

**Kaunas University of Technology** Faculty of Informatics

**Project Report**

Component Based Software System Design T1210M16

**Adhithiyan Arul Mani**

Project authors

**dr. Šarūnas Packevičius** Supervisor

**Kaunas, 2025**

**Documentation :**

**1. Component Description:**

**a) Provided Functions:**

The Blender Add-on or the Component has the core functionalities for Gear generation:

* **Create\_gear\_mesh()**: This geometry creation function build a mesh of gear based on the user's input parameters like teeth, radius, depth, and bevel, and also the user can modify it dynamically
* **Update\_gear()**: This function is mainly for the dynamic changes or the modification of the gear based on the user's input and the property changes in the UI
* **GEAR\_OT\_add\_gear** : Adds a new gear to the layout window based on the user's given details
* **GEAR\_PT\_Panel** : which creates a UI panel in the 3D viewport on the sidebar which is used to control the parameters and initiates generation
* **GearSettings** : stores user-defined gear settings like the number of teeth, radius and other properties.

The Blender Add-on or the Component has the core functionalities for Wheel generation:

* **Update\_Wheel() :** when any property changes in the UI it triggers the wheel regeneration operator
* **GenerateWheelOperator :** This is the main function which builds the whole wheel with rim, tire, and spokes, then other parameters.
* **Create\_rim() :** which creates the rim geometry using a cylinder
* **Create\_tire() :** which creates the tire using a torus and scales it to simulate thickness
* **Create\_spokes() :** The evenly spaced spokes where created around the wheel by this function

# b) Purpose

This add-on Gear-Generator is mainly focused on the users comfort to quickly generate customizable 3D gear models directly in Blender with a live preview. It can be helpful for game development and other 3D modelling. The add-on Wheel-Generator is mainly to help designers and engineers create realistic-looking wheels with minimal no directly inside Blender. It supports customization of rim, tire, and spoke properties.

# c) Business Problems That the Component Solves

The Component addresses several real-world 3D development problems:

* **Auto Asset Creation and Reduce Manual Efforts:** It reduces the time spent on modelling mechanical structures like gears and Wheels by automating the task and reducing the manual approach
* **Enhance Design and Prototyping:** Allows for rapid iteration by dynamic updates by changing the parameters. giving the privilege of visually seeing the prototype and validation of multiple designs for artists and designers.
* **Democratize Technical and Mechanical Modelling:** Makes the asset creation process so easy and automates and more accessible to new users, even non-engineers and people who lack specialized modelling skills.
* **Promotes Reusability :** The Add-on generated meshes are editable and can be easily reused and used across different users

# d) Intended Use:

* Mainly for 3D modelers, designers, and students to create gear and wheel meshes
* For product visualization, automotive asset modelling, design concept, and 3D printing previews
* Artists to create game assets or pre-rendered vehicles
* It can be used for educational purposes for designing

# e) Restrictions for Component Usage (Technical and Business)

**Technical**

* **Tiers or Wheel:**
* Limited to a simple wheel with cylinder spokes, no complex design
* Cannot generate complex rim geometries with all the math calculations
* No image texture is used; only material is used for the build meshes
* No combined with physics systems for the rim and tier deformations
* **Gears:**
* Only flat gears are created, just a basic model
* Minimal control over fine gear tooth profiles, no geometry is used
* Not intended for high-end mechanical simulation

**Business:**

* It is only applicable and used inside the blender
* Add-ons have to be installed manually, or the class and function can be used by importing the Python library in the Blender Python environment
* Can't be used for high-precision CAD, CNC workflows

# e) Other Important Information for Component Users

# Removes previously generated wheel and Gear parts before creating a new mesh

* It fully supports Blender’s built-in functions, like the undo or redo system.
* Material nodes are simple yet realistic, designed for visual rather than physically accurate rendering.

**2. ComponentArchitecture(internal architecture that is not provided to end-users):**

# a) Design component with easy maintenance in mind

* **UI Layer:** which handles the user input through Blender panels and updates properties when it is modified
* **Core Logic Layer:** which includes the functions responsible for generating the geometry wheel and gear, creating and applying materials, and updating
* **Data Layer:** Manages property groups that store user-configurable parameters, triggering updates in the core logic when changed.

Artist and user can easily create a Gear and a wheel with minimal effort and can dynamically edit the mesh dimensions and properties

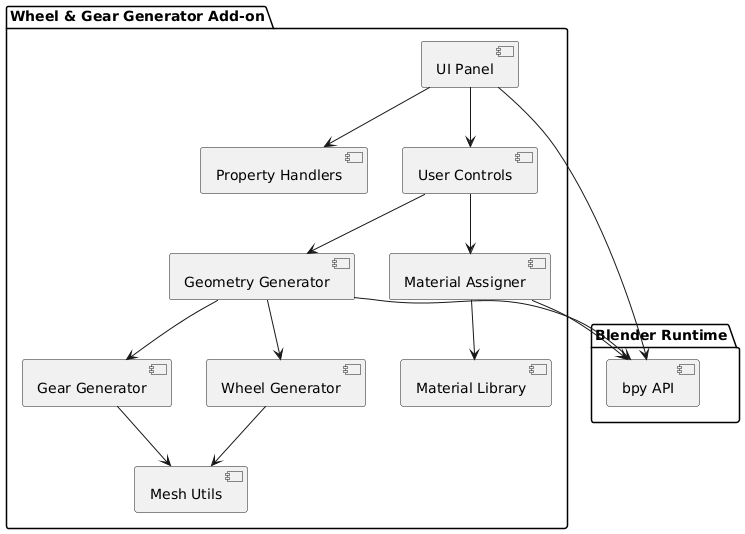


Figure 1 ComponentArchitecture

# b) Class Diagram

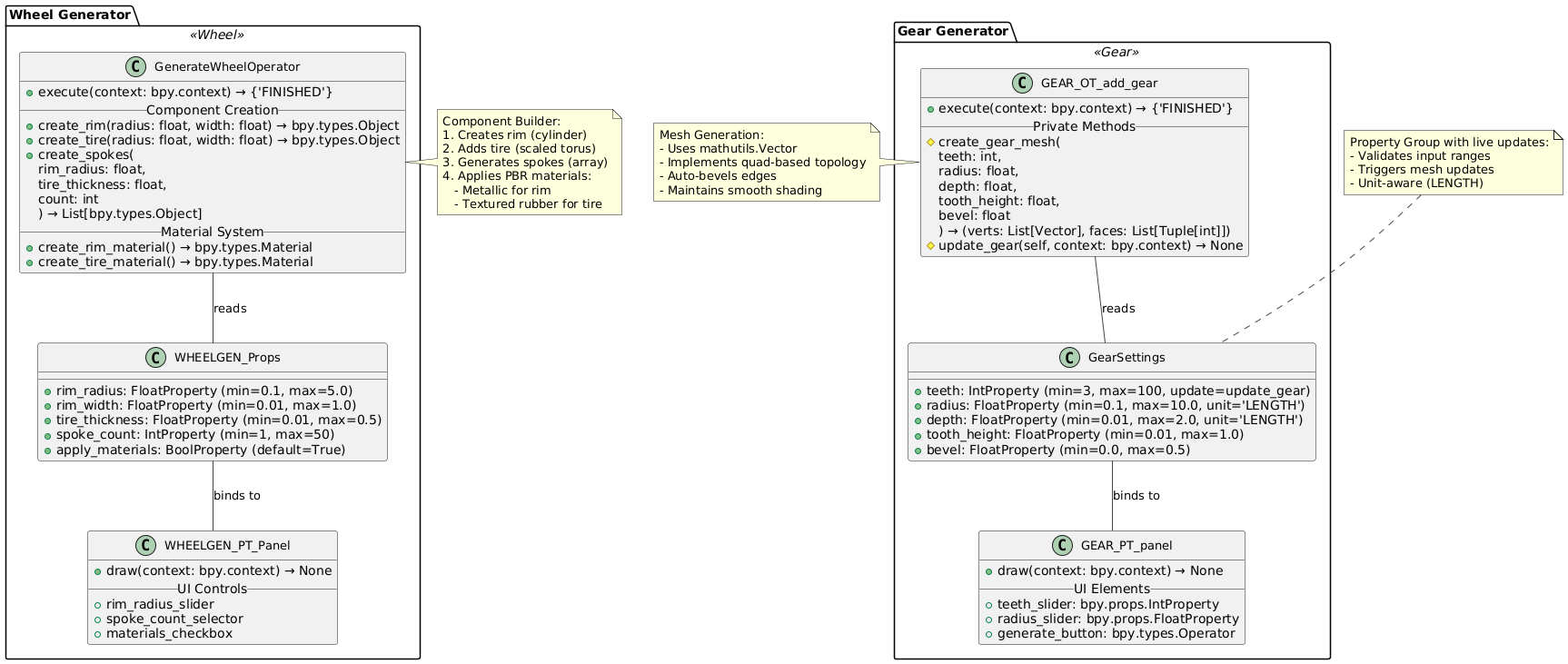


Figure 2 Class Diagram

# c) Usage scenarios (sequence diagram of predicted component usage in end user scenarios)

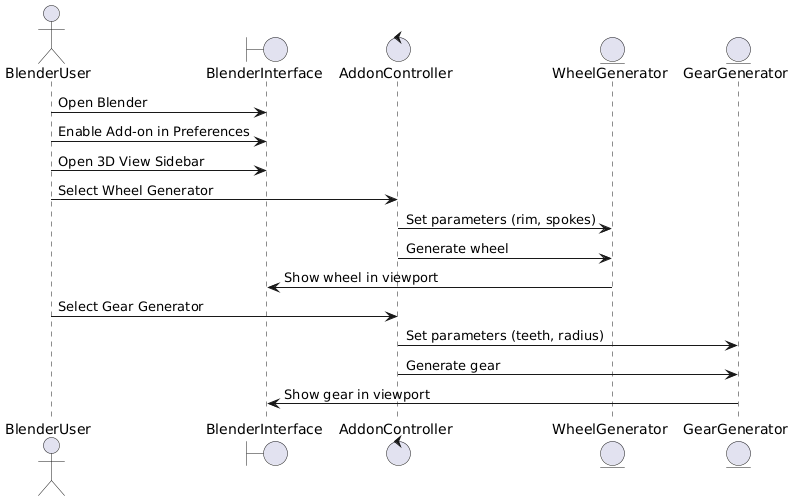


Figure 2 How to create Gear and Wheel

# d) State Diagram

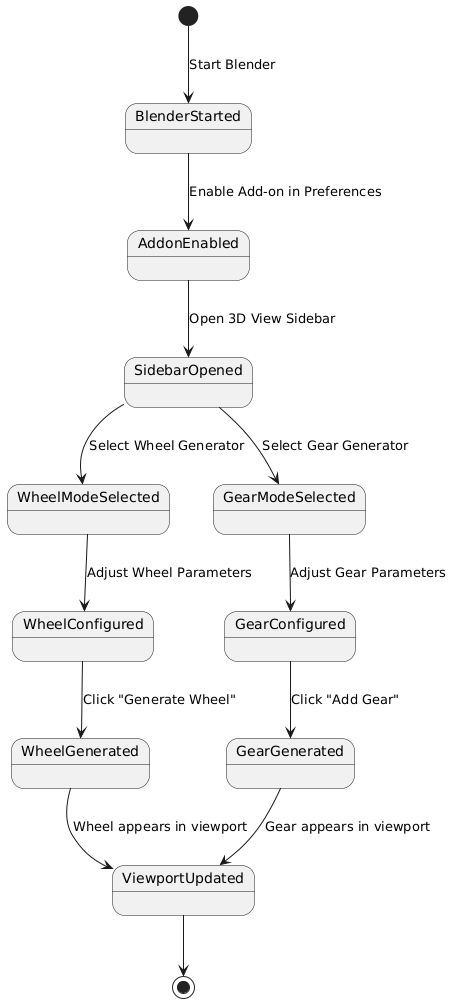


Figure 3 state diagram

# e) Deployment Diagrams (how the Component must be deployed)

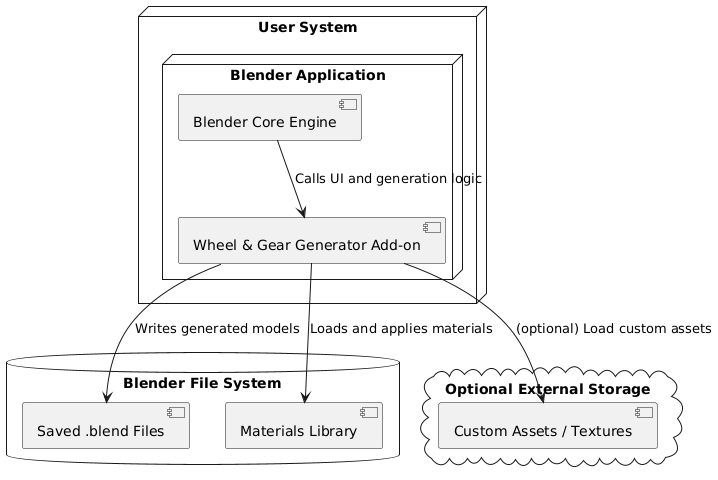


Figure 4: How the component is used

**3. Component description (documentation that is provided to component users):**

**Git link :** [**https://github.com/A-D-H-I/Blender-Generators.git**](https://github.com/A-D-H-I/Blender-Generators.git) **Gum road : https://adhithiyan.gumroad.com/l/sfyxoc**

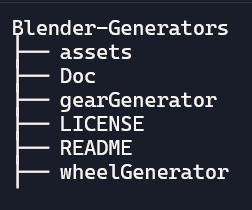
**Pyply : pip install Wheel-generator (with limitations )**

# Overview

These two Blender add-ons streamline mechanical modelling: The Gear Generator creates customizable parametric gears with real-time adjustments and non-destructive editing. The Wheel Generator builds realistic wheels with adjustable rims, spokes, and automatic rubber/metallic materials. Both support modifiers work in Blender 2.80+ and are free under the MIT license, ideal for vehicles, machinery, and prototyping. Install via Preferences > Add-ons, then access them in the sidebar (gears) or Add > Mesh menu (wheels). Combine them for drivetrain systems or vehicle setups.

**Added Demo Video**

# Folder Structure



# Features

1. Gear Generator

* Creates parametric gears with real-time updates
* Live preview while adjusting parameters
* Customizable teeth count, radius, thickness
* Automatic beveling and smooth shading
* Non-destructive workflow

2. Wheel Generator

* Produces realistic wheels with rims and tires
* Adjustable rim radius and width
* Customizable spoke count (0 for solid wheels)
* Automatic material assignment:
* Rubber texture for tires
* Metallic finish for rims/spokes
* Realistic tire tread pattern

# Setup Instructions

* Open Blender (version 3.0 or higher recommended).
* Go to Edit > Preferences > Add-ons.
* Click Install, then browse and select the wheel\_generator.zip in add-on file.
* Or download files from my GIT and select wheelGenerator for Wheel and Generator for gear
* Enable the Wheel Generator add-on by checking the box next to its name.
* Open the 3D Viewport.
* Press N to open the sidebar.
* Find the Wheel Generator tab in the sidebar panel.
* Adjust parameters such as rim radius, spoke count, and materials in the UI panel.
* Click the Generate Wheel button to create the wheel in the scene.
  + To update wheel properties after changes, click the Update Wheel button.
  + Change the values in the UI to dynamically modify the meshes

# To Add the Wheel gear, and Usage Guide:

1. Once the Add-ons are installed, you can see tool bar named WheelGen and Create on the right side of the window, or just press N.
2. UI will ask for user inputs, you can give the required dimension of the wheel and gear on the respective tabs, and click Generate Wheel to create the Wheel and Add Gear for Gear.
3. Once the Wheel and Gear is created user can dynamically change their properties.

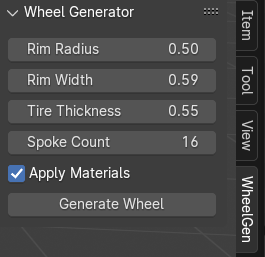
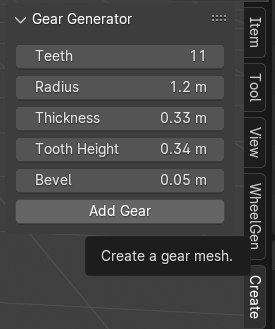


Figure 5 UI Wheel and Gear

**Interfaces**

**create\_gear\_mesh**(teeth, radius, depth, tooth\_height, bevel)

* USE: This function will calculate the raw geometric data for the gear based on the user's parameters.
* It determines the positions of all points (vertices) that make up the gear.
* It maintains the connection of gears.
* It returns a list of vertex coordinates and a list of face definitions.

**update\_gear(self, context)**

* USE: If the user changes and properties of the Gear, this will be called
* It checks if an active\_gear object exists.
* If an active\_gear exists, this function:
* Gets the new values of all gear parameters.
* Calls create\_gear\_mesh() to create a new Gear based on these updated parameters.
* Deletes the old geometry from the active\_gear object's mesh.
* Populates the active\_gear object's mesh with the new geometry.

mainly, it rebuilds the active gear in real-time as parameters are tweaked

**register()**

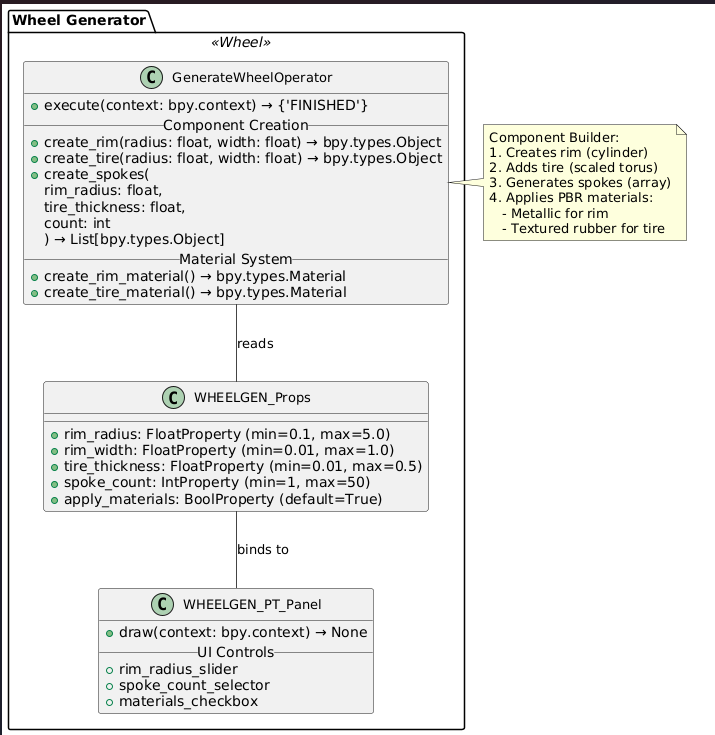
* USE : This function is called when the addon is enabled in Blender.
* It tells Blender about all the custom classes defined in the script (GearSettings, GEAR\_OT\_add\_gear, GEAR\_PT\_panel) so Blender knows how to use them.
* It specifically adds the GearSettings property group to Blender's Scene type, making scene.gear\_settings accessible for storing and retrieving the gear parameters.

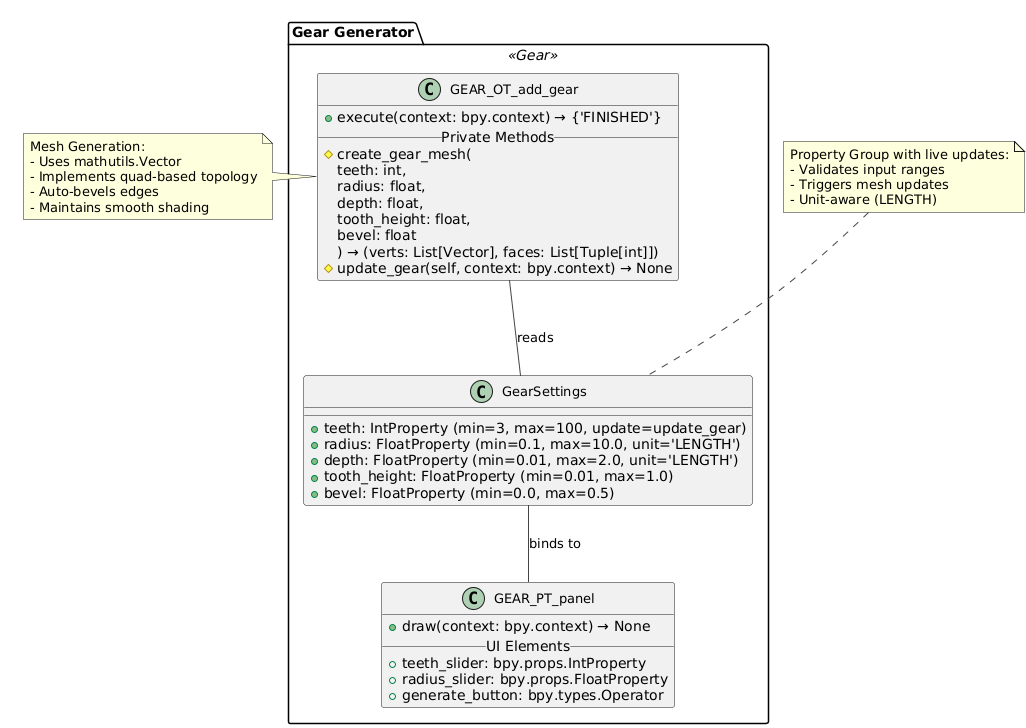
**unregister()**

* USE : This function is called when the addon is deactivated.
* It cleans up by removing the custom classes and the gear\_settings property from Blender.

# 

# UML Class Diagram – System Architecture Overview

****



# How to use the Add-on and it works:

class GenerateWheelOperator(bpy.types.Operator):

    bl\_idname = "mesh.generate\_wheel"

    bl\_label = "Generate Wheel"

    bl\_options = {'REGISTER', 'UNDO'}

    def execute(self, context):

        props = context.scene.wheel\_gen\_props

        # Clear previous wheel objects

        for obj in bpy.context.scene.objects:

            if obj.name.startswith(("Rim", "Tire", "Spoke\_")):

                bpy.data.objects.remove(obj, do\_unlink=True)

        # Create rim (a cylinder)

        rim = self.create\_rim(props.rim\_radius, props.rim\_width)

        # Create tire (a torus scaled in Z to simulate tire width)

        tire = self.create\_tire(props.rim\_radius + props.tire\_thickness, props.rim\_width + 0.05)

        # Create spokes (each spanning from the rim outer edge to the inner tire boundary)

spokes = self.create\_spokes(props.rim\_radius, props.tire\_thickness, props.spoke\_count)

        all\_parts = [rim, tire] + spokes

        if props.apply\_materials:

            metal\_mat = self.create\_rim\_material()

            rim.data.materials.append(metal\_mat)

            for spoke in spokes:

                spoke.data.materials.append(metal\_mat)

            tire.data.materials.append(self.create\_tire\_material())

        bpy.ops.object.select\_all(action='DESELECT')

        for obj in all\_parts:

            obj.select\_set(True)

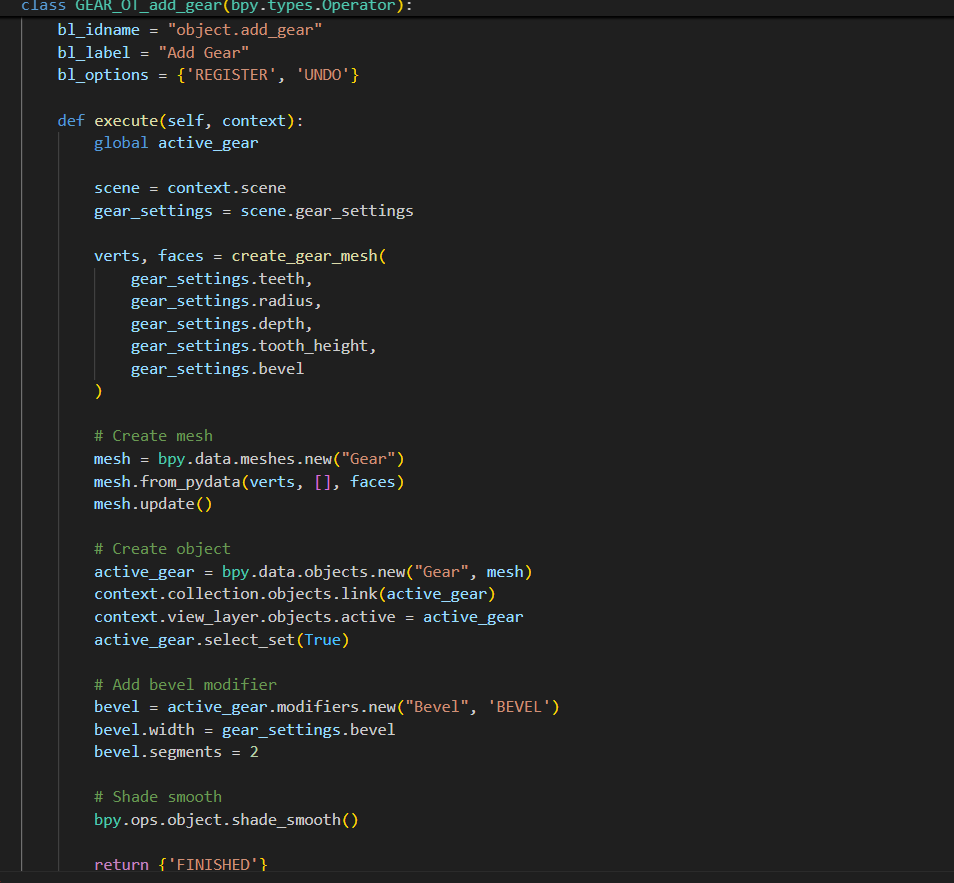
        context.view\_layer.objects.active = rim

        bpy.ops.object.join()

        bpy.context.active\_object.location = (0, 0, 0)

        return {'FINISHED'}

* **GenerateWheelOperator** will clear all previously created wheel objects
* Class the rim, tier, and spokes functions to create them based on user parameters
* Apply materials to the created wheel
* After creating all these, it combines all meshes into one with the object centered at the origin
* We can modify mesh properties using the UI



* This Function **GEAR\_OT\_add\_gear()** creates a UI panel to customize gear parameters
* Adds gear object in the window with bevel modification and smooth shading
* And this function enables the live updating function for the created gear with respect to the user's modification

**Summery:**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Wheel Generator** | **Gear Generator** |
| Parametric Controls | Yes (rim radius, width, spokes) | Yes (teeth, radius, depth, bevel) |
| Live Preview | Yes (live mesh update on change) | Yes (live mesh update on change) |
| UI Complexity | Simple, one panel | Simple, one panel |
| Modularity | Multi-part (rim, tire, spokes) | Single object |
| Geometry Detail | Moderate (cylinders and spokes) | Moderate (simplified tooth shape) |

# Requirements

* Blender version above 2.3.
* Python env with byp (if using through Pypl)

# Limitations

* Usable only inside Blender (byp needed to run)
* Can not do geometrical modification
* Requires a manual Add-on through Blender

**License**

MIT License

# Changelog

v1.0.0

* Initial release
* Wheel Generator
* Gear Generator
* Dynamic Wheel generator (updated)
* Dynamic Gear generator
* Test and run

# Support

For bug , reports, feature requests, or help installing and using the Add-on, contact adhithiyan12899@gmail.com

**4. Analysis of similar, competing components:**

# 1. Add Mesh: Extra Objects (built-in add-on )

# 2. Gear Builder by Daniel Salazar (community add-on)

The above two are what is used commercially and the popular ones that I found, which are similar to my component.

Existing add-ons offer more advanced geometry which are involved in gear teeth and wheel parts. compared to simple once, competitors often provide better material presets or shader integration, while this add-on provides very simple materials. By having this as a reference, I have updated my code from static wheel and gear generation to dynamic creation, and which I observed in those add-ons, I improved this code for a better and robust manner.

# Conclusion

The development of the Gear and Wheel generator add-ons successfully delivered parametric modelling features with user interaction and better performance with a better and user-friendly UI for better understanding, and involved low maintenance and low processing time to create the mentioned products. Through comparative analysis, it is evident that while these add-ons provide ease of use and quick generation capabilities, they currently lack some advanced features such as highly detailed geometry, realistic materials, and modular design found in leading commercial and open-source components. By the evaluation gives understanding for future enhancements.