MXND MODEL

POWERED BY MAXIIONED

Farming Simulator 19

Static Lights

Von Maya über GiantsEditor bis zum Spiel

Vorwort

Zunächst möchte ich mich erstmal bei allen Beteiligten aus den Foren von Marhu.Net und der LS-ModCompany bedanken. Diese haben die ersten und auch für dieses Tutorial wichtigen Schritte gemacht. Über Tage hinweg probiert und somit auch verstanden, wie der neue VehicleShader.xml funktioniert. Nur durch diese zwei Bereiche ist es mir ermöglich worden, meine eigenen Leuchten in das aktuelle Farming Simulator 19 zu importieren.

Die meisten und auch wichtigsten Einstellungen müssen leider direkt im 3D-Programm getroffen werden.

Der Unterschied zwischen LS17 und LS19

Generell unterscheidet man zwei Arten von Beleuchtung. Die realLights und die Coronas/StaticLights.

Im LS17 wurde das optische Licht über sogenannte Coronas gesteuert. Dies war nicht mehr als eine einfache Plane welche mit einer Textur belegt war. Dazu benutze man zusätzlich den „emessiveLightShader“ welcher dann genau deklariert hat, ob es ein TurnLight oder StaticLight ist. [Bild1]

Im LS19 wird dies nun über einen neuen Shader Namens vehicleShader und sogenannte VertexAttributes gelöst. Vertexattribute sind zb „COLOR“. Der Vorteil hierbei ist, dass die eigentlichen Vertices (Flächen) selbst leuchten. Somit fallen unnötige Texturen komplett weg. Ebenso sind die Einstellungen in dem GiantsEditor deutlich magerer ausgeführt als wie bei dem Vorgänger. Hier weise ich nur noch den Shader zu und setzte ein Attribut auf StaticLight.

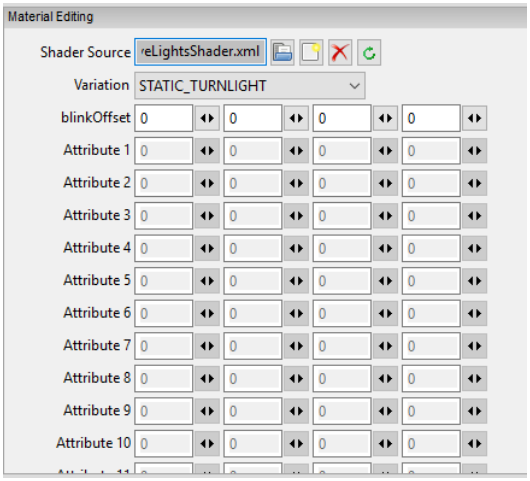


Abbildung 2 - LS17 – GiantsEditor 7.1.0

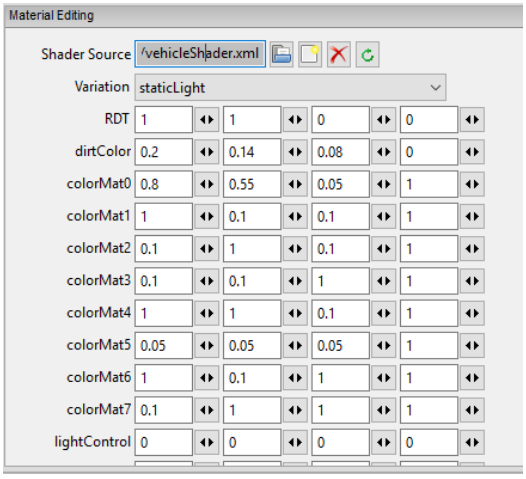
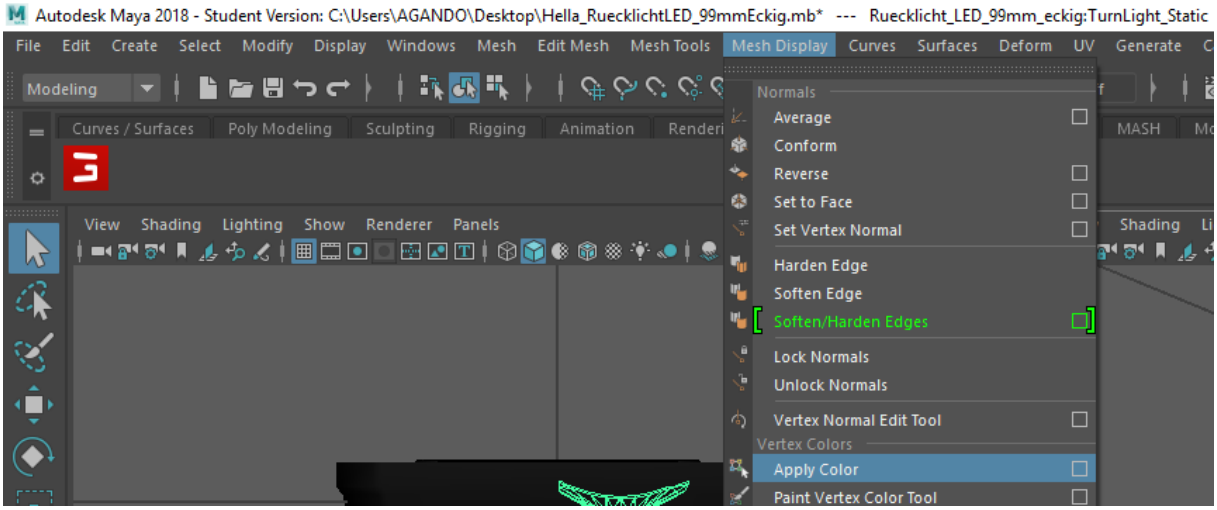


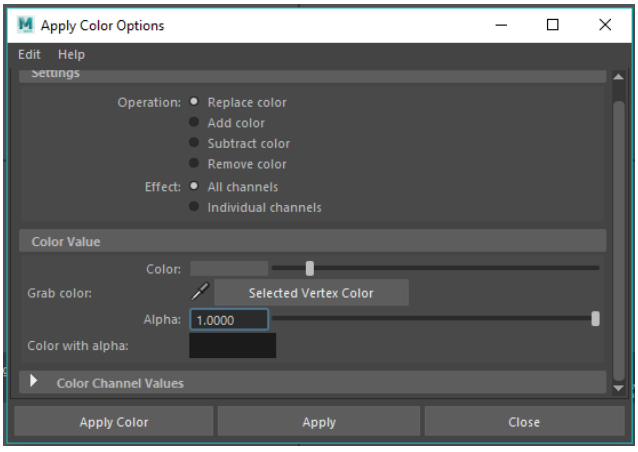
Abbildung 1 – LS19 – GiantsEditor 8.1.0

Maya

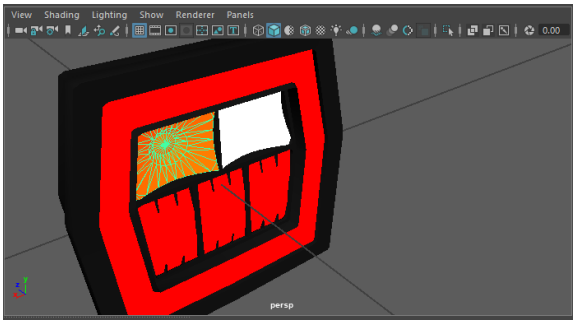
In Maya muss ich natürlich gewisse Grundeinstellungen Tätigen. Auch ist hier die Position der UV im Grid entscheidend. Das Grid befindet sich direkt im UV-Editor und kann entsprechend verschoben werden. Dies ist für uns in so weit wichtig, da darüber die Funktion der Leuchte gesteuert wird. Aber zunächst einmal zu den Attributen, die uns überhaupt die Möglichkeit geben einen Teil als Licht, Decal oder ähnlichem zu deklarieren. Als erstes wählen wir unser Mesh aus, welches wir zu einem Licht oderähnlichen machen wollen. Hierzu geht ihr in Maya auf folgenden Reiter.



Hier klicken wir auf das kleine Kästchen hinter Apply Color. Danach sollte sich folgendes Fenster öffnen.



Hier können wir zunächst die Farbe mit RGB Werten Festlegen. Klickt auf Color, wählt als Farbe z.B. Orange und unser gewähltes Mesh sollte dann auch die entsprechende Farbe angenommen haben. Das können wir ja auch ganz einfach im View überprüfen.

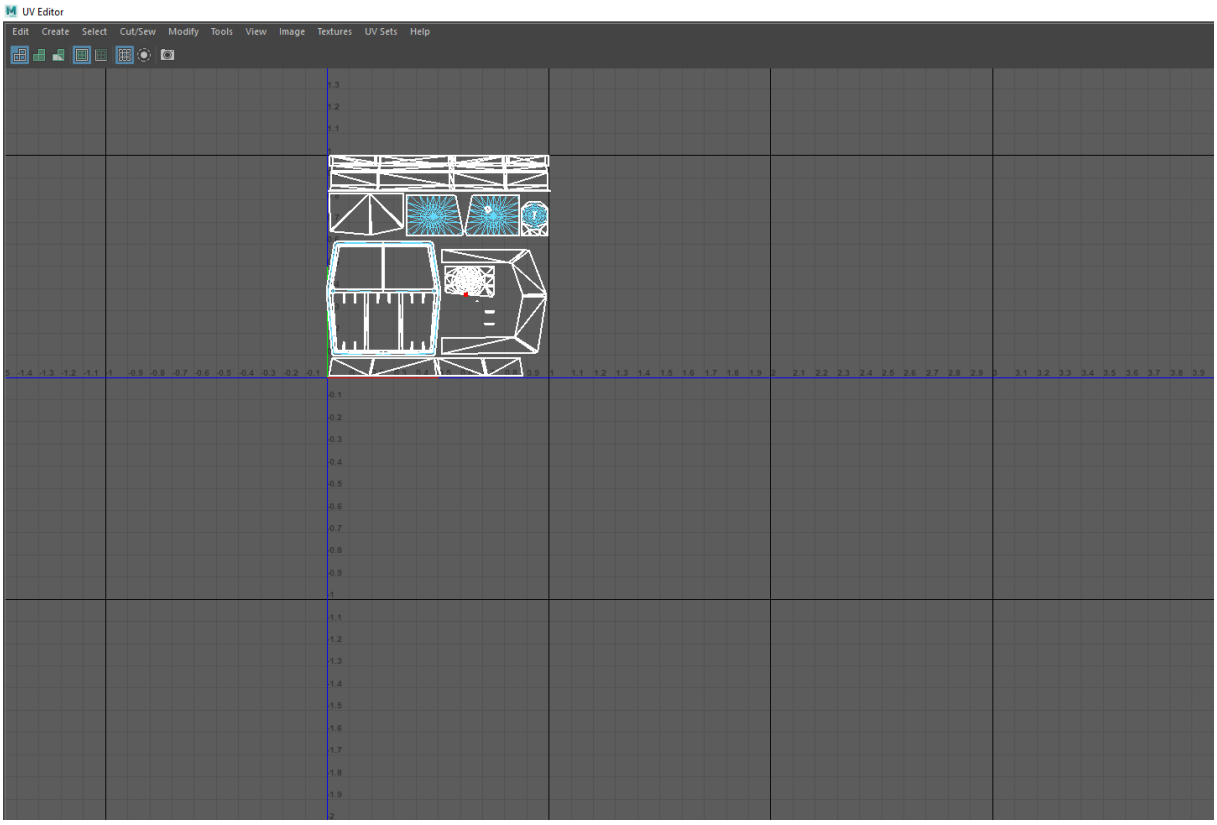


Okay, also alles richtig gemacht.

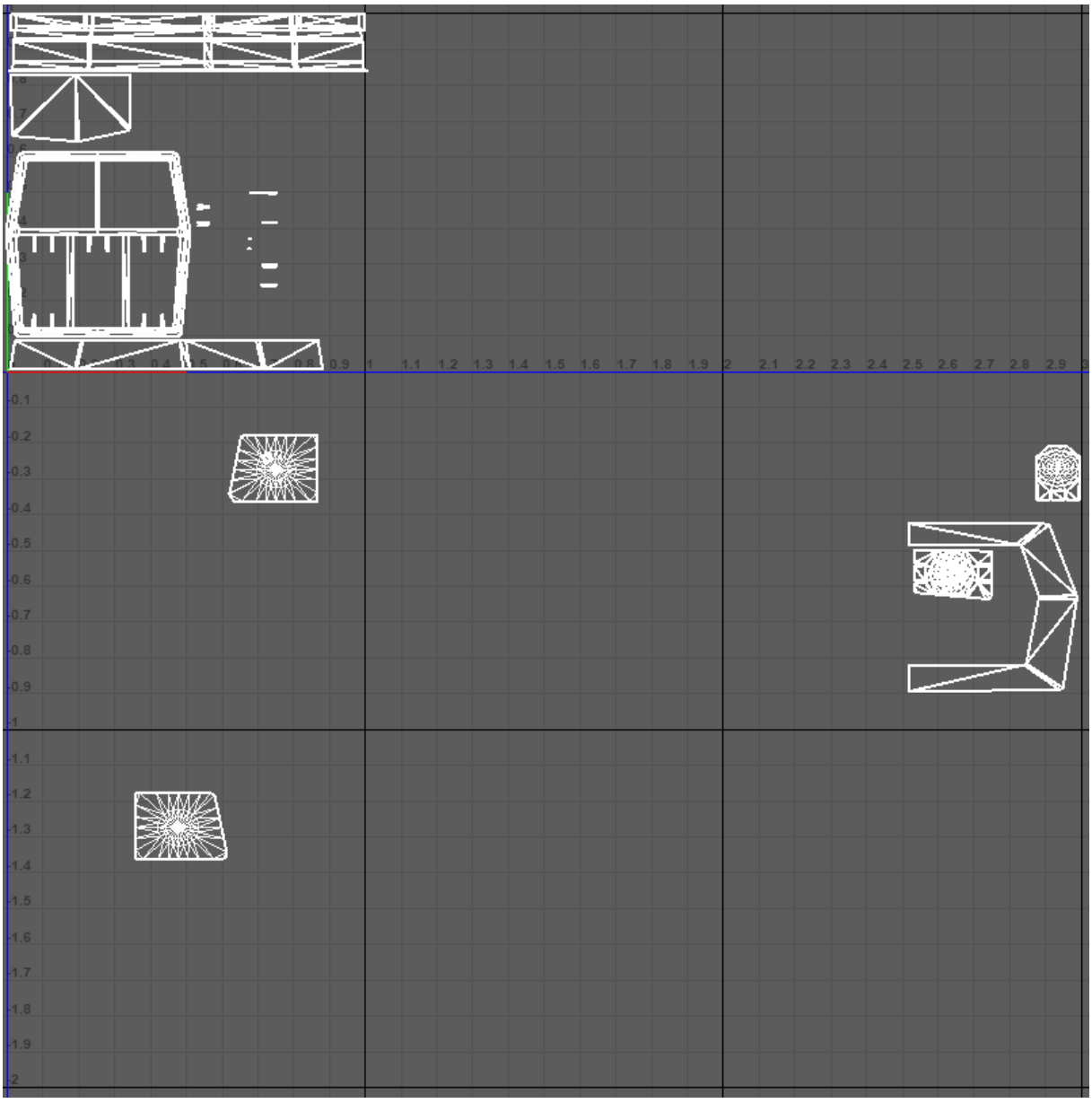
Natürlich empfiehlt es sich dies für alle einzelnen Meshes durchzuführen (wie im oberen Bild). Ist das ganze geschafft, so kommen wir zum etwas komplizierteren Teil.

Die neuen UV

Der vehicleShader weiß nur über den definierten Bereich der UV, was er am Ende damit machen soll. Wir müssen also Außerhalb unseres Kastens denken.



So in etwa sieht eine herkömmliche UV aus. Der genutzte Bereich liegt bei 0-1. Alles was darüber hinaus geht bewirkt normalerweise eine Wiederholung der Textur, aber ist spieltechnisch nicht entscheidend. Der LS17 kam damit z.B. meistens gar nicht klar und hat in der Regel die Texturen zerpflückt. Im LS19 brauchen wir diesen Bereich nun für die Shader um damit die nötigen Optionen festlegen zu können. Also nehmen wir uns dieser Aufgabe an und kommen zu folgendem Ergebnis.



Das sieht natürlich erst einmal verwirrend aus aber ist prinzipiell ganz einfach. Wir benutzen hier insgesamt 4 UV-Bereiche. 0, -0, -0.2 und -1.

Nun stellt man sich die Frage warum man so viele Bereiche brauch. Auch das ist ganz einfach erklärt. Jedem Bereich wird eine Funktion zugeordnet.

0

Hier kommen einfache Bauteile hinein wie zb die Scheibe oder der Body

-0

Dieser Bereich dient scheinbar für Rückfahrscheinwerfer also alles was Weiß leuchten soll. Es funktioniert aber auch im Bereich 0

-0.2

Hier ist es vom Spiel her wohl für Rote Leuchten vorgesehen. Es funktioniert aber auch im Bereich 0

-1

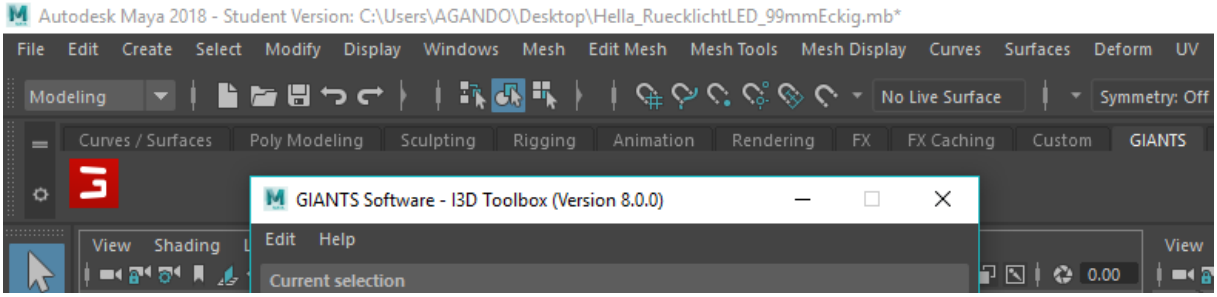
Und hier ist der Knackpunkt. Der Bereich -1 legt fest, dass dieses UV blinkt!

Diese Bereiche sind also für uns sehr wichtig, um die einzelnen Funktionen der Leuchte definieren zu können. Machen wir dieses nicht, legen unseren Blinker z.B. auf den Bereich 0 oder -0 so wird er zwar aktiviert aber leuchtet nur dauerhaft.

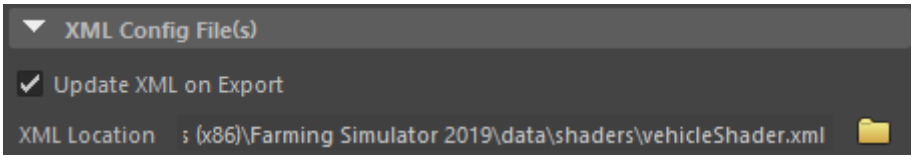
Nun gut kommen wir zu den nächsten Schritten, um unsere Leuchte in das Spiel zu bekommen. Das wäre dann der Exporter.

Exporter Modul

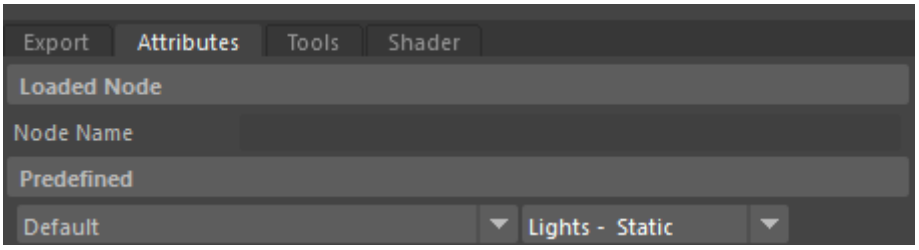
Zunächst müssen wir dieses erst einmal öffnen. Das geht wie gewohnt alles ganz einfach.



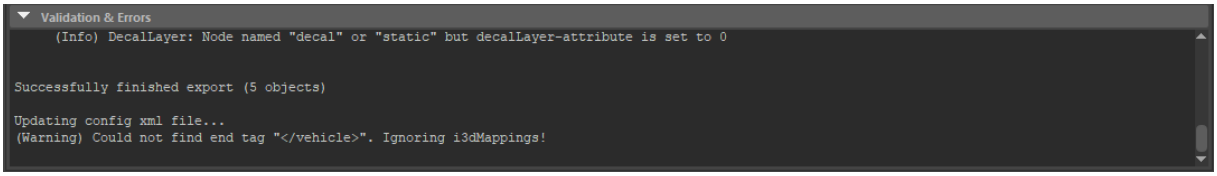
Ist dieses geschafft, so eröffnen sich uns viele Funktionen und Einstellungen. Die für uns wichtigen sind dabei.



und



Haben wir diese Einstellungen getätigt, geben wir unserem Objekt noch einen Namen und klicken auf exportieren.



Dieser Fehler kommt dabei öfter einmal vor, kann aber aus unserer Sich ignoriert werden. Er hat keine nachteilige Wirkung im Spiel.

Die neue i3D

Hier gibt es einige Neuerungen mit zugewiesenen Materialien bzw Attributen.

```
<Material name="MXND_Corona_Weiss" materialId="7" ambientColor="1 1 1" alphaBlending="true" customShaderId="6" customShaderVariation="STATIC_LIGHT">
  <Emissivemap fileId="5"/>
  <CustomParameter name="blinkOffset" value="0 0 0 0"/>
</Material>
```

Abbildung 3 - LS17 - Beispiel i3D geöffnet mit Notepad++

```
<Material name="Ruecklicht_LED_99mm_eckig:ReverseLight_Static2" materialId="283" customShaderId="2" customShaderVariation="staticLight">
  <Texture fileId="8"/>
  <CustomParameter name="RDT" value="1 1 0 0"/>
  <CustomParameter name="dirtColor" value="0.2 0.14 0.08 0"/>
  <CustomParameter name="colorMat0" value="0.8 0.55 0.05 1"/>
  <CustomParameter name="colorMat1" value="1 0.1 0.1 1"/>
  <CustomParameter name="colorMat2" value="0.1 1 0.1 1"/>
  <CustomParameter name="colorMat3" value="0.1 0.1 1 1"/>
  <CustomParameter name="colorMat4" value="1 1 0.1 1"/>
  <CustomParameter name="colorMat5" value="0.05 0.05 0.05 1"/>
  <CustomParameter name="colorMat6" value="1 0.1 1 1"/>
  <CustomParameter name="colorMat7" value="0.1 1 1 1"/>
  <CustomParameter name="lightControl" value="0 0 0 0"/>
  <CustomParameter name="blinkOffset" value="0 0 1 0"/>
</Material>
```

Abbildung 4 - LS19 - Beispiel geöffnet mit Notepadd++

Ihr seht also, es hat sich viel geändert. Ich gehe hier aber nur auf das nötigste ein.

Der Giants Editor 8 und neuer

Nun überprüfen wir noch fix ob unsere Einstellungen übernommen wurden. Wenn dies nicht passiert ist, klicken wir auf unser Material/Objekt und weisen ihm im MaterialEditing Fenster den vehicleShader zu. Noch fix das Attribute auf staticLight gesetzt und fertig.

Die neuen Einstellungen in der XML

Schauen wir uns zunächst einmal eine XML eines Fahrzeuges an. Aus dem 17er kennen wir ja den Tag <Lights> welcher alle Lichter enthält.

```
<lights>
    <sharedLight linkNode="backlightLeft" filename="MXND_Hella_99mmEckigRearLightLED_left.xml"/>
    <sharedLight linkNode="backlightRight" filename="MXND_Hella_99mmEckigRearLightLED_right.xml"/>
    <sharedLight linkNode="triangle01" filename="$data/shared/assets/reflectors/lizard/redTriangle_01.xml"/>
    <sharedLight linkNode="triangle02" filename="$data/shared/assets/reflectors/lizard/redTriangle_01.xml"/>
    <sharedLight linkNode="reflector01" filename="$data/shared/assets/reflectors/lizard/yellowRectangle_01.xml"/>
    <sharedLight linkNode="reflector02" filename="$data/shared/assets/reflectors/lizard/yellowRectangle_01.xml"/>
    <sharedLight linkNode="reflector03" filename="$data/shared/assets/reflectors/lizard/yellowRectangle_01.xml"/>
    <sharedLight linkNode="reflector04" filename="$data/shared/assets/reflectors/lizard/yellowRectangle_01.xml"/>
    <sharedLight linkNode="reflector05" filename="$data/shared/assets/reflectors/lizard/yellowRectangle_01.xml"/>
    <sharedLight linkNode="reflector06" filename="$data/shared/assets/reflectors/lizard/yellowRectangle_01.xml"/>
    <realLights>
        <high>
            <light node="backLightsHigh1" lightTypes="0"/>
            <turnLightLeft node="turnLightLeftBack"/>
            <turnLightRight node="turnLightRightBack"/>
            <brakeLight node="backLightsHigh1"/>
        </high>
    </realLights>
</lights>
```

Gähnende Leere, aber das realLight kommt uns ja bekannt vor.

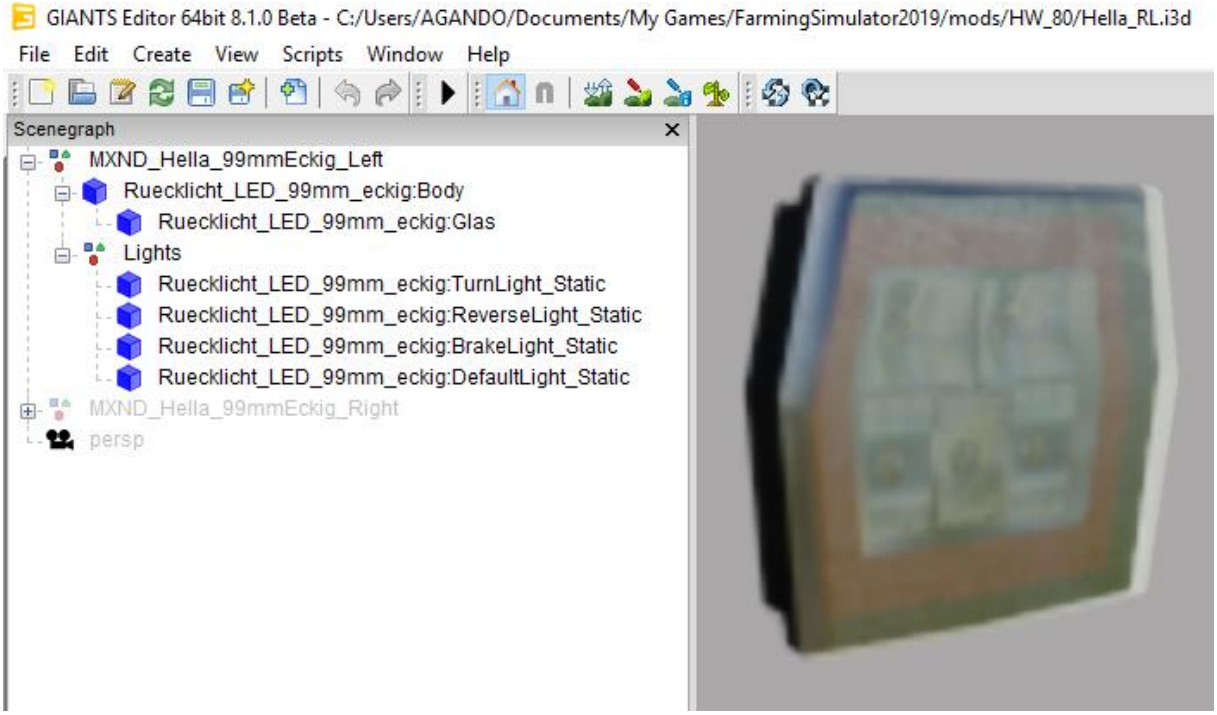
Die große Neuerung nennt sich sharedLights welche Leuchteinheiten aus einer Datenbank laden. Es mag für den einen gut, für den anderen schlecht sein. Für uns ist es einfacher. Denn wir können über nur einen kurzen Eintrag eine komplette Leuchte mit allen Funktionen laden. Der Mod wird also kleiner und die Ladezeit verkürzt sich.

Nun gut, wie ist so eine neue Leuchte überhaupt aufgebaut. Sie besteht aus:

- I3D Datei
- LeuchteXYZLinks.xml
- LeuchteXYZRechts.xml
- Texturen

Auch das spart uns wieder sehr viel Aufwand. Denn gemoddete Einzelleuchten kann ich so ganz einfach in meinen Mod ziehen und muss nur noch die zwei XML Dateien verlinken.

Also schauen wir uns mal direkt so eine i3D an.



Die ist auch ziemlich Leer oder? Erstaunlicher Weise besitzt das ganze Objekt sogar nur 2 Materialien. Ebenso erstaunlich, die reichen auch!

Material 1 – Body + alle staticLights

Material 2 – Glas

Das schöne hier ist wirklich, dass ich das ganze in der XML deklariere. Diese sieht dann wie folgt aus.

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<light>
    <filename>Hella_RL.i3d</filename>
    <rootNode node="0" />
    <defaultLight node="0|1|3" lightTypes="0" />
    <brakeLight node="0|1|2" />
    <turnLightLeft node="0|1|0" />
    <reverseLight node="0|1|1" />
</light>
```

Und da war es auch schon. Ich gebe wie auch im 17er die IndexPath an, wichtig ist hier die rootNode, da diese das ganze Objekt festlegt. Also Pfade übertragen, speichern und schon funktioniert alles.

euer OcciRS