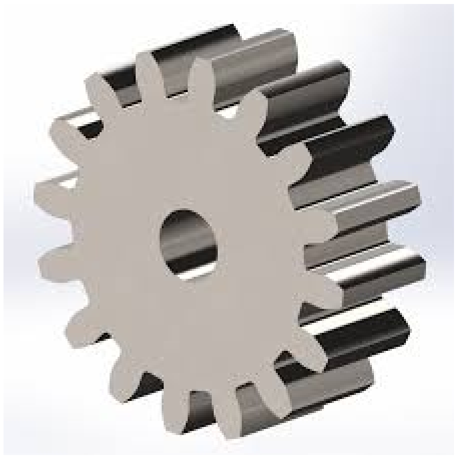
Computer Graphics

Lab Assignment #1 – Primitives and Meshes

In this assignment you will build your own parameterized object primitive and convert it to a collection of triangles, or a mesh. The primitive object that you will implement is a 3D gear primitive such as this one:



Requirements

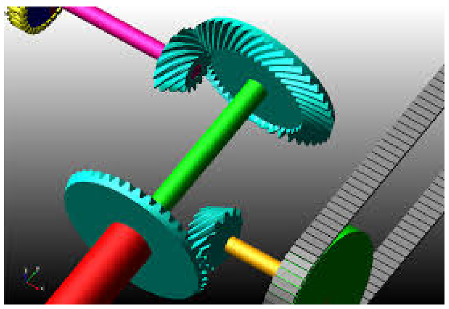
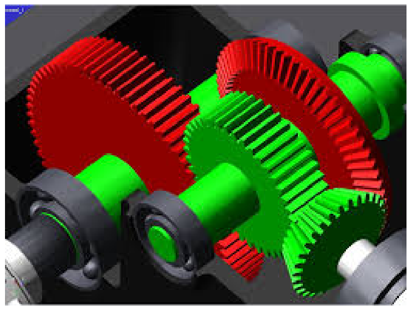
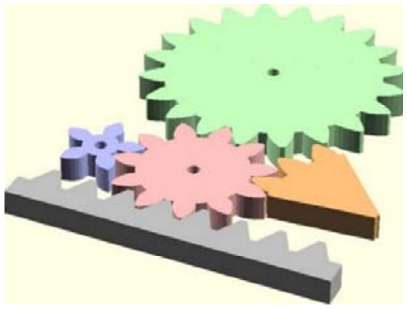
Requirement 1 – Gear Parameterization (40%): Write a function that receives at least three parameters: Center point (**c**), radius (*r*), and thickness (*d*), and produces a list of triangles approximating the boundary of the 3D shape representing the respective gear. The number of teeth can be decided automatically or parameterized but the result must look correct. The profile of each teeth can be simplified, the goal is that the overall model looks like a good approximation.

You will have to **represent your model using "indexed triangles"**: you will store the vertices in an array of vertices V and the list of faces as an array F of indices to V, as explained in the Aug 31 lab. The recommended approach is to use the arrays in the GsModel class of the support code.

Requirement 2 – Animation (20%): Implement a scene graph where multiple gears correctly rotate, forming a physically correct gear system. **Implement at least 3 connected gears of different sizes and correctly rotating** together.

Requirement 3 – Extensions (20%): Implement one of the following options:

Option 1: Add to your animated scene another gear element of different shape type, and which is connected to the other gears producing an interesting different motion, for example like a translational or pendulum movement. Below are examples illustrating multiple possibilities for achieving interesting mechanisms:

Option 2: Implement your gear primitive as an independent scene node making its own calls to OpenGL. This means you will need to implement at least two classes: a SnGear class and a GlrGear class, as explained in the Aug 30 lab. You will need to study the support code in order to understand how to access resources and link the renderer class to the scene node class.

Option 3: Implement shadow generation for your gears by using a projective transformation applied to the triangles forming your gears. In this case you will project every 3D triangle in a plane which will display the shadow, such as the ground and/or wall. The projected triangles can be all rendered in black so that the final result will resemble a shadow. The shadow has to be animated according to the animation of the gears.

Option 4: Generate a correct profile for the gear teeth. For this option you will need to search for information on how to generate an approximation of the involute gear profile. Start with the links below. Your animated gears should then display perfect contact between them.
https://en.wikipedia.org/wiki/Involute_gear
http://cadquest.com/books/pdf/gears.pdf

Requirement 4 – Explanation and Evaluation (10%)**:** Your submission must contain a readme file (.txt,.doc,.pdf) where you will **explain the main points** of your project, where to find the key parts in the code, and all other relevant information. This should not be long but it should provide enough information giving a good idea of the main points of your implementation.

You will also evaluate your project by writing in your readme file some **evaluation information including both text and a table.** The table will summarize with numbers the results of your evaluation. Examples of possible evaluations are to:
- Measure how much time it takes to build primitives with different number of triangles.
- Measure how many frames per second you achieve in different computers and with scenes with different number of triangles.
- Compare the difference in performance when using two different approaches in a given part of your implementation.

The results should be summarized in a simple table and you also have to say **where in your code you implemented your measurement functions** which provided your numbers. Even if the code is not used in your final version, you have to leave your measurement functions and calls there. Comment the calls which are not being used in your final version but leave them there. This will be important to check which exact operations are include in each computed time, etc.

Here is an example of a table you could include in your readme file:

```
CPU         GPU         FPS   SCENE SIZE
Intel I5    NVidia XX   25    50789 triangles
Intel I5    NVidia XX   29    20789 triangles
Intel I7    NVidia XX   30    20789 triangles
...
etc
```

Requirement 5 – Overall Quality (10%)**:** Your project has to look good and be well implemented with an organized code. Note that quality does not mean complexity. The appearance of your models can be very simple, focus on creating a **scene that looks interesting and is correct.**

Grading and Submission

Please read files grading.txt and rules.txt posted at CatCourses.