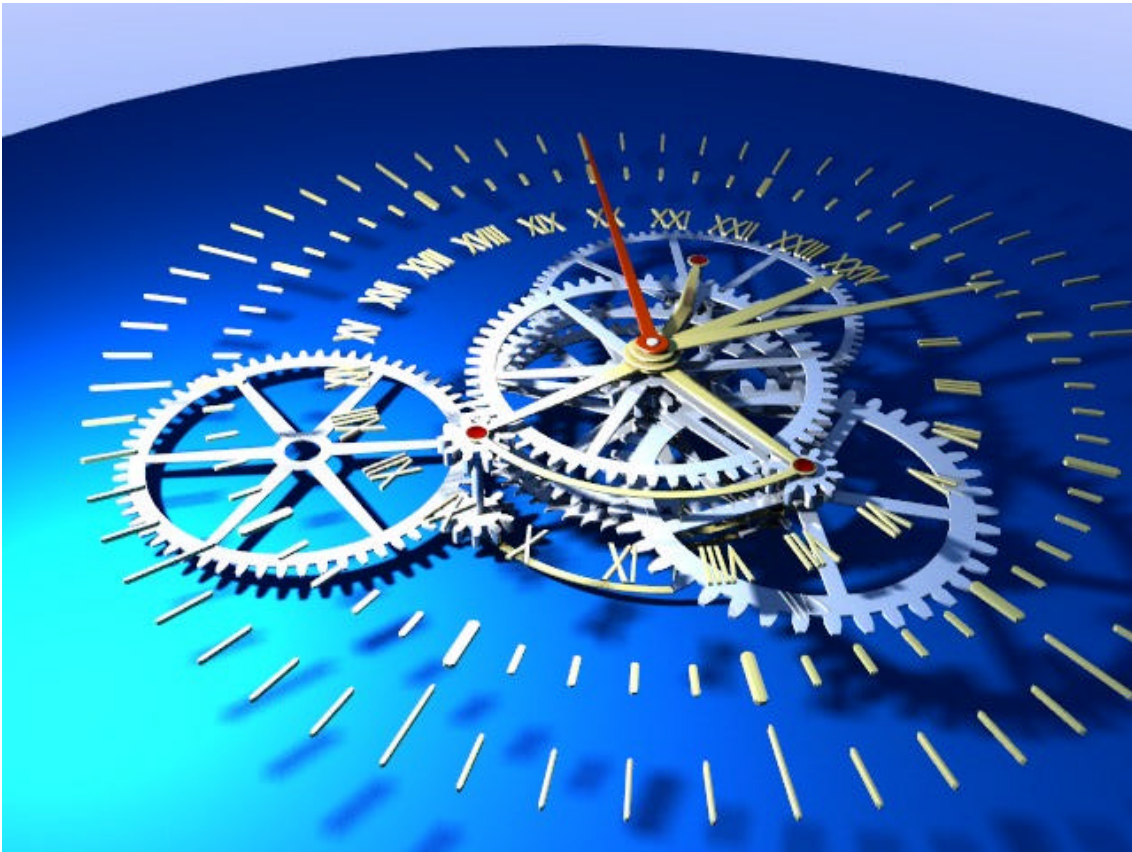


# Blender Mechanical Gears



© Nov 2004 – Stefano Selleri a.k.a. S68  
**Vers. 0.0.2**  
**‘Myrica’**

# Table of Contents

---

<b>INTRODUCTION</b>	<b>3</b>
<b>PACKAGE CONTENT</b>	<b>3</b>
<b>DEPENDENCIES</b>	<b>3</b>
<b>INSTALLATION</b>	<b>3</b>
<b>USAGE</b>	<b>4</b>
<b>LOAD AND SAVE</b>	<b>4</b>
<b>GEAR INTRODUCTION</b>	<b>4</b>
GEAR BASICS	4
GEAR MESHING	5
FINISHING TOUCHES	6
THE PINION(S)	6
<b>GEAR TYPES</b>	<b>6</b>
CYLINDER	6
RACKS & CROWNS	7
CONICAL	7
<b>PLACE THEM!</b>	<b>8</b>
<b>SPIN THEM!</b>	<b>8</b>
<b>TO DO</b>	<b>10</b>

## Disclaimer

---

This software is provided 'as it is' no guarantees expressed or implied are given, except that it will occupy some space on your disk(s). The software, bmg.blend and this document are © 2004 Stefano <S68> Selleri and released under Blender Artistic license which you should have received with the package. If this is not the case a copy of said license is available at [www.blender.org](http://www.blender.org) or directly from the author [selleri@det.unifi.it](mailto:selleri@det.unifi.it).

# Introduction

---

**Blender Mechanical Gears (BMG)** is a suite of (only two) Python scripts designed to build correct gears (involute profile) and make them turn. This script is inspired by old (1999) **grinder** script by J. Merritt

## Package Content

---

The **BMG** package should contain:

1. **BMG.pdf** – This file you are reading ✍
2. **BMG.blend** – An example Blender file with the script within
3. **BMGm-0.0.2.py** – The *mesher* script, as a separate text file.
4. **BMGs-0.0.2.py** – The *spinner* script, as a separate text file.
5. **Licence.txt** – The Blender Artistic License under which this package is given.

## Dependencies

---

**BMG** needs the following softwares to be correctly installed on your computer to work:

1. An operating system of your choice ✍.
2. Blender 2.34 and maybe later version ( © The Blender Fund - [www.blender.org](http://www.blender.org) ).
3. Python 2.3.4 or whatever Python is recommended with your version of Blender ( © - [www.python.org](http://www.python.org) )

## Installation

---

Unpack the package in a directory of your choice.

You might want to copy the two Python (\*.py) files in your **.blender/scripts** directory. This is the preferred way of handling script in Blender now!

Once you are done you can run the **BMGm** script in any blender file or load the



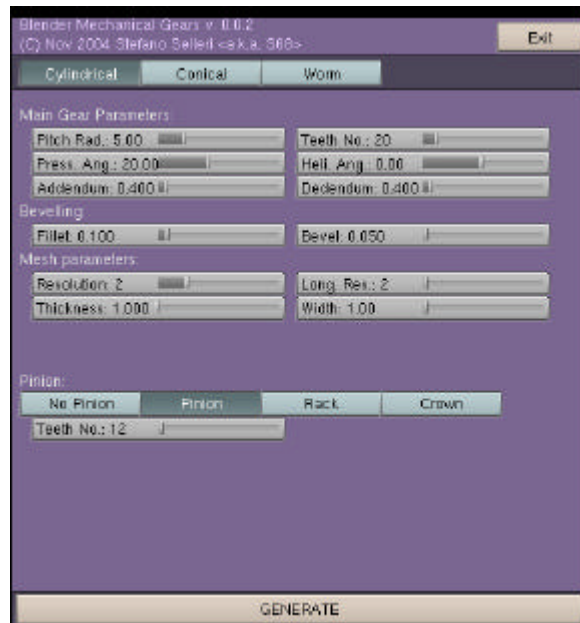
**BMG.blend** file, place your mouse cursor on the left Blender window, the text window containing the **BMGm** script and press **ALT-P**.

## Usage

Once you have launched **BMGm** the Graphical User Interface (**GUI**) here on the right should appear. If not you have a problem.

The **GUI** is divided into three part:

1. A top part with credits and three buttons: **Load**, **Save** and **Exit** – The three buttons does exactly what you are thinking of (Only the **Exit** button is there in 0.0.2).
2. A lower-top part letting you decide the kind of gear(s).
3. A central, big, part, which lets you define the gear(s).
4. A lower part with just one button: **GENERATE** to launch execution.



## Load and Save

**BMG** will allow you to load and save any gear you have defined. Gears are stored in a XML file defined in a chapter of its own later on. This will be there in 0.0.4... maybe...

## Gear Introduction

**BMG** can produce 2 types of gear (up to now – worm gears wil come in 0.0.4) all share some data:

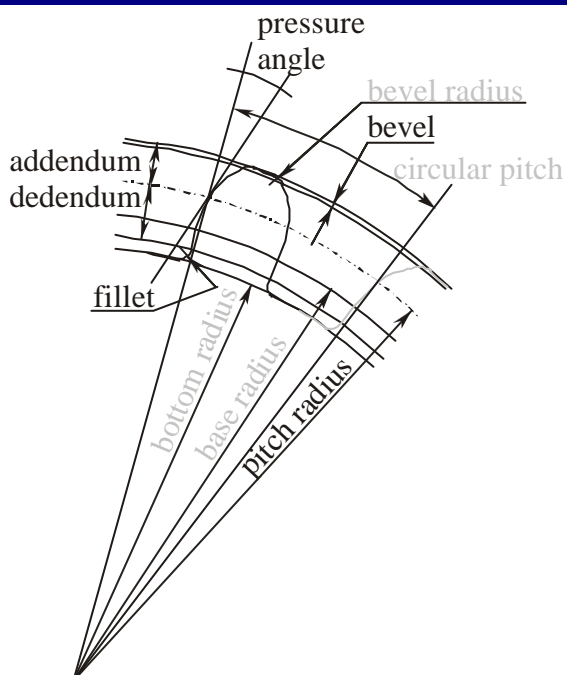
### Gear basics

First some dry theory! Look left. A Correct, working mechanical gear is done like that (User defined parameters in black, internally computed parameters grey):

The really basic data are:

**Number of teeth:** how many teeth are there in the gear

**Pitch radius:** the ideal circle defining the gear. Two (correctly) meshing gears mus have their respective pitch circle mutually *tangent*.



The data you might want to play with more often are:

**Addendum:** how much the tooth protudes beyond the pitch radius

**Dedendum:** how deep goes the tooth before any fillet take place

**Pressure angle:** The angle between the radial direction and the tangent to the tooth at the pitch circle. The default value of 20 is a good choiche.

Based on pressure angle is the value of the **base circle**. There are a few rules here:

- ✂ The smaller the pressure angle, the smaller the base radius
- ✂ Tha addendum must be less than the dedendum
- ✂ The dedendum must be more than the difference between the pitch and the base radius.

Experiment!

As a minor cosmetic improvement you can define:

**Fillet radius:** defining the bottom of tooth bevel

**Bevel:** defining top of the tooth bevel and all other bevels (see later)

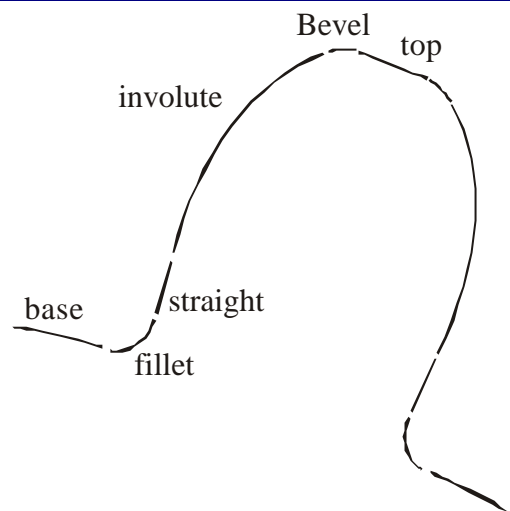
## Gear meshing

The gear is then created as a single tooth mesh (you have to SpinDup it, see later on) This is created by first defining a profile, the black thick line in the figure here on right.

Such a line has a base circle, a fillet circle, a straight part, an involute profile, a bevel circle, a top circle and so on back to the base circle

This profile is created with a number of nodes proportional to the **resolution** slider.

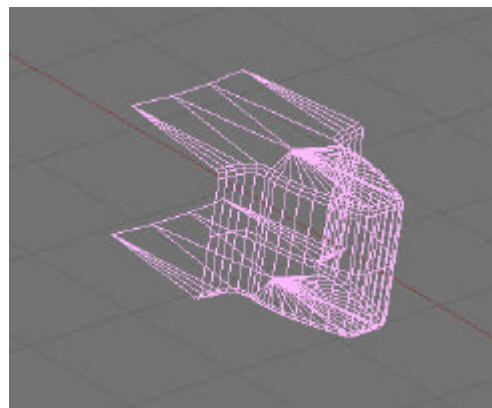
Base circle	= 2 x resolution
Fillet	= resolution
Straight	= resolution
Involute	= 3 x resolution
Bevel	= resolution
Top	= 2 x resolution



So, be warned that a resolution 2 gear can already have a lot of verts!

This profile is then taken and extruded **Long. Res.** Times so as to make the tooth **Thickness** blender Units thick. Top and bottom of tooth are beveled at **Bevel** radius and closed with some faces. The base circe is extruded radially **Width** blender units to give a better thickness.

The resulting tooth is shown here on the side. This is done with all default values.



## Finishing touches

---

Please note that a single tooth is generated, and the mesh is left open towards the center of the tooth. This is intentional.

From top view you can go mesh editing mode, select all vertices, and do a Spin Dup over 360° for a number of repetitions equal to the teeth number in the gear. Remember to remove doubles!

The inner border has regularly (angular) spaced vertices. It is very easy to do whatever bulk you desire for your wheels, solid, with holes, etc.

## The Pinion(s)

---

A Gear alone is seldom useful. You need meshing gears, two gears able to interconnect and revolve. You can create each gear one by one, but meshing gear needs to share few parameters, so it is much easier to generate gears in pairs. The second gear, usually smaller, is defined a *Pinion*.

Below the Gear parameters there is an area of the GUI devoted to whatever mechanism you want meshing with the main gear. You can select three devices... so there are four toggles.

If **No Pinion** is selected... then no pinion is generated!

If the **Pinion** toggle is on (the default) then there is an additional slider, asking for the number of teeth in the pinion. The pinion is then correctly generated next to the gear. Usually the pinion has less teeth than the gear, but this is not strict and the script can handle a larger number of teeth in the pinion. Strictly speaking then the gear is the pinion and the pinion is the gear, well, who cares.

The need to create both gears at once is very strong in conical gears, were the math is more complex.

If the gear is Cylindrical then two more options are there: **Rack** and **Crown**. These are impossible on Conical and Worm gears due to their nature.

A **Rack** is a degenerate pinion of infinite Pitch Radius (and hence infinite number of teeth) The involute side degenerates into a straight line.

A **Crown** gear ideally comes when the pinion pitch radius becomes negative. It is a circle with teeth *on the inside*. In this case it is mandatory that the 'pinion' (actually the crown) has more teeth than the gear, or the gear won't fit inside the crown!

## Gear Types

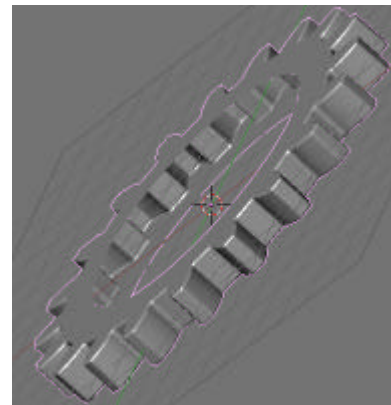
---

As said above we have 3 basic types of gears:

### Cylinder

---

Basic gears, the gears 80% of the people think to when talking of gears are cylinder gears with **Heli. Ang.** Set to 0 (the default in BMG). They are great for low-tech, noisy, big machines or, at the other end of the scale, for small precision clockwork.

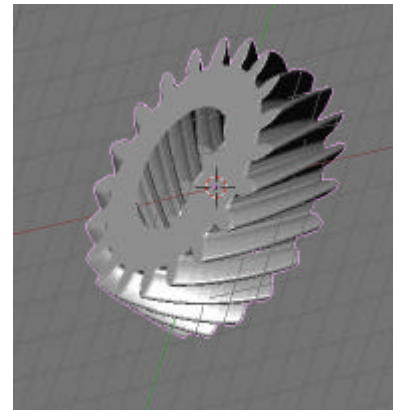




If **Heli. Ang.** Is nonzero then the longitudinal extrusion of the profile is done rotating along the gear axis. The tooth winds, producing an helix. These are more high tech, less noisy, less vibration, at the cost of some extra solicitation on the axis, which can be resolved by doubling it, with inverted helices.

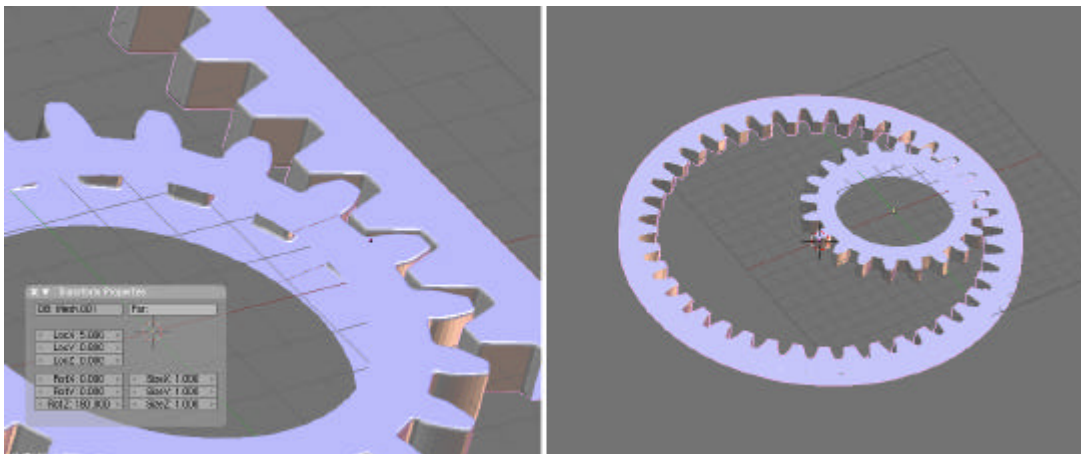
The wheel on the right has a 45° angle. More than 45° is insane.

The pinion needs to have an **Helical angle** of the *same* value and opposite sign (If you create both Gear and Pinion at the same time they are automatically created OK, this holds true also for Racks and Crowns.)



## Racks & Crowns

Racks and crowns are special, interesting kind of pinions (in BMG). Rack is herebelow on the left, Crown on the right.



A single tooth is always created. While for the Crown this is not a problem, you can SpinDup it, for the rack it is less easy, since you cannot Dup along a line.

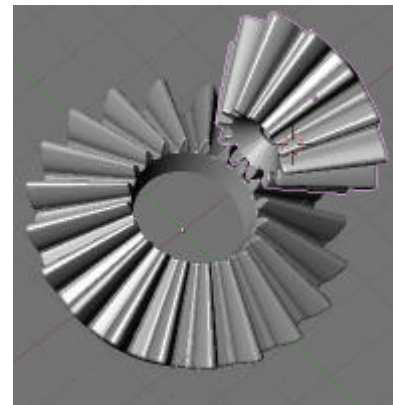
Solution is to create a segment with as many vertices as you want to have teeth, scale it so that vertices are at a distance equal to the distance the teeth must have (This number is kindly written by the script on the console!) And use Duplivot. It is a nice thing to make dups real then and join all teeths.

## Conical

Conical gears is what you need when the rotation axes of the two wheels are not parallel (but meet in a point).

The additional parameter here is **Axis Ang.**, which is the angle formed by the axes.

Please note that both the gear and the pinion are created with the axis parallel to the z global axis. *do not rotate the pinion*. Keep it as it is, or you will have problem in animating. Correct positioning of the gears will be discussed later on.



Spiral gears are just conical gears with spiraling tooth, much like the helical with respect to the spur. The **Heli. Ang.** Parameter handle this.

## Place Them!

Once you've created a gear, you need to place it in its correct position. Don't move it! Unless you don't need to animate it ✍

Instead, place the cursor on the Gear center (Snap to) and add an empty, possibly with z axis parallel to the gear rotation axis, then parent gear to empty.

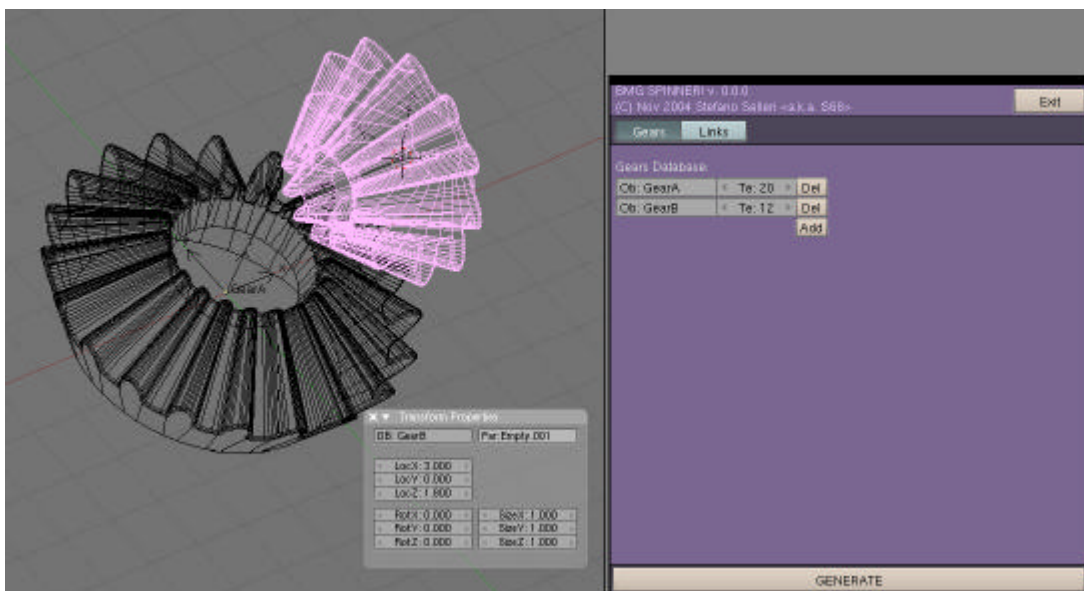
From now on, move and rotate only the empty! Also, give the gear objects some meaningful name!

This will allow you to use the second script!

## Spin Them!

Now that you have all the gears set up you might want to spin them, here the second script, **BMGs**, comes into play.

Assume we have the conical gears of the last example, the gear object name is GearA, the pinion object name is GearB.



Run Scripts -> Anim -> BMGs

The interface will show, in the central part, a simple **Add** button. Press it, a new line appears. Press **Add** again, a second line appears. Write in GearA and 20 in the first line, GearB and 12 in the second. These lines defines the name of the gear object and the relative number of teeth.



Why? Wasn't it possible to get these data at mesh creation? Yes and No, is it possible to get, but when you change name or delete an object there are problems, especially if the scrip is not running. I found it easier to have you write in this data...

Once you have entered all the gears, press to **Links** toggle.

This new panel let you define the gear connections. The **Add** button lets you add a connection (link).

In each link there is a driver, a driven and a link method.

The driver is the gear whose rotation is taken, the driven is the gear whose rotation is set, the link type says the formula used to compute the driven rotation.

There are three possible connections:

**Fixed:** The gears are linked to the same rotation axis, the turn together at the same angular speed.

**Mesh:** The gears are meshing, the angular speed of the driven depends on the angular speed of the driver and the ration between the teeth number. The Driven rotates in the *opposite* direction then the driver

**Inverse Mesh:** Same as above, but direction is the same, this is unphysical... it is necessary if you have a crown gear or if the z-axis of the gear do not match.

Indeed Crowns and racks have gears who do not only rotate, but also translate, so it is probably not that easy to handle... who knows, a next release (0.0.4?)...

Here GearA is the driver, GearB is the Driven, and the link is f **Mesh** type.

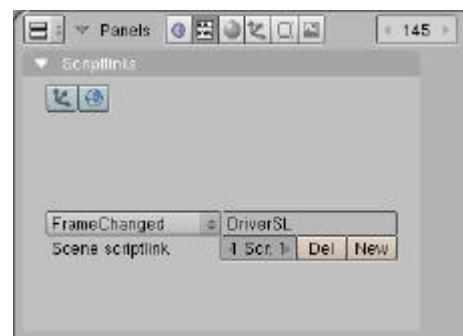
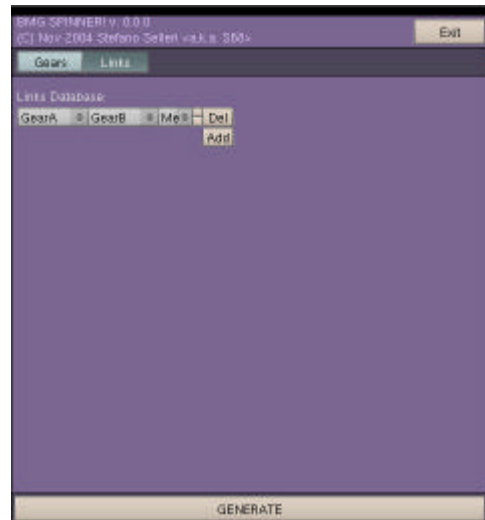
Once you are done, press the **GENERATE** button. A new script is generated, whose name is **DriverSL**. Put it as a scene scriptlink on FrameChanged. Beware! Blender is case sensitive!

Now we're almost done. Select the driver and add a RotZ IPO to it. Press ALT-A on the 3D window... the Gear will mothe and the other gear follows!

If not you probably messed up with empties.

Please note that you can have multiple wheels, not just two, you only need to define the links in the correct order,m since the DriverSL script will evaluate rotations in the order of definitions of the links. The two tiny buttons next to the link allows order change.

Note also that you must give the empties a starting rotation to have teeth meshing correctly.



# To Do

---

For Next release:

- ?? Worm Gears
- ?? Load & Save
- ?? DriverSL handling of Racks and Crowns