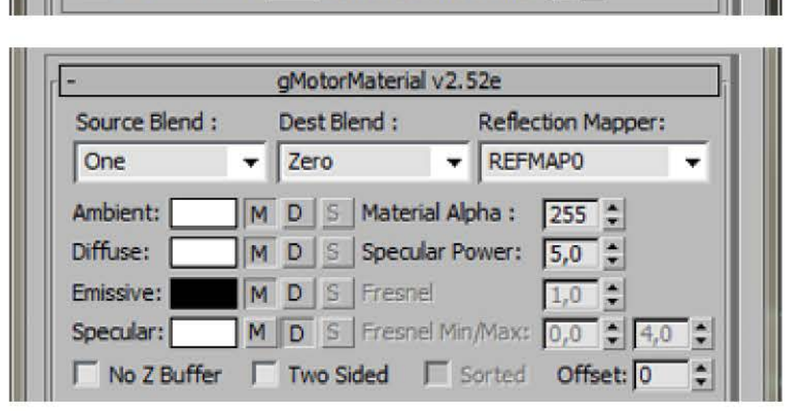
Drag and drop the test_car_body_mats material onto the test_car_body mesh.
In modify mode, select all polygons, scroll down to the Polygon:Material IDs tab and assign the 1 ID.

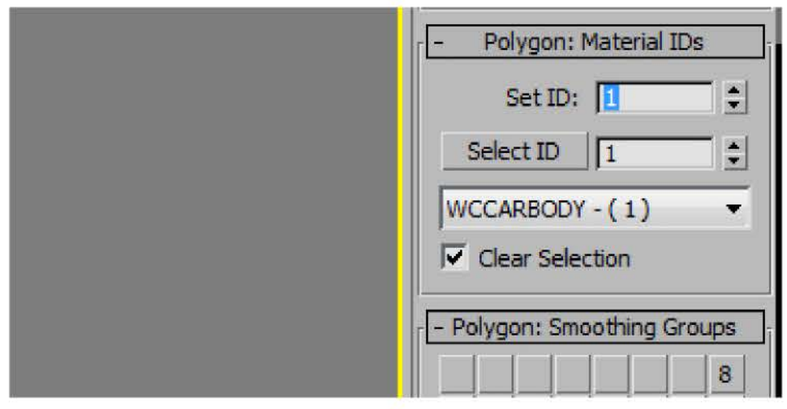
The car body mesh must have a pivot point located at 0,0,0. Select car body mesh go to Hierarchy tab, Affect Pivot Only, and set values to 0,0,0.







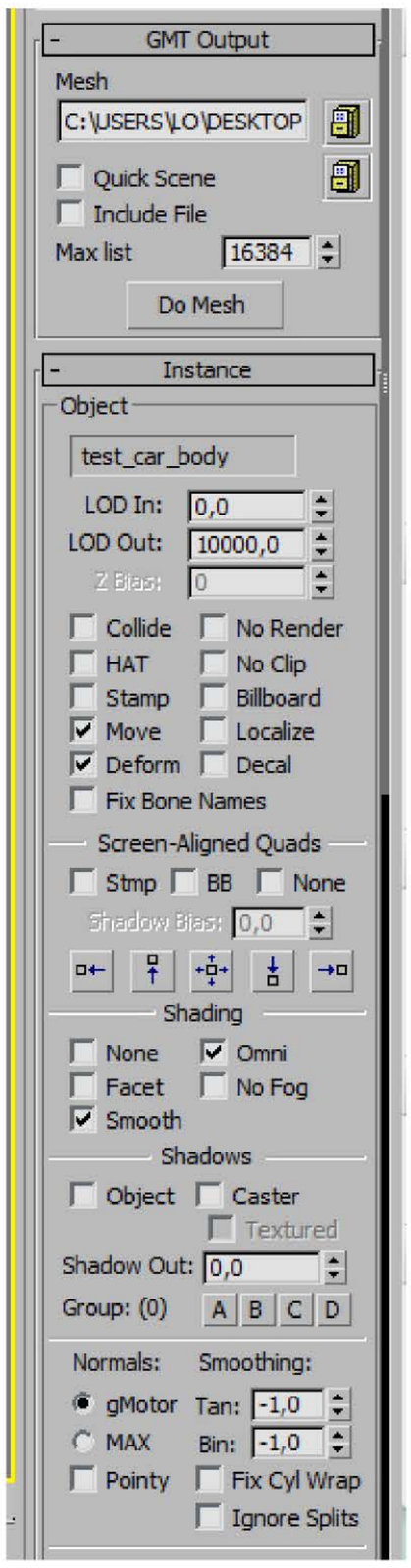




• ***checking car body ingame***

As seen in Workflow tutorial we can load a one mesh only car in spinner scene, this will allow us to check our car body ingame now.

- Pick GMT Converter in Utiities / More panel
With test_car_body mesh selected, use these settings :



Click Do Mesh button in GMT Output tab to export the mesh.
Now open the test_car_Spinner file, and replace the test_cube.gmt mesh file, with test_car_body.gmt like this :

```
Instance=SLOT<ID>
{
  Moveable=True
  MeshFile=test_car_body.gmt CollTarget=False HATTarget=False LODIn=(0.0) LODOut=(15.0) ShadowCaster=(Dynamic, Solid, 256, 256) Reflect=True

  Actor=VEHICLE
}
```

Launch rF2 in ModDev mode and load the Showroom scene of Test_Car :

• ***Getting minimal car ingame***

Now we will add all mandatory instance to be able to load the car on track.

- Instance=COCKPIT

Hide test_car_body in max, and unhide test_car_inside.
This mesh needs to have a proper material, the most common shader used for this is Color+Spec+Bump.

- Add a new sub mat in the main test_car_body_mats
- Name it : test_car_inside (fill both Name and Sub_material fields)
- Load gMotorMaterial and select Bump Specular Map T1
- Load test_car_inside.dds for Color
- Load test_car_inside_s.dds for Specular
- Load test_car_inside_b.dds for Bump

- Drag and drop material onto test_car_inside mesh
- select all faces
- Set material ID #2

All cockpit meshes need to have a pivot center set to 0,0,0.
Select mesh, Hierarchy tab, Affect Pivot Only, enter 0,0,0 values.
Open GMT Converter, use same settings as for carbody (page 4), except from Deform that should be unchecked, export mesh.

You would want to check exported meshes in Spinner scene as much as you build them.
Open test_car_Spinner.gen file and add a new instance for the cockpit mesh :

```
Instance=BODY_IN
{
  Moveable=True
  MeshFile=test_car_inside.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(20.0)   Reflect=True
}
```

You have to be careful how to insert instances after the main body instance, you have to insert them before latest } caracter, like this :

```
//--------------------------------------

Instance=SLOT<ID>
{
  Moveable=True
  MeshFile=test_car_body.gmt CollTarget=False HATTarget=False LODIn=(0.0) LODOut=(15.0) ShadowCaster=(Dynamic, Solid, 256, 256) Reflect=True

  Actor=VEHICLE



   Instance=BODY_IN
     {
      Moveable=True
      MeshFile=test_car_inside.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(20.0)   Reflect=True

     }

}
```

Once you have checked that the interior mesh is loading fine in Spinner scene, let's move on to the wheel / tire group.

- Instance=LFTIRE

Under this instance, you will have to place all objects that are rotating with the tire
- Tire itself
- Rim
- Brake disc
- Brake disc glow

All meshes need to have the same pivot coordinates, as they are rotating around the same pivot point.
- Unhide test_car_lf_tire, test_car_lf_rim, test_car_lf_bd meshes.
- Select test_car_lf_tire, Hierarchy, Affect Pivot Only, Center to Object. Note the pivot coordinates
- Select test_car_lf_rim, apply those values to pivot, test_car_lf_bd and apply those values too.

Wheel / tire materials
For a deeper explanation on tire material refer to the rF2_car_modding_tires.pdf tutorial.

To keep things organized, it is better to have a dedicated multi/sub material for the wheel ensemble.
As seen for body materials, create a new Multi/Sub-Object material, and delete all subs but the two first one.
Name the main Multi/Sub as test_car_wheels.
The tire needs 2 different mats, one for the tire tread, one for the tire wall.

- name #1 Sub mat as test_car_tire_tread
- Load gMat
- Load Bump Specular Map T1 Lerp All Vertex Alpha
First 3 maps are new tire material :
> Color map
- load generic_ttread.dds
- set Animation Source to Texture Maps
- set Animation Data to Manual
- set Name to generic_ttread.dds
- set 6 frames
- set Seq to (0,1,2,3,4,5)

> Specular map
- load generic_ttread_s.dds
- set Animation Source to Texture Maps
- set Animation Data to Manual
- set Name to generic_ttread_s.dds
- set 6 frames
- set Seq to (0,1,2,3,4,5)

> Bump map
- load generic_ttread_b.dds
- set Animation Source to Texture Maps
- set Animation Data to Manual
- set Name to generic_ttread_b.dds
- set 6 frames
- set Seq to (0,1,2,3,4,5)

Last 3 maps are damage tire material :
> Color map
- load generic_ttread_damage.dds

> Specular map
- load generic_ttread_damage_s.dds

> Bump map
- load generic_ttread_damage_b.dds

- name #2 Sub mat as test_car_tire_wall
- Load gMat
- Load Bump Specular Map T1 Lerp All Vertex Alpha
First 3 maps are new tire material :
> Color map
- load generic_twall.dds
- set Animation Source to Texture Maps
- set Animation Data to Manual
- set Name to generic_twall.dds
- set 6 frames
- set Seq to (0,1,2,3,4,5)

> Specular map
- load generic_twall_s.dds
- set Animation Source to Texture Maps
- set Animation Data to Manual
- set Name to generic_twall_s.dds
- set 6 frames
- set Seq to (0,1,2,3,4,5)

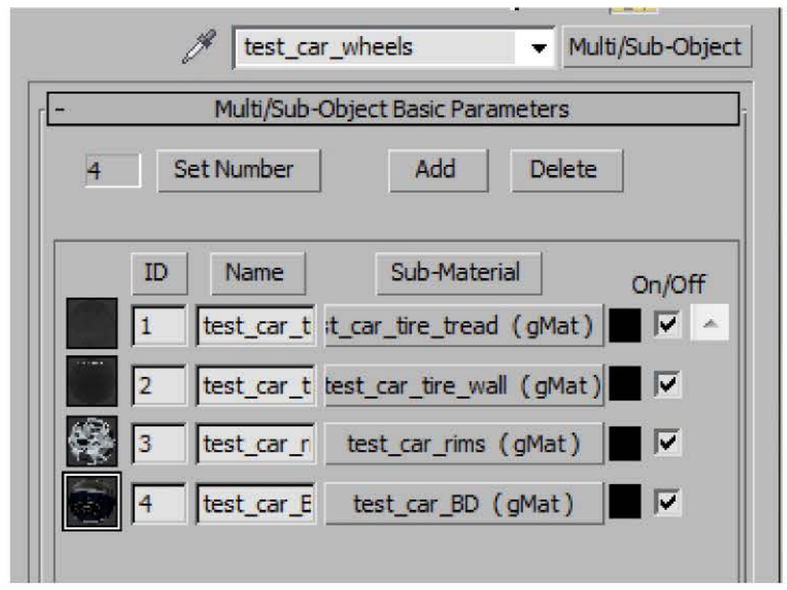
> Bump map
- load generic_twall_b.dds
- set Animation Source to Texture Maps
- set Animation Data to Manual
- set Name to generic_twall_b.dds

- set 6 frames
- set Seq to (0,1,2,3,4,5)

Last 3 maps are damage tire material :
> Color map
- load generic_twall_damage.dds

> Specular map
- load generic_twall_damage_s.dds

> Bump map
- load generic_twall_damage_b.dds



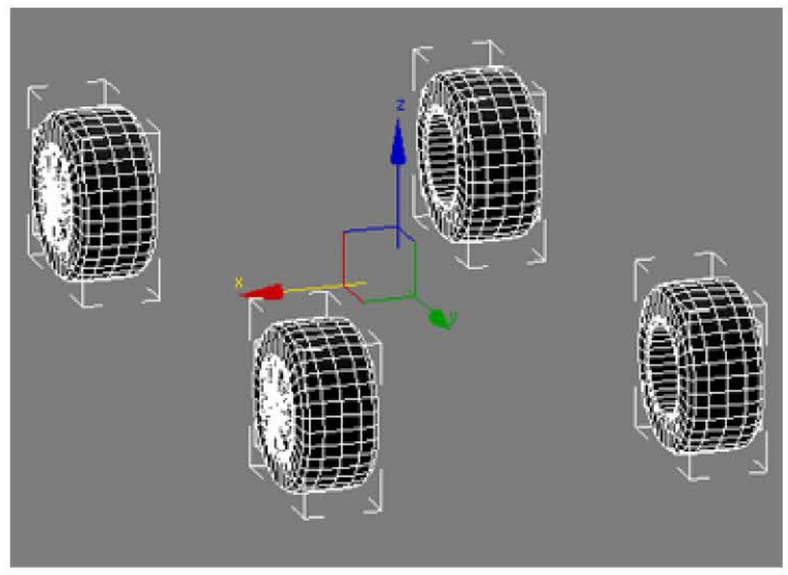
Drag and drop test_car_wheels main material onto test_car_lf_tire mesh.
Tire mesh is already mapped so tire tread is using mat ID#1 and tire walls are using mat ID #2
Remember to do the same thing with your own meshes.

Now we will add materials for rim and brake disc.
- Add a new sub mat to the main wheels Multi/Sub
- Name it test_car_rims
- Load gMotor material
- Load Bump Specular Map T1

> Color map
- load test_car_rim.dds

> Specular map
- load test_car_rim_s.dds

> Bump map
- load test_car_rim_b.dds



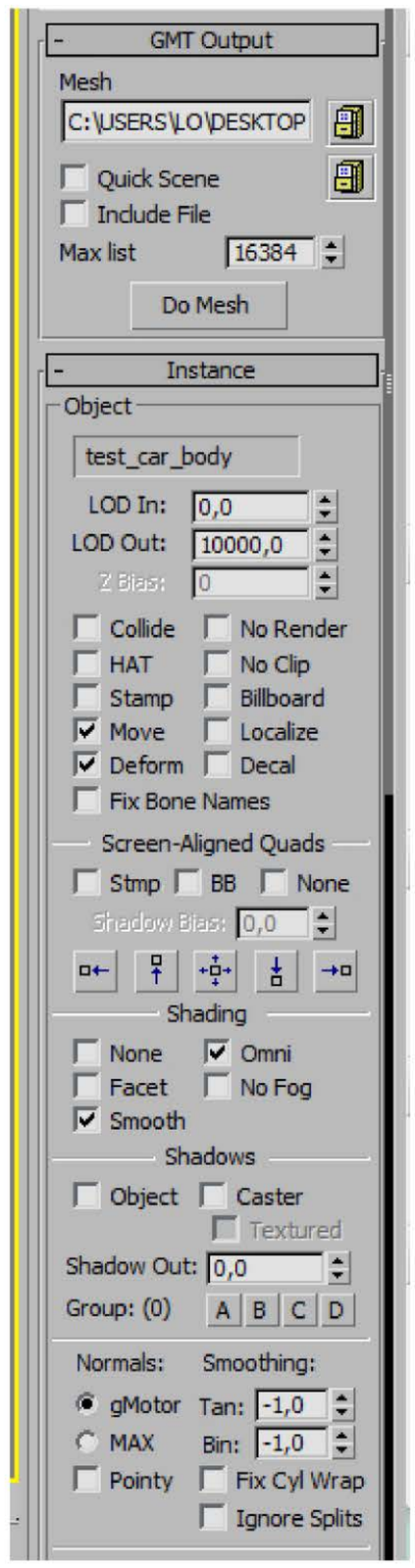
Drag and drop test_car_wheels main material onto test_car_lf_rim mesh.
Select rim mesh, select all faces and set mat ID to 3.

- Add a new sub mat to the main wheels Multi/Sub
- Name it test_car_BD
- Load gMotor material
- Load Specular Map T1

> Color map
- load test_car_BD.dds

> Specular map
- load test_car_BD_s.dds

Drag and drop test_car_wheels main material onto test_car_lf_BD mesh.
Select rim mesh, select all faces and set mat ID to 3.



Select the 3 meshes from the left front wheel (test_car_lf_tire + test_car_lf_rim + test_car_lf_BD) and clone them to build right front wheel. Rename the 3 new meshes to test_car_rf_tire + test_car_rf_rim + test_car_rf_BD.
Select the two front wheels groups and clone them to build rear wheels.
rename the new meshes to :
- test_car_rr_tire + test_car_rr_rim + test_car_rr_BD (for right rear group)
- test_car_lr_tire + test_car_lr_rim + test_car_lr_BD (for left rear group)

Select all wheels elements (12objects), in GMT Converter / Instance tab tag the meshes (Move+Omni+Smooth+Normals gMotor).
Export them.

Adding new instances to .gen files

- Open test_car_Spinner.gen
Right after the body_in instance add these lines :

```
//--------------------------------------- TIRES

 Instance=LFTIRE
 {
  Moveable=True
  MeshFile=test_car_lf_tire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
  MeshFile=test_car_lf_rim.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
  MeshFile=test_car_lf_bd.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(80.0) ShadowReceiver=True Reflect=True

 }

Instance=RFTIRE
{
 Moveable=True
 MeshFile=test_car_rf_tire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_rf_rim.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_rf_bd.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(80.0) ShadowReceiver=True Reflect=True
}

Instance=LRTIRE
{
 Moveable=True
 MeshFile=test_car_lr_tire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_lr_rim.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_lr_bd.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(80.0) ShadowReceiver=True Reflect=True

 }

Instance=RRTIRE
{
 Moveable=True
 MeshFile=test_car_rr_tire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_rr_rim.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_rr_bd.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(80.0) ShadowReceiver=True Reflect=True

 }
```

- Open test_car.gen
delete everything after the MASFile=cmaps.mas line.

Copy and paste all instances from Spinner.gen file.
We must have a cockpit instance in the .gen file, with all objects supposed to show in cockpit view.
Just copy and paste the BODY_IN instance, and rename it COCKPIT, see next page for the complete test_car.gen file.

Your car should now load in showroom and on track.

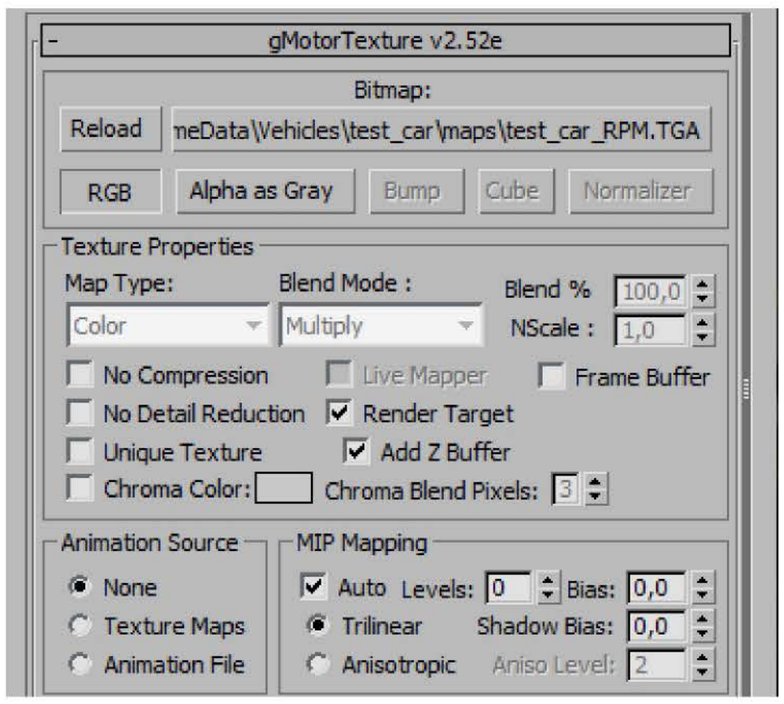
```
//---------------------------------------

SearchPath=<VEHDIR>

SearchPath=test_car
SearchPath=<VEHDIR>test_car\maps
SearchPath=<VEHDIR>cmaps_maps
SearchPath=test_car\maps
MASFile=test_car\test_car.mas
MASFile=cmaps.mas

//---------------------------------------


Instance=SLOT<ID>
{
 Moveable=True
 MeshFile=test_car_body.gmt CollTarget=False HATTarget=False LODIn=(0.0) LODOut=(15.0) ShadowCaster=(Dynamic, Solid, 256, 256) Reflect=True

 Actor=VEHICLE


  Instance=COCKPIT
   {
    Moveable=True
    MeshFile=test_car_inside.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(2.0)   Reflect=True

   }


  Instance=BODY_IN
   {
    Moveable=True
    MeshFile=test_car_inside.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(20.0)   Reflect=True

   }

//---------------------------------------- TIRES

 Instance=LFTIRE
{
 Moveable=True
 MeshFile=test_car_lf_tire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_lf_rim.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_lf_bd.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(80.0) ShadowReceiver=True Reflect=True

}

Instance=RFTIRE
{
 Moveable=True
 MeshFile=test_car_rf_tire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_rf_rim.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_rf_bd.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(80.0) ShadowReceiver=True Reflect=True
}

Instance=LRTIRE
{
 Moveable=True
 MeshFile=test_car_lr_tire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_lr_rim.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
 MeshFile=test_car_lr_bd.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(80.0) ShadowReceiver=True Reflect=True

 }

  Instance=RRTIRE
  {
   Moveable=True
   MeshFile=test_car_rr_tire.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
   MeshFile=test_car_rr_rim.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(20.0) ShadowReceiver=True Reflect=True
   MeshFile=test_car_rr_bd.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid)  LODIn=(0.00) LODOut=(80.0) ShadowReceiver=True Reflect=True

  }
}
```

Now we have to adjust the cockpit eyepoint.
- Open test_car_cockpitinfo.ini.
- First line, enter these values :

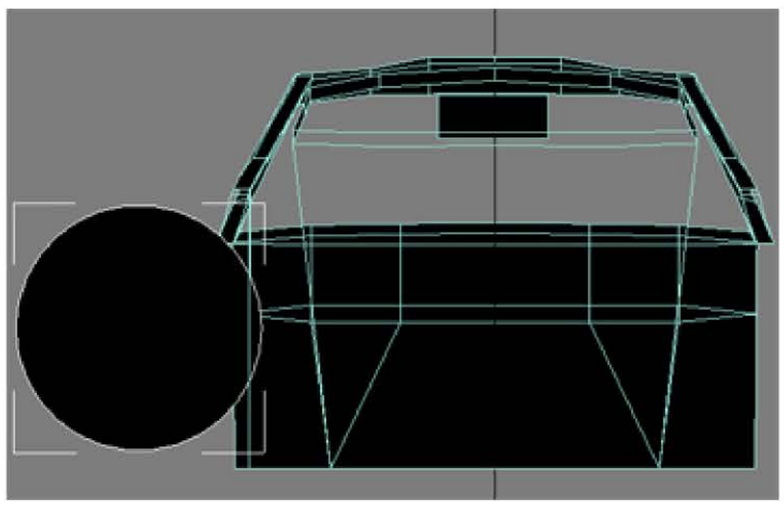
```
Eyepoint=(0.350, 0.945, -0.10)
```

Cockpit objects
There is several objects with special materials visible only in cockpit view, as gauges, motec and all information lights.

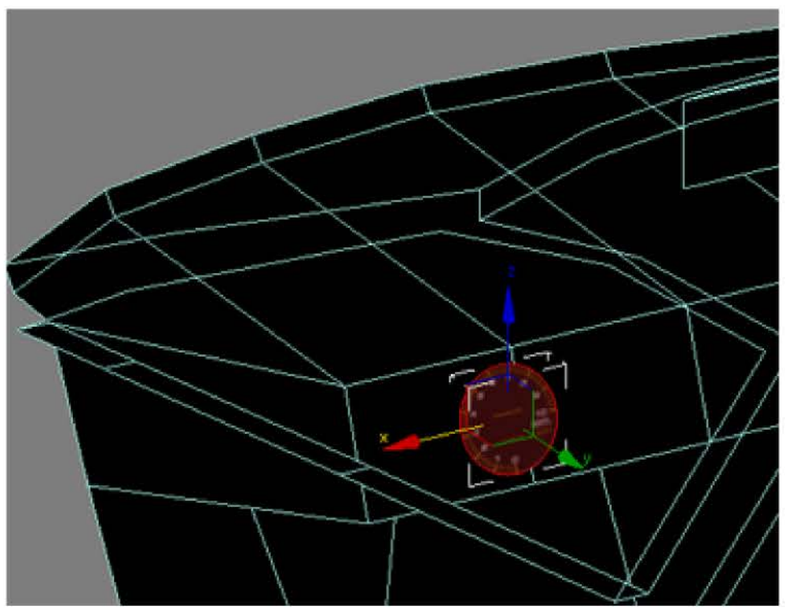
• Gauges
Rpm, speed and various temperature gauges are working the same way
- one dedicated background map in .tga format for each gauge.
- one needle map in .tga format for needle.



- Open material editor, add a new sub mat to the main car body multi sub.
- Name it test_car_RPM (there is no mandatory naming here)
- load a T1 gMat
- load test_car_RPM.tga in color slot
- tag the map as Render Target



- use max Back view
- create a cylinder, 30 Sides, 1 Height Segment
- convert it to Editable poly
- Delete all but the the polygon facing you



- drag and drop main body mat onto the new object
- select face
- select mat ID#3
- rotate, move and resize this object to make it fit the dashboard

- rename this object to test_car_gauges
- set pivot point to 0,0,0

- export it with GMT Converter (Move, Omni)

We have now to add it to .gen file :

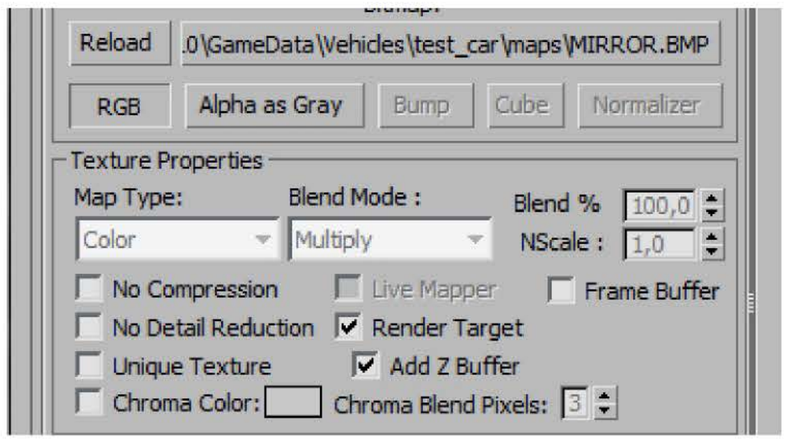
```
Instance=COCKPIT
 {
  Moveable=True
  MeshFile=test_car_inside.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(2.0)   Reflect=True
  MeshFile=test_car_gauges.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(2.0)
 }
```

And we have to edit cockpitinfo.ini file like this :

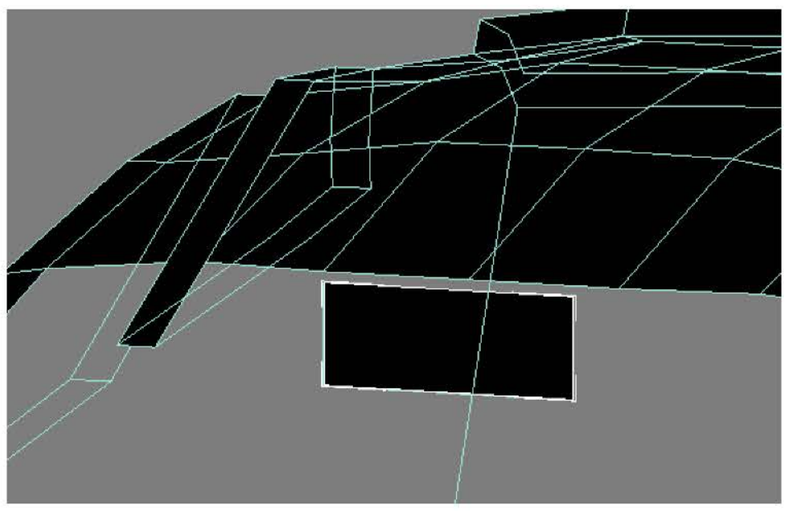
```
TachometerRange=(100, 10000, 323, 420)    // TachometerRange=(<minvalue>, <maxvalue>, <beginangle>, <endangle>)
TachometerBackground=test_car_rpm.tga
TachometerNeedle=(1.5,1.5)
TachometerNeedlemap=test_car_SNEEDLE.tga
RelativeTachometer=0                // Ignores <maxvalue> and uses default rev limit instead (which makes it work better for upgrades)
```

To get any gauge working :
- you have to define the background according to the name of the map you used in gMat gauge background
- you have to assign a needle map
- Add it to cockpit instance
- Add it to cockpitinfo.ini file

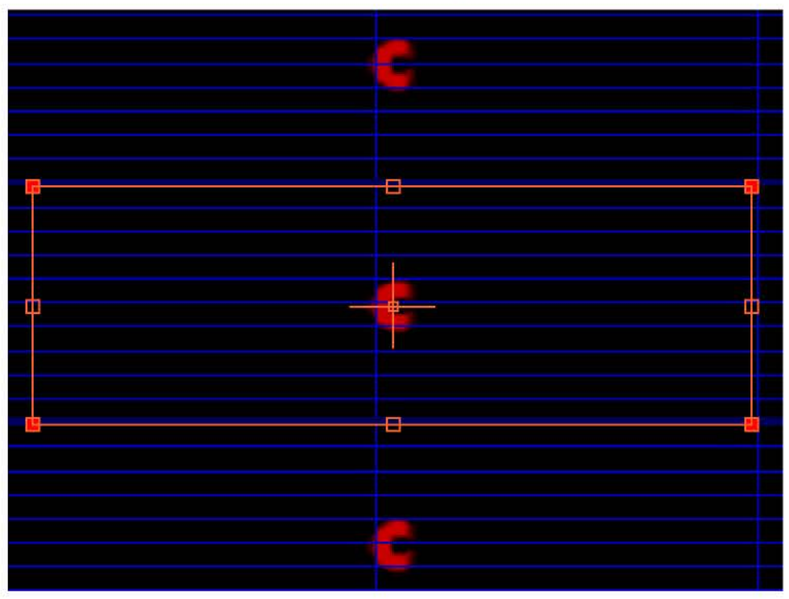
• **Mirrors**



Material :
- Add a new sub mat to main body multi sub.
- name it MIRROR (mandatory naming)
- load a T1 gMat
- load MIRROR.bmp map in color slot
- tag the map to render target



Mesh :
- Select the test_car_inside mesh
- duplicate the back facing poly of the rear view mirror
- move it back from a few mm (to be sure it will be on top of the inside mesh)
- rename it to test_car_mirror
- be sure the pivot point is still 0,0,0
- drag and drop main body mat onto it
- select face and assign mat ID#4



- In Modify/Vertex mode :
- Add a Unwrap UVW modifier
- adjust UVW using MIRROR.BMP map in background,
C letter on the map is center of the view.

- Rename the object to test_car_mirror
- export it with GMT Converter (Move, Omni)

- Open test_car.gen file
- Add the new object to cockpit instance :

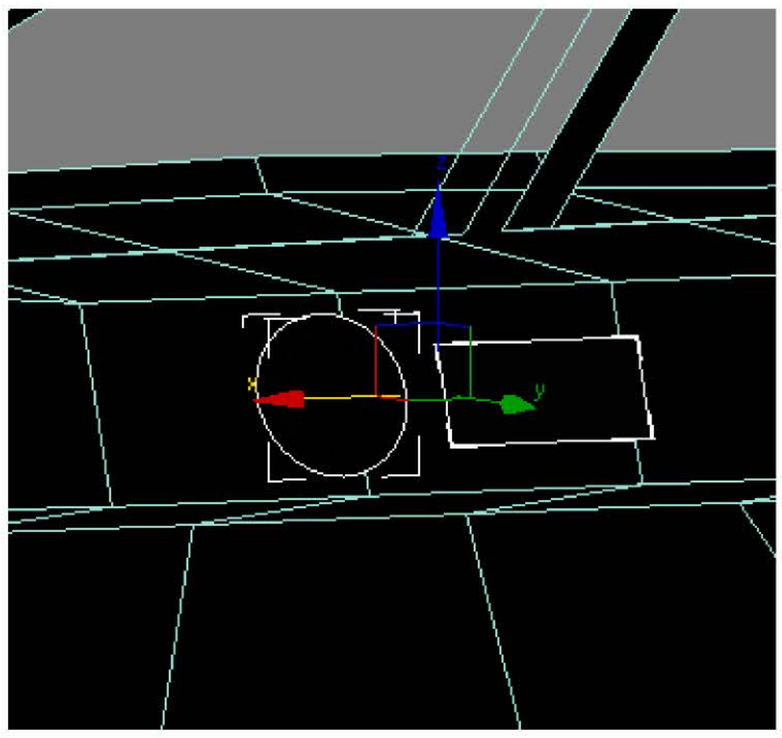
```
Instance=COCKPIT
 {
  Moveable=True
  MeshFile=test_car_inside.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(2.0)   Reflect=True
  MeshFile=test_car_gauges.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(2.0)
  MeshFile=test_car_mirror.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(2.0)
 }
```

Check if the new mirror is working in game, then add side mirrors :

- clone the rear view mirror face (do not clone object, just faces), adjust it to match car body side mirrors
- adjust UVW using MIRROR.BMP map in background, L is for left side of view, R for right side
- Export test_car_mirror mesh again

• Digital gauges
It is working the same way as analogic gauges, we are rendering information on a background map.
Each information rendered (rpm, speed, temperature) needs its own unique background and a font file.
The example below if for speed display.

Material :
- Add a new sub mat to main body multi sub.
- name it test_car_speedbkg
- load a T1 gMat
- load SPEEDBKG.tga map in color slot
- tag the map to render target



Mesh :
- in back view
- create a simple plane (1x1 seg)
- convert it to editable poly
- drag and drop main body mat onto it
- select face and assign mat ID#5
- resize, rotate and move the poly and place it at the rpm gauge righ side.
- select the test_car_gauges object and attach the new plane to it
- Export test_car_gauges mesh again

- Open test_car_cockpitinfo.ini file
Add these lines :

```
SpeedFont=test_car_font.bmp             // defaults to FONTREAD
SpeedBackground=test_car_speedbkg       // defaults to READOUT.BMP
SpeedScale=(1.0,1.0)                    // how big to draw the font onto the background
```

You can repeat the exact same operations for :

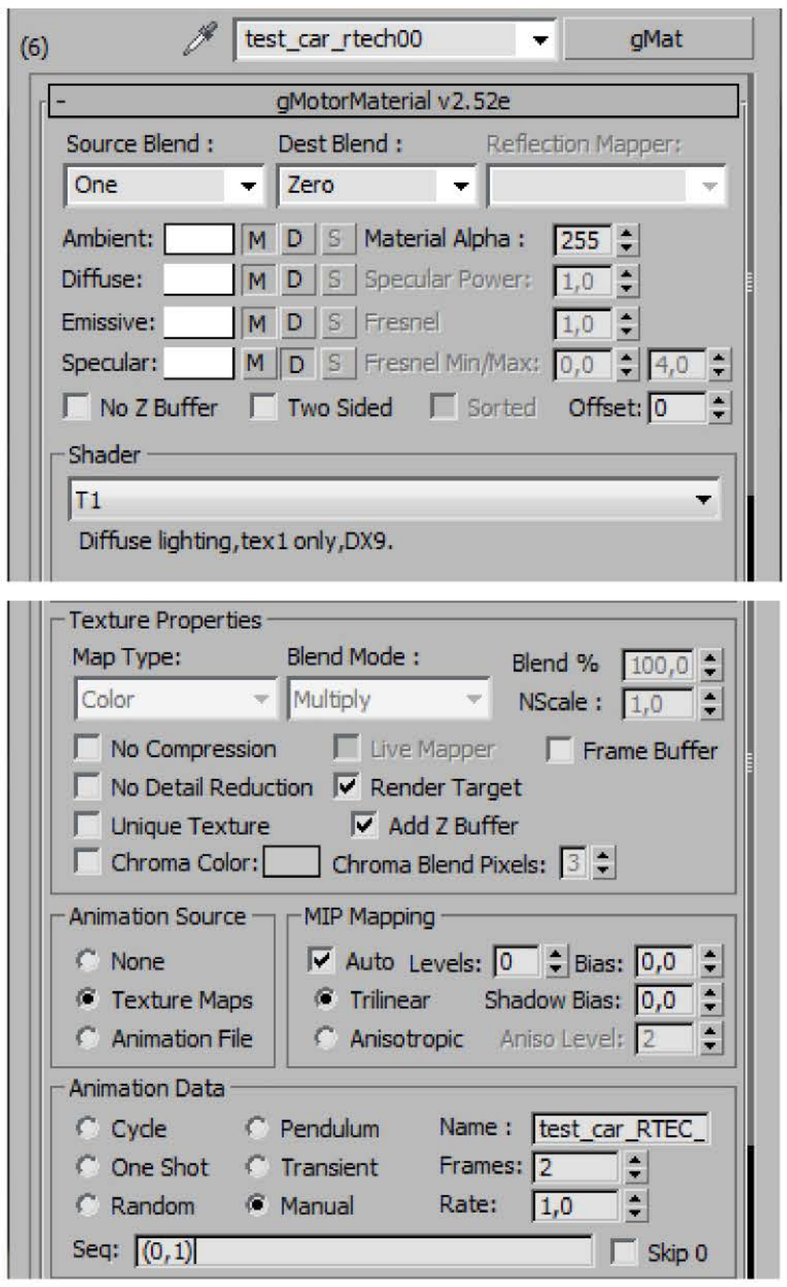
```
RPMFont=
RPMBackground=
RPMScale=

OilTempDigitFont=
OilTempDigitBackground=
OilTempDigitScale=

WaterTempDigitFont=
WaterTempDigitBackground=
WaterTempDigitScale=

FuellevelFont=
FuellevelBackground=
FuellevelScale=

GearsFont=
GearsBackground=
GearsScale=

PlaceFont=
PlaceBackground=
PlaceScale=

LapsFont=
LapsBackground=
LapsScale=
```

You can also diplay the full HUD LCD information onto one map, use same method with these entries :

```
LCDFont=
LCDBackground=
LCDScale=
```

• Digital RPM bars

To make a digital RPM bars display, we have to use one material for each bar.
Each material will be triggered for a given RPM value.
Each material is a 2 maps animation, showing map1 when idle, showing map2 when desired RPM value is reached.
Those maps can use alpha transparency and have to be in .tga format.
From 0 RPM to max (engine redline of your car), there is 31 materials available.

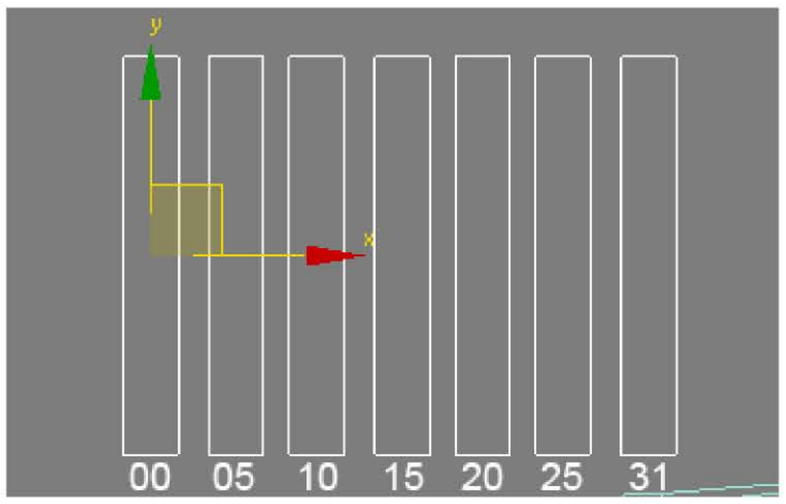


- load test_car_RTEC_BAR.TGA in color slot
- tag the map Render Target
- Set Animation Source to Texture Maps
- Set Animation Data to Manual
- Enter test_car_RTEC_BAR.TGA Name
- Set 2 frames
- Enter (0,1) Seq

Back to multi/sub level
- add a new sub mat
- copy previous mat (test_car_rtech00) onto it
- rename it test_car_rtech05
- repeat for :

test_car_rtech10
test_car_rtech15
test_car_rtech20
test_car_rtech25
test_car_rtech31

Mesh :
- in back view
- create a simple plane (1x1 seg)
- convert it to editable poly
- duplicate face to get a 7 bars poly
- drag and drop main body multi sub onto it
- select faces from left to right, assign one rtech mat for each one as indicated here -->



- resize, move and rotate RPM bars to place it on the left side of our gauges object
- select test_car_gauges mesh and attach new rpm bars object to it
- export test_car_gauges again

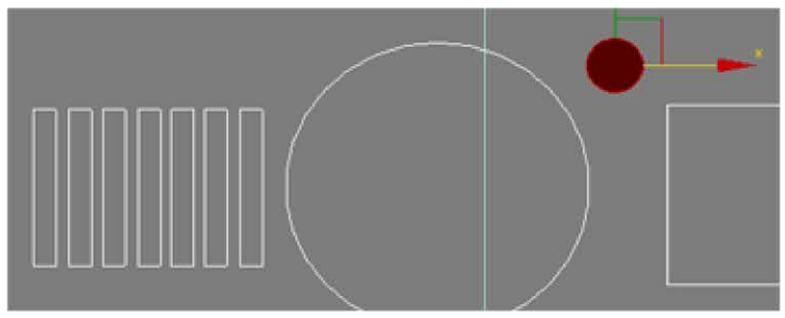
- Open cockpitinfo.ini file and add this line :

RPMLED=test_car_rtech

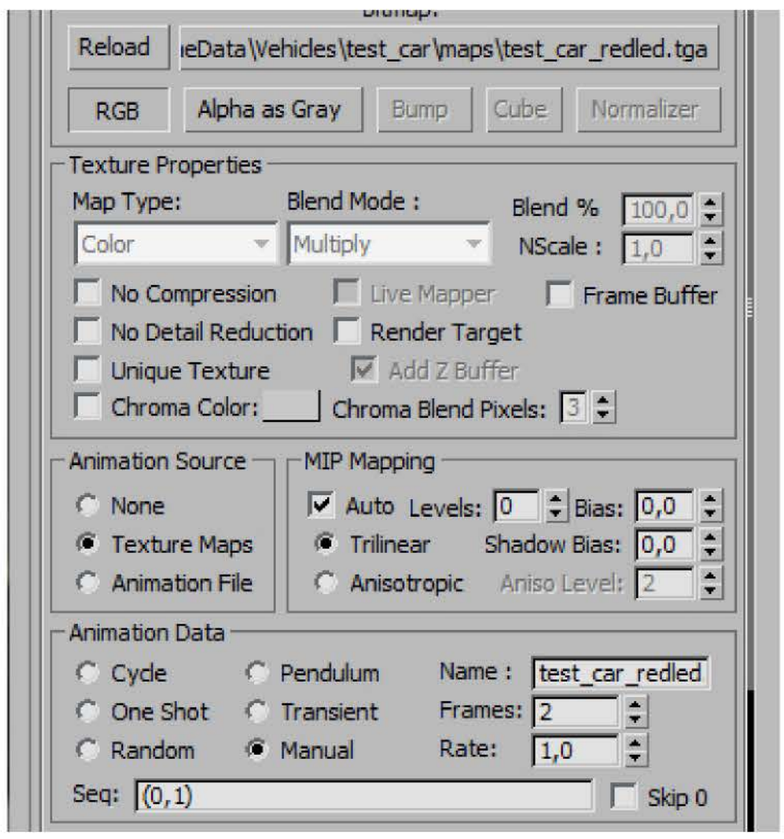
• **Warning lights**
Warning lights are two frames animations with mandatory material name.
Here is the materials list :
LED165 = rpm light
LED170 = rpm light
LED175 = rpm light
LED182 = rpm light (redline)
LEDSL = Overheating/Temperature Warning.
LEDPSR = Pit Stop Requested (Pending).
LEDPSY = Pit Stop Accepted (Yes/Go for pit stop).
LEDLF = Low Fuel
LEDFR = Black Flag
LEDFY = Yellow Flag
LEDFB = Blue Flag

For the example, we will build red warning light for pit stop request.

- Select the analog rpm gauge plane, duplicate it and resize it to make a warning light size poly.



- Add a new sub mat to main body multi sub.
- name it LEDPSY
- load a T1 gMat
- set Emissive color to 100% white
- load test_car_redled.tga in color slot
- tag the map Render Target
- Set Animation Source to Texture Maps
- Set Animation Data to Manual
- Enter test_car_redled.tga Name
- Set 2 frames
- Enter (0,1) Seq

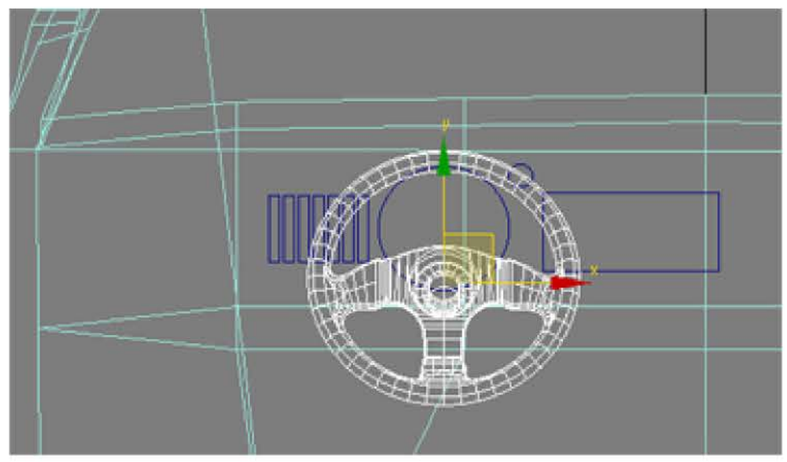


- Select the warning light poly, assign mat ID#13 (LEDPSY) to it.
- Export test_car_gauges again

- Once ingame hit the “S” key to request a pit stop, the red light should switch on.

Now you can test other warning lights, just copy the LEDPSY mat and rename it to another name on the list below. Assign each one of these new mats to a new poly.

• Steering wheel
Steering is an object of any shape, that is rotating around its pivot point.



- Unhide test_car_swheel object
- adjust its position

- Add a new sub mat to main body multi sub.
- name it test_car_swheel
- load a Bump Specular Map T1 gMat
- load test_car_swheelN.dds to Color slot
- load test_car_swheel_s.dds to Specular slot
- load test_car_swheel_b.dds to Bump slot
- drag and drop main body multi sub onto swheel mesh
- select faces and assign test_car_swheel sub mat
- Export test_car_swheel with GMT Converter (Move, Omni, gMotor Normals)

Open test_car.gen file
Add this instance :

```
Instance=WHEEL
{
 Moveable=True
 MeshFile=test_car_swheel.gmt CollTarget=False HATTarget=False ShadowCaster=(True, Solid) LODIn=(0.00) LODOut=(20.0)   Reflect=True

}
```

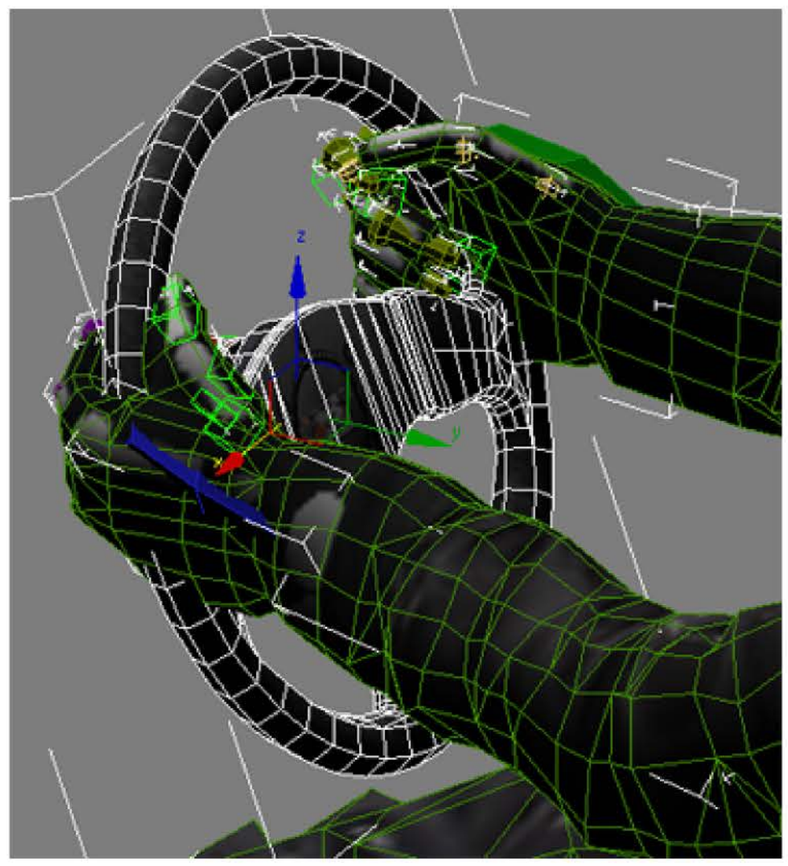
Instance name (WHEEL) is mandatory.

• Driver
Driver is splitted in two parts :
- body
- head (helmet)

Body is rigged to a Biped skeleton and can be animated.
Head is a simple mesh with an object centered pivot point

Body
Hide all objects
Unhide :
- all object starting with “Bip01”
- rF2_generic_driverB
- test_car_swheel

The body is rigged and placed in driving position in a 1 frame animation.
Adjust the steering position to exactly match hands position.



The driving animation length (hands and arms matching steering wheel rotation) has to be calculated from the .hdv file.
1 degree of rotation = 1 frame in Max by default.
- Open the test_car.hdv file
- Locate the line :

[CONTROLS]
TurnsLockToLock=1.5

It means that the steering range is 1.5xcomplete rotation (360°)
360x1.5=540
So the max animation must cover at least 540°

There is another value to check out in cockpitinfo.ini file
- Open the test_car_cockpitinfo.ini file
- Locate this line :

SteeringDegreesPerFrame=2.0

You can multiply/divide the Max animation length. Value is here 2, it will allow us to have a shorter Max animation.
Animation is a linear movement starting from frame 0 (left steer lock) to last frame (right steer lock), with neutral hands position (driving straight line) in the middle.
If the animation was 6 frames :

0 1 2 **3** 4 5 **6**
Left lock angle=0 Righ lock

It means we need a symetric animation, with an even number of frames.
The way max animations are working, you don't have to build every frame of the animation, just key frames every x frames. You can use a keyframe every 10 frames for a half-length animation. So the number of frames must be multiple of 10 too.

Back to our 540° animation :
- Half is 270
- We have to split it in 2 to have half-animation (from left lock to neutral)
- it makes 135 frames
As we want to work with 1 keyframe every 10 frame it would be better to work with a multiple of 10, just easier to handle.
So we will jump to the next multiple of 10, here it is 140.
0----------140----------280
Left Center Right

We will have an extra 10 frames in the animation that will be not used, but it won't make any difference ingame. (those extra frames are skipped).

It may sound confusing at first, but the calculation is easy, here with 670° range :
- real range = 670
- rough max animation = 670/2 = 335
- next 10 multiple = 340
- half animation = 170 **Ok for keyframe each 10 frame**
- final max animation length = 340
0----------170----------340
Left Center Right

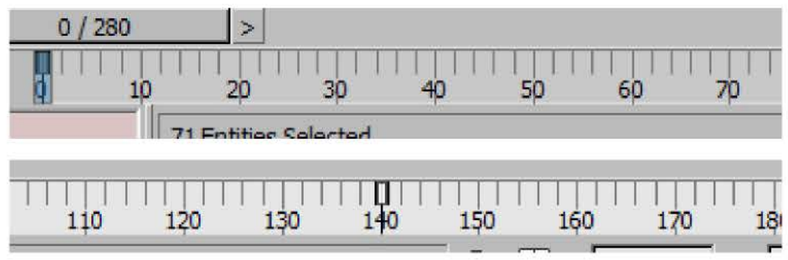
- real range = 610
- rough max animation = 610/2 = 305
- next 10 multiple = 310
- half animation = 155 **Not Ok for keyframe each 10 frame**
- next 10 multiple = 160
- final max animation length = 320
0----------160----------320
Left Center Right

When working with a boned / rigged object, you want to move bones, not the mesh itself. So :
- select rF2_generic_driverB mesh and Freeze it.
- select test_car_swell and freeze it too

Now we have to add the needed frames to the animation time-line.
- with ctrl+alt+right click, drag the time line until it ends at frame 280

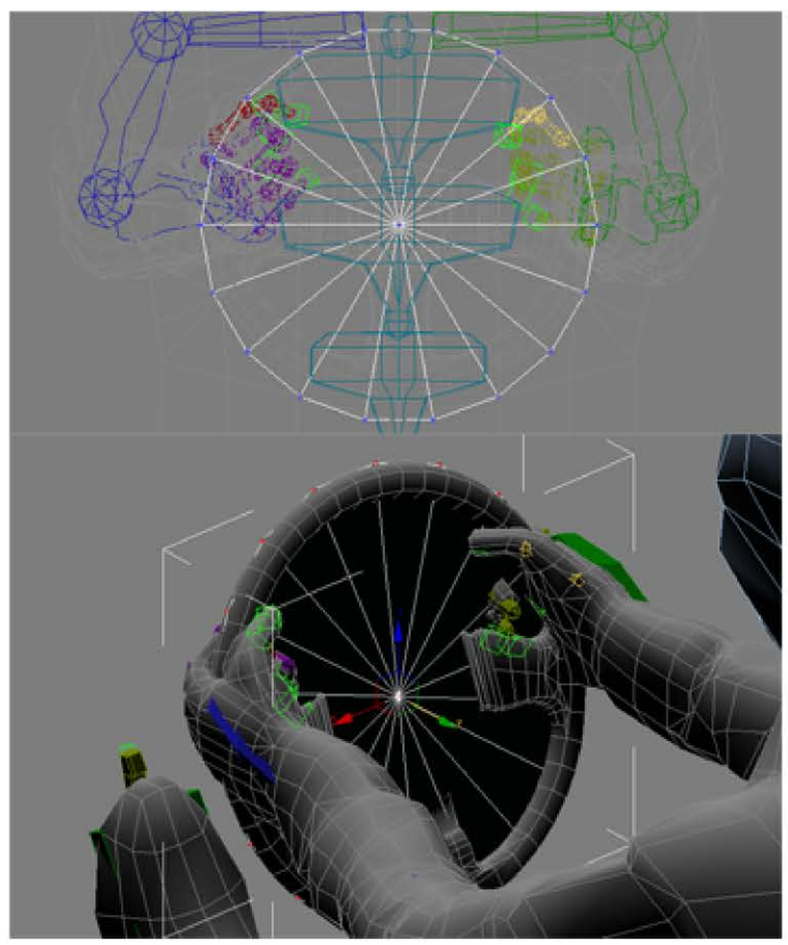
Currently, the neutral position is located at frame 0, we want it to be at frame 140
- select all biped elements (you should have only this available to selection in the scene)
- the keyframe at frame 0 is now showing

- select this keyframe in timeline, and deplace it to frame 140



Now, every 10 frames we have to move hands by 20° around the wheel (1 frame=2°)
We need visual aid for this.
During all the process of building driving animation we only need 2 views :
back and orthographic.

- in back view, create a cylinder (1,1,18 sides), centered on the steering wheel, with the same size.
- convert it to editable poly, remove all faces but the one facing back view
- slice this face drawing diameters from all vertex



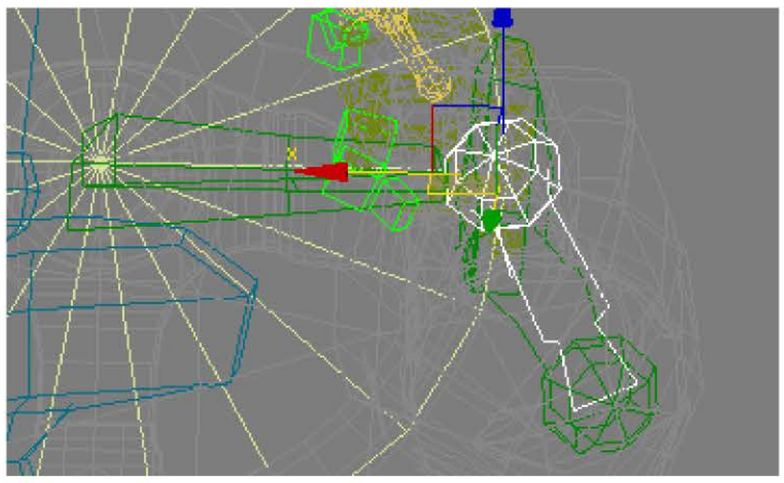
- move and rotate it to match exactly sterring wheel position, diameter and angle.
- assign bright display color to the poly (top right, next to the mesh name)

This will be our guide to move the hand bones, with a line every 20°

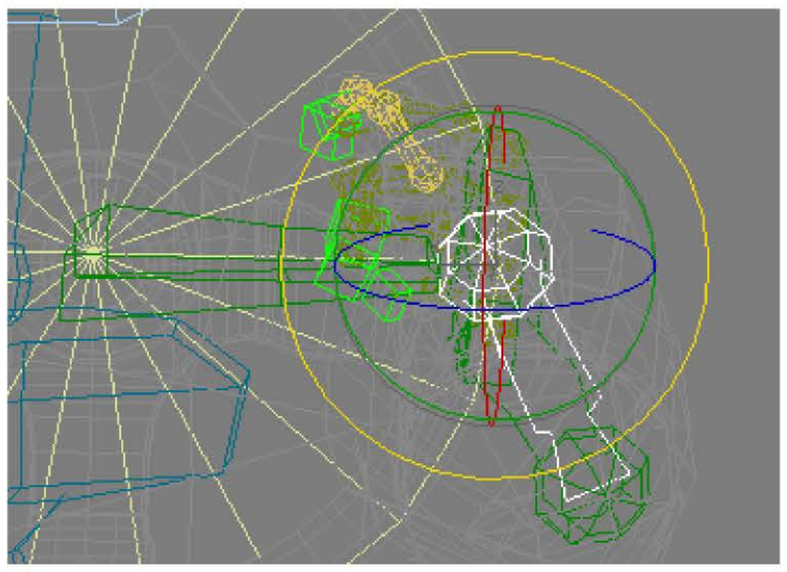
We will build the right hand anim, from neutral position, to right lock position, from frame 140 to 280.
- activate Auto Key mode (button just under the time-line)
- Select the Bip01 R Hand bone and place the Time Slider at frame 140
- adjust bone postion to line up top left with first guide (see screenshot -->)
- check that the hand is correctly handling sterring wheel (orthographic view)

For each keyframe, we'll have to rotate the arm a bit (hand have to stay lined up with steering wheel side), and also move the Bip01 R Hand bone

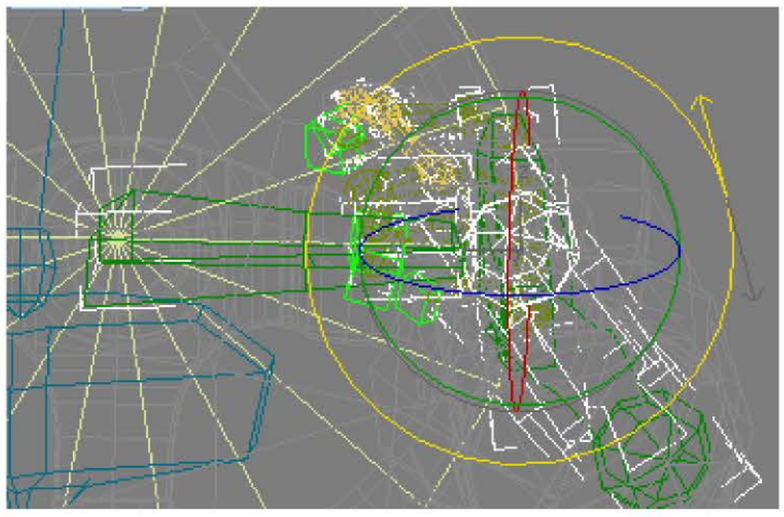
- move Time Slider to frame 150
- select Bip01 R UpperArm bone
- in orthographic view, line up the Bip01 R UpperArm bone with the hand



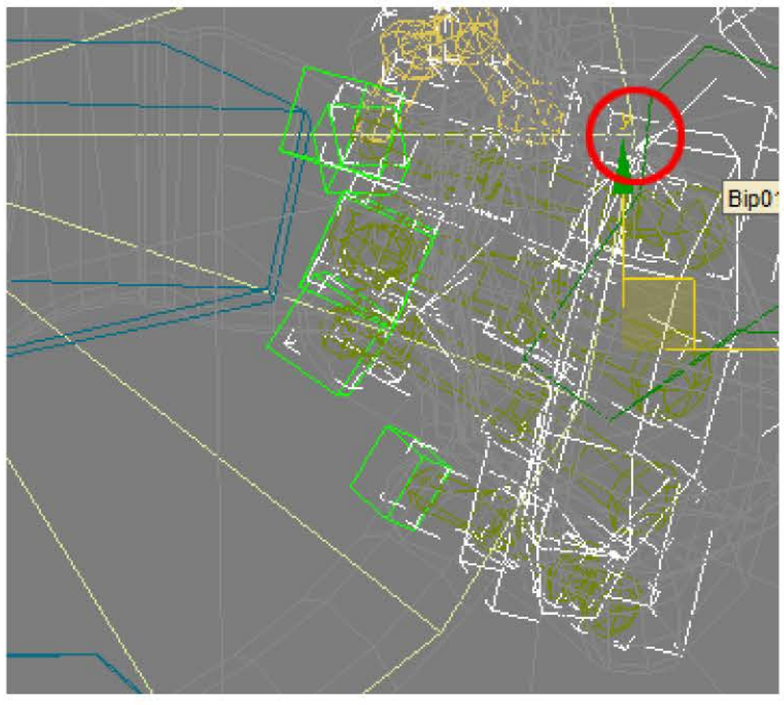
- select rotation tool
- use absolute rotation gizmo (outside circle)



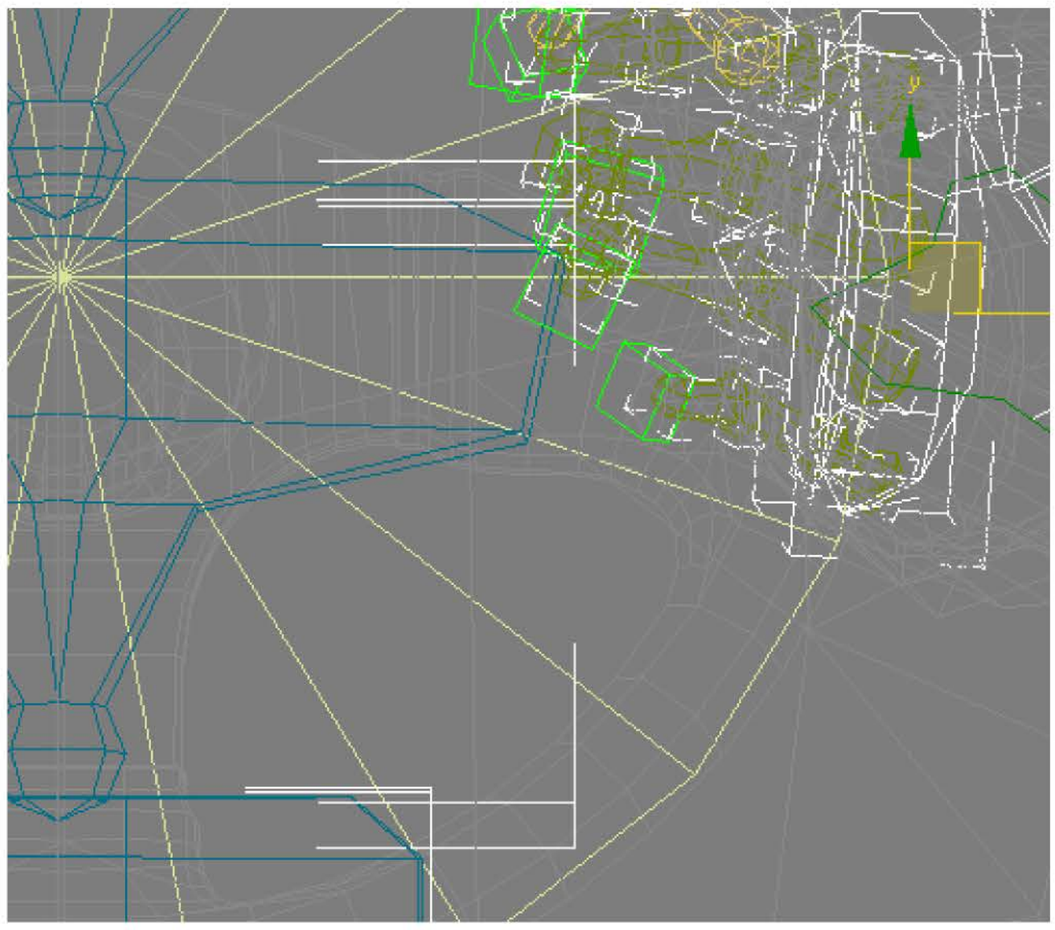
- rotate object from about -10° (check highest value in bottom angle numbers windows)



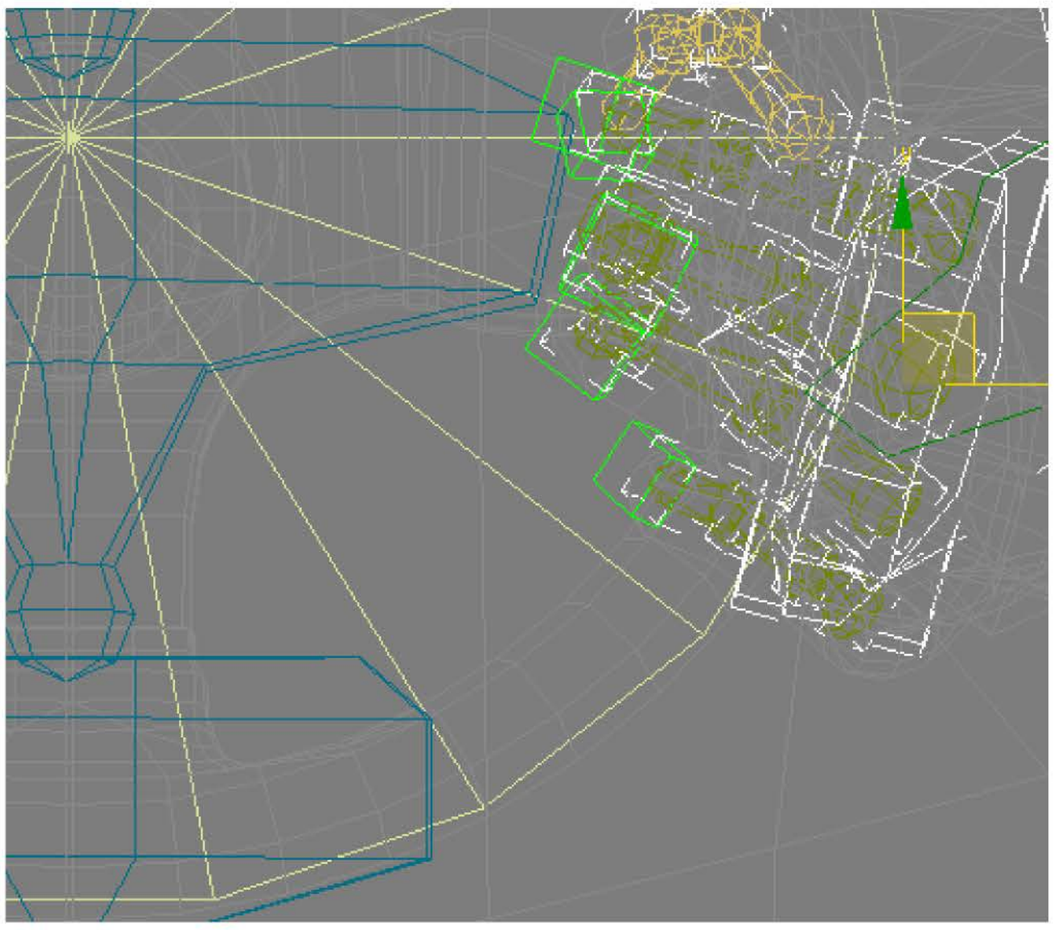
- in back view select the Bip01 R Hand bone
- line up top left corner with next line guide
- check in orthographic view if the hand is still well adjusted to steering wheel, it may need to be moved back a bit.



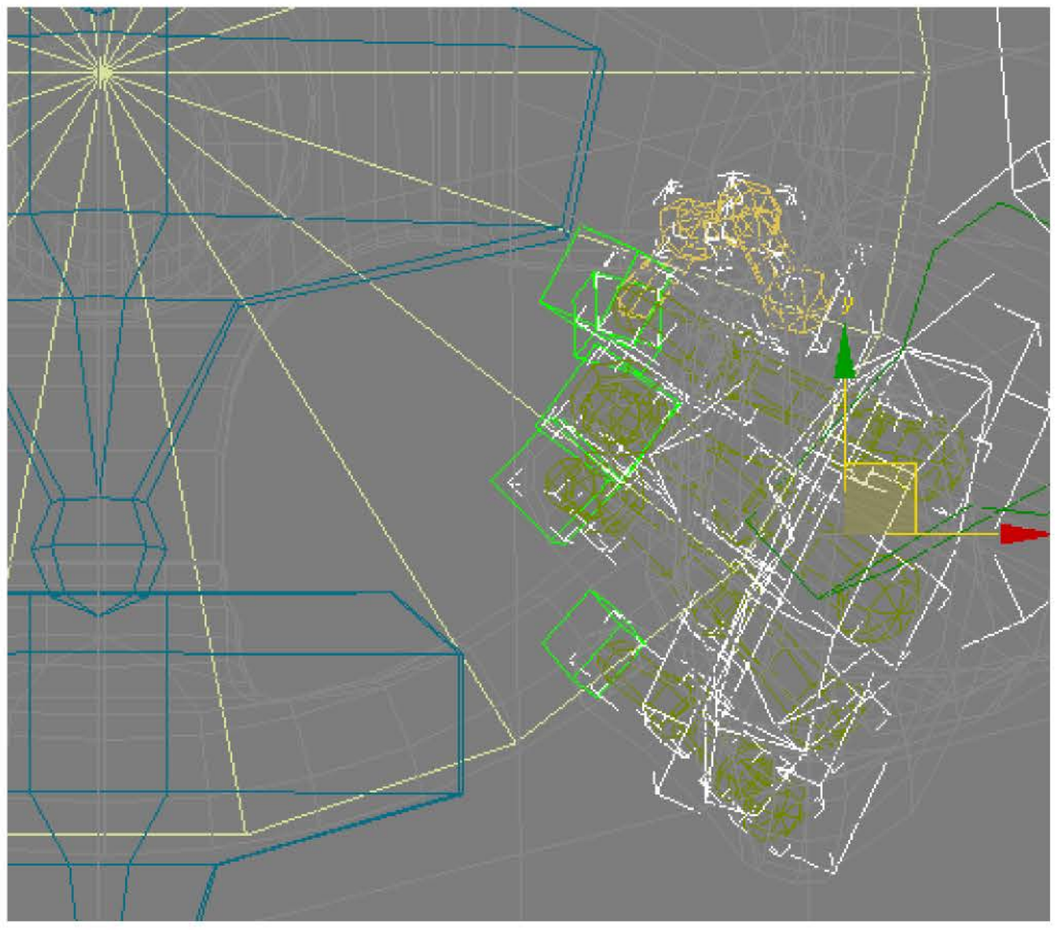
Repeat the same operation every 10 frames for each hand,
here is hand position from frame 140 to180



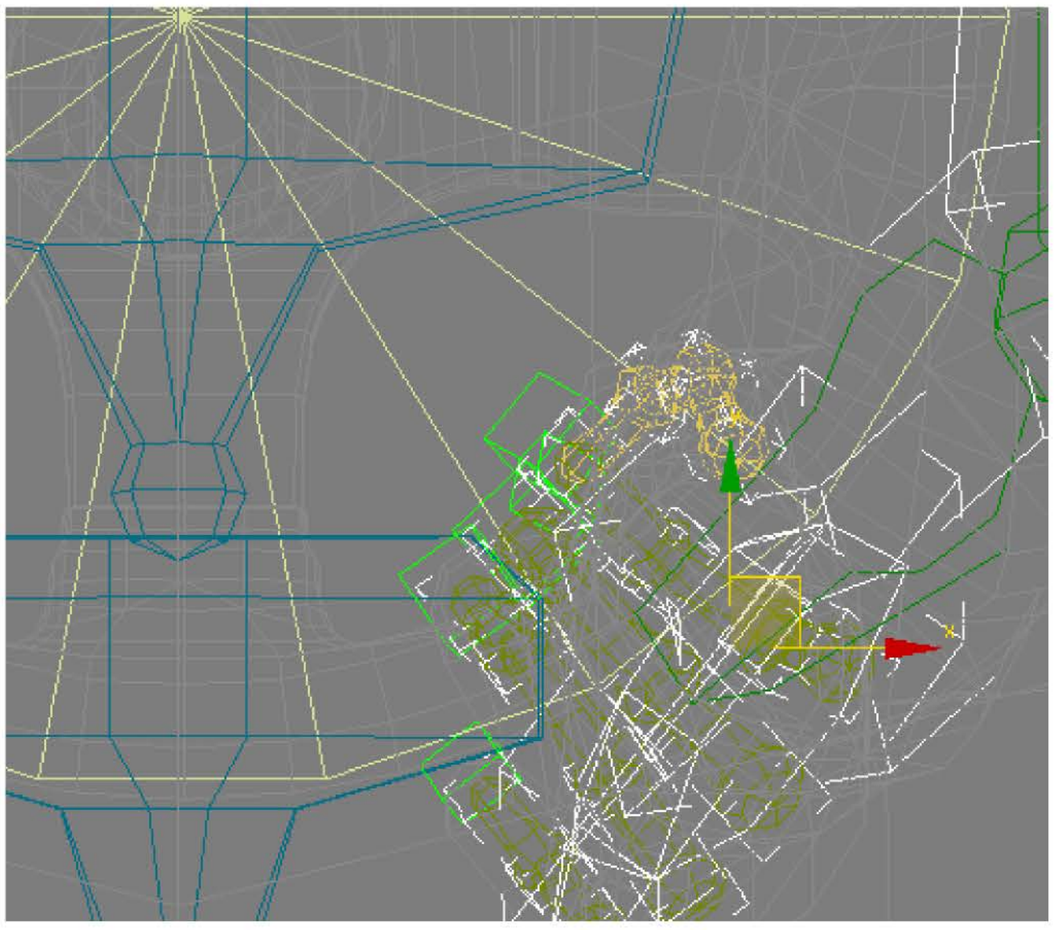
140



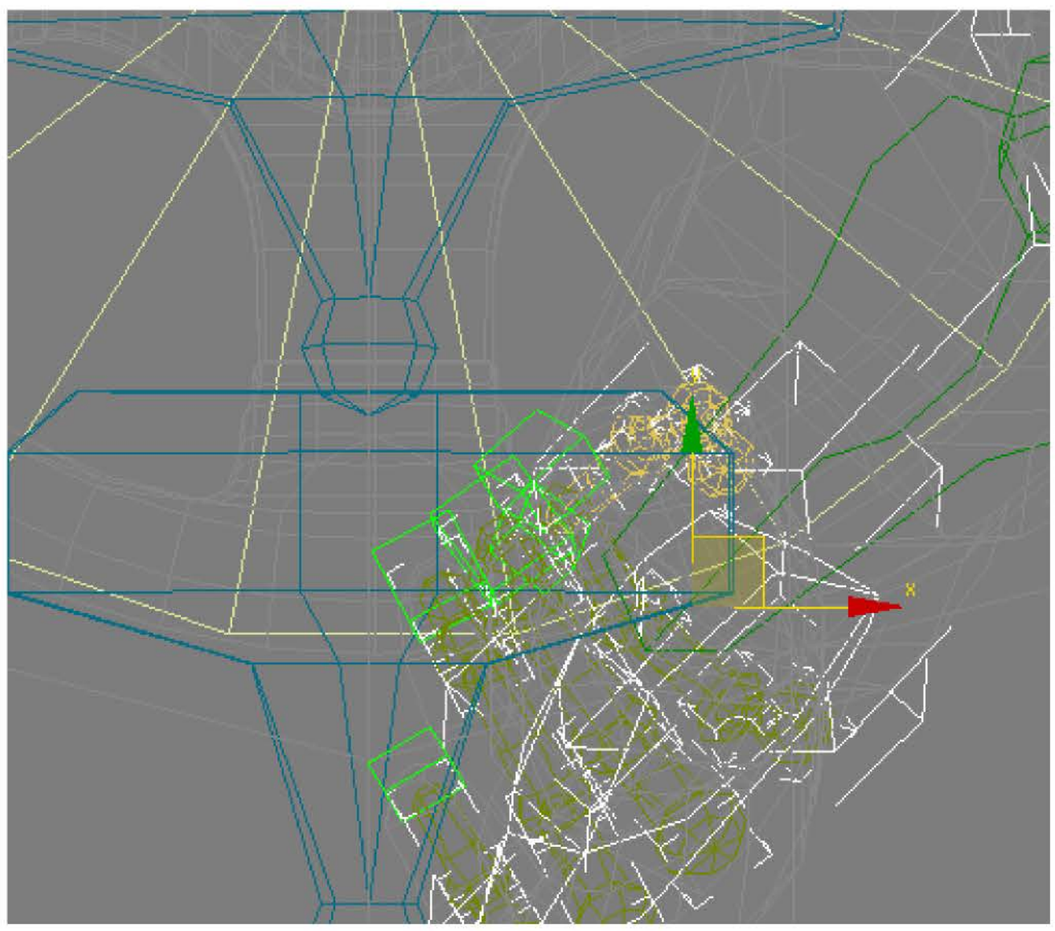
150



160



170



180