Jojuss

Zeste de savoir

Découvrir la conception 3D avec CadQuery

19 mai 2021

Table des matières

Aujourd'hui j'aimerai vous présenter un projet pour le quel je porte une attention croissante et qui se nomme CadQuery.

1. CadQuery qu'est ce que c'est ?

CadQuery est une librairie python Opensource qui permet de créer des modèles 3D paramétriques. Le point positif de cette librairie est qu'elle permet de créer des modèles CAO [??] en BREP.

1.1. BREP 🍊 ?

BREP pour Boundary Representation est un paradigme de modélisation 3D. C'est une façon de représenter des objets 3D numériquement en définissant ces objets par des représentation mathématiques de ce qui les composent (point, courbe, surface, volume).

L'autre paradigme de modélisation est le paradigme CSG [??] ce paradigme définit un objet 3D par une composition d'additions ou de soustractions de formes 3D élémentaires dites primitives (sphère, cube, cone, tore, etc.)

Mais pourquoi avoir un paradigme BREP est un avantage me direz vous? Et bien car modéliser en BREP permet une bien plus grande liberté dans la modélisation, par exemple il est très facile de créer un congé avec un paradigme BREP alors que c'est une tâche très ardue pour un paradigme CSG. (Et oui comment arrondir des arrêtes d'une forme quelconque en y ajoutant ou enlevant des sphères ou des rectangles)

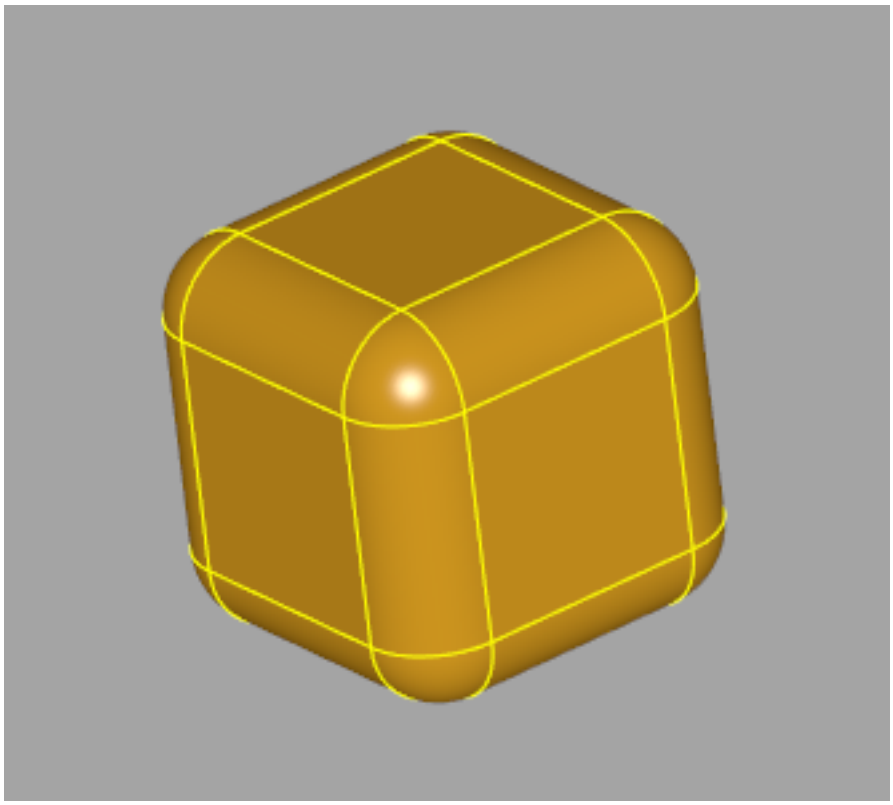

FIGURE 1.1. – Un cube avec des congés

1.2. Tu nous cache quelque chose

Mais comment CadQuery fait pour générer ces modèles 3D qui ont l'air si compliqués à créer? 🤔 Et bien en réalité tout comme son homologue FreeCad, Cadquery se base sur la librairie C++ OpenCascade Technology . Dans le milieu on appelle ce type de librairie un noyau géométrique. Il en existe plusieurs sur le marché par exemple Parasolid ou bien ACIS qui équipent des logiciels comme SolidWorks ou bien SpaceClaim. Le fait que CadQuery utilise sous la capuche OpenCascade Technology (OCCT) n'est pas un hasard, c'est en fait le seul noyau géométrique BREP open-source!

C'est une chance d'avoir à disposition une telle librairie gratuitement car habituellement les logiciels de conception 3D (et donc leurs noyaux géométriques) coutent très cher! Et on peut être fier OCCT a été créé en France!

??: Conception Assistée par Ordinateur ??: Constructive Solid Geometry

2. D'accord mais ils sont où mes modèles 3D ?

Un peu de patience, on y vient 🍊 . Avant de présenter quoique ce soit je vous invite si vous le souhaitez à télécharger CadQuery pour pouvoir essayer par vous même. Pour savoir comment faire rendez vous sur:

— La page GitHub du projet
— La documentation officielle

La documentation ainsi que le `readme.md` du GitHub expliquent comment installer la librairie et plein d'autres choses.

2.1. Créons notre premier modèle

Allez c'est parti on va créer notre premier modèle. Et pas n'importe lequel nous allons créer le cube avec des congés que l'on a vu plus haut. On créer un nouveau fichier python et c'est parti!

```
1 import cadquery as cq
2 box = cq.Workplane("XY").box(5,5,5).edges().fillet(1)
```

On exécute et voilà, notre premier modèle est créé!

> Mais je ne vois rien?

Ah et bien oui de base CadQuery n'est pas un logiciel avec IHM mais une simple librairie. Si on veut voir notre modèle on peut l'exporter par exemple en `.stl`:

```
1 import cadquery as cq
2 box = cq.Workplane("XY").box(5,5,5).edges().fillet(1)
3 box.val().exportStl("box.stl")
```

Si j'ouvre le fichier dans Ultimaker Cura voici ce que j'obtiens:

2. D'accord mais ils sont où mes modèles 3D?

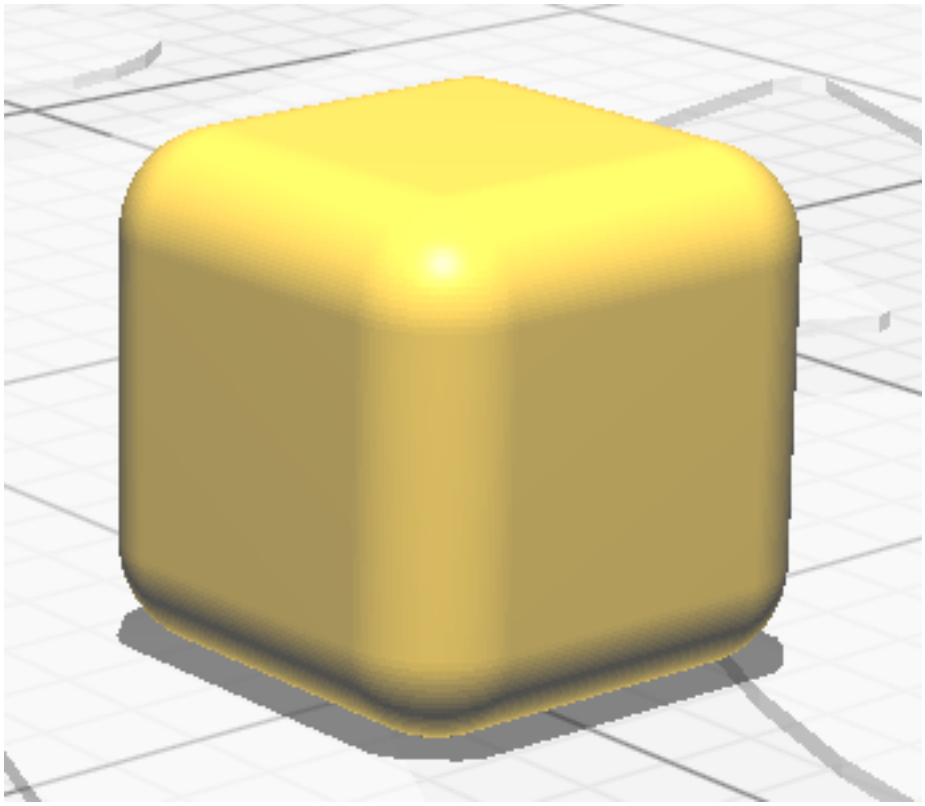

FIGURE 2.2. – box.stl

2.2. Mais ce n'est pas pratique pour débuguer mon modèle !

En effet si l'on veut créer un modèle complexe c'est quand même embêtant de ne pas pouvoir voir ce qu'on fait pendant notre conception. Heureusement pour ça nous avons deux solutions:

- L'environnement de développement intégré CQ-Editor qui permet de pouvoir coder et débuguer visuellement son modèle. (Vous l'aurez remarqué la première image du cube avec congés a été prise via CQ-Editor)
- Jupyter Lab avec la librairie jupyter_cadquery qui permet d'afficher les modèles CadQuery dans un Jupyter notebook via Jupyter Lab

Maintenant que vous avez tout le nécessaire je vais pouvoir vous expliquer un peu plus "comment ça marche".

2.3. L'API fluide

Le code que je vous ai montré pour créer le cube avec congés est particulièrement court: 1 ligne. Pour permettre une telle simplicité CadQuery enrobe la librairie OCCT de 3 différentes couches:

- OCP: C'est la couche très bas niveau, OCP est un binding python d'OCCT ainsi on peut utiliser OCCT via python en utilisant OCP. Ce type de binding est semblable à un binding tel que pyqt pour Qt.
- L'implémentation d'OCCT par CadQuery: C'est une simplification des concepts d'OCCT/OCP qui facilitent leur utilisation, par exemple pour créer une spline:

```python
import cadquery as cq
pts = [cq.Vector(0,0,0), cq.Vector(0,0,0) ,cq.Vector(0,0,0)]
spline = cq.Edge.makeSpline(pts)
```

Je vous invite à aller voir le code source de `makeSpline` pour vous convaincre que cela nous simplifie bel et bien la vie.

— L'API fluide: Finalement la dernière couche de CadQuery que l'on appelle l'API fluide permet de créer des modèles très complexes en seulement une ligne de code. Pour cela elle se base sur un concept clé:

2.4. L'objet `Workplane`

L'objet `Workplane` est un concept essentiel de CadQuery, après initialisation chaque opération que l'on va effectuer (via un appel à une méthode) va venir s'ajouter à la "stack". La "stack" est la pile d'opération qui définissent notre modèle 3D.

C'est un peu déroutant au début car assez particulier comme approche et peu pythonique, on peut le voir comme un language de commande.

Pour mieux se le représenter voici un exemple un peu plus long que notre cube avec congés:

```python
import cadquery as cq
from cadquery.selectors import AreaNthSelector

model = (cq.Workplane("XY")
       .box(12,10,5)
       .edges("<X")
       .chamfer(1)
       .faces(">Z")
       .workplane(centerOption = "CenterOfMass")
       .lineTo(5,3)
       .lineTo(4,4)
       .lineTo(-2,1)
       .radiusArc((-4,-2),-5)
       .close()
       .twistExtrude(5,50)
       .faces("<<Z[2]")
       .wires(AreaNthSelector(0))
       .fillet(0.5)
       )
```

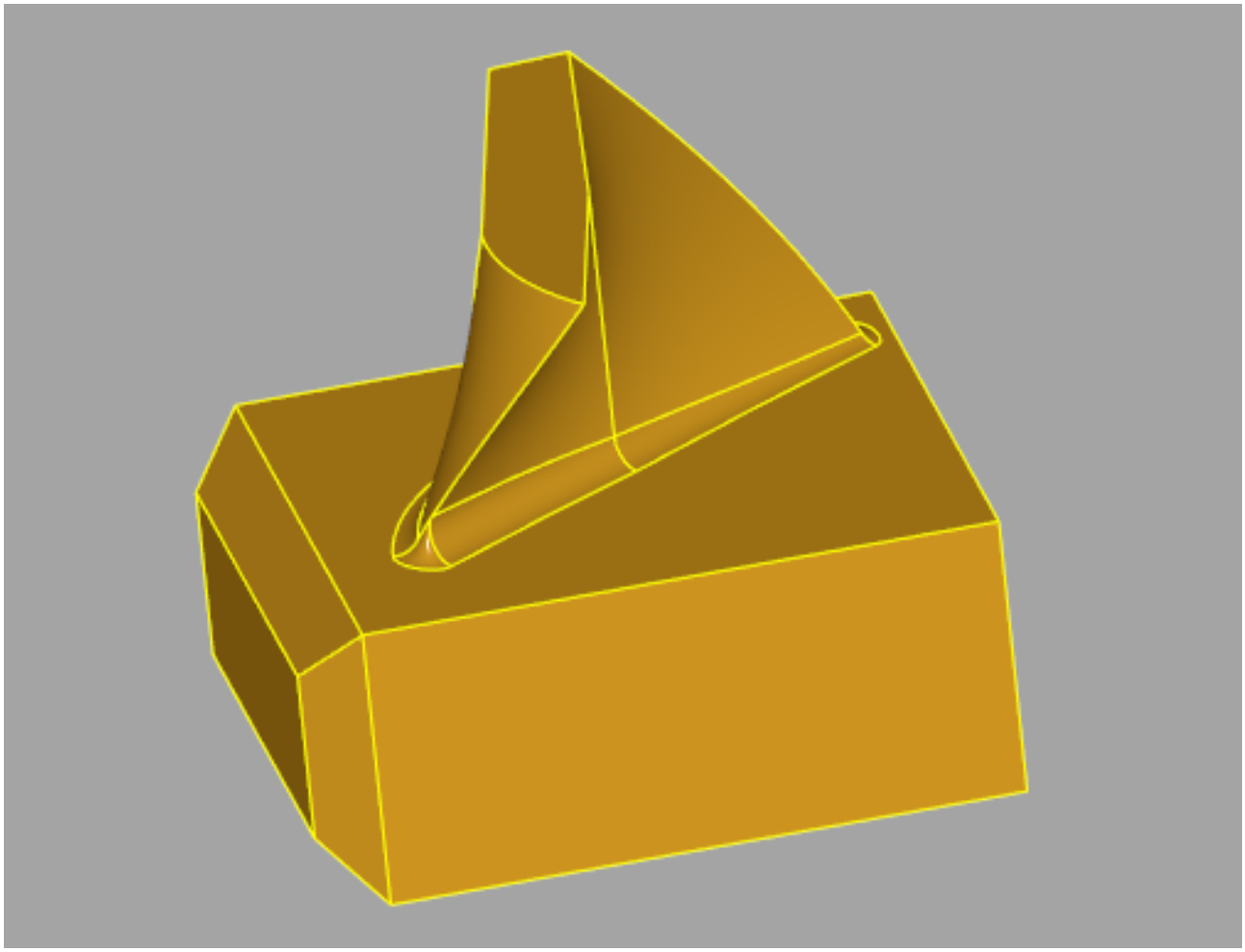

FIGURE 2.3. – Un modèle moche mais complexe !

C'est pas très joli mais ça permet de montrer un peu plus en profondeur la philosophie de l'API fluide de CadQuery, pour plus de lisibilité j'ai ajouté des parenthèses et sauté des lignes mais dans les faits mon modèle ne fait bel et bien qu'une seule ligne de python .

Cela dit quand bien même on peut effectuer beaucoup de choses avec l'API fluide, rien ne nous empêche de mixer les 3 couches de complexités entre elles pour étendre quasiment infiniment le champ de possibles.

3. Montre m'en plus !

Bon maintenant que nous avons vu quelques exemples somme toute relativement simples je vais maintenant vous montrer ce qu'il est possible de faire quand on commence à maitriser la bête.

Un point fort de CadQuery est qu'il est facile de développer des librairies annexes, des "plugins" pour étendre ses capacités. J'en ai notamment créé une qui s'appelle `gear_generator` qui comme vous l'aurez deviné permet de créer des engrenages très simplement:

```python
import cadquery as cq
from gear_generator import BevelGearSystem

m = 2
z1 = 15
z2 = 23
width = 13
system = BevelGearSystem(m, z1, z2, width)
pinion, gear = system.build()
```

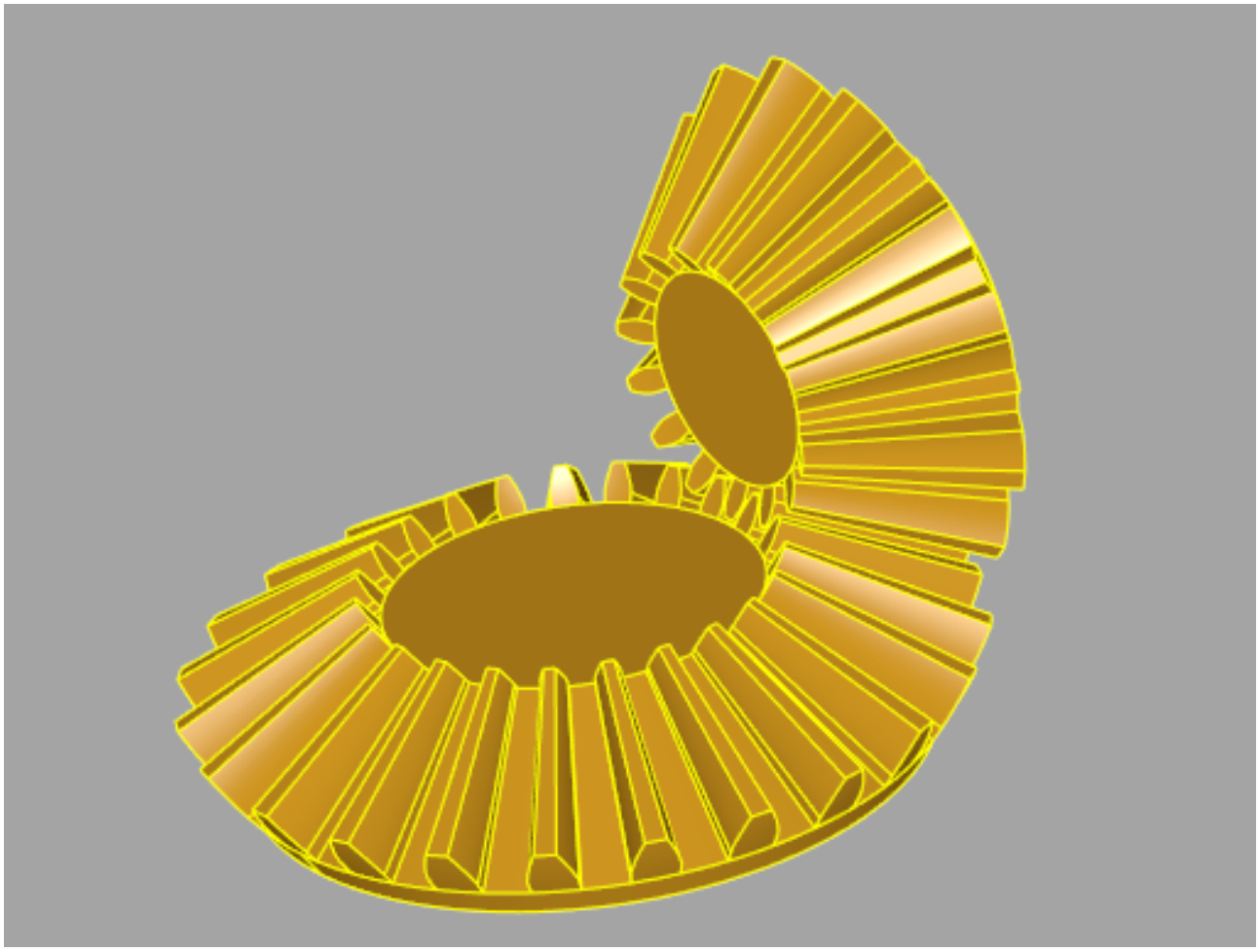

FIGURE 3.4. – Couple d'engrenages coniques droits

Et en changeant un simple paramètre:

```python
import cadquery as cq
from gear_generator import BevelGearSystem

m = 2
```

```
z1 = 15
z2 = 23
width = 13
helix_angle = 30
system = BevelGearSystem(m, z1, z2, width, helix_angle =
    helix_angle)
pinion, gear = system.build()
```

FIGURE 3.5. – Couple d'engrenages coniques hélicoïdaux

4. Conlusion

Nous voilà arrivé au bout. J'ai démarré la rédaction de ce tuto en voulant faire une brève présentation de CadQuery qui s'est en réalité transformée en mini tuto 🍊 .

Si vous en voulez plus je vous invite à consulter la documentation et surtout à poser vos questions sur GitHub, le google groups ou encore le discord du projet, c'est une petite communauté mais les gens sont très sympas et essayerons toujours de vous aider dans vos modélisation.

4. Conlusion

C'est la première fois que je rédige un contenu sur ZdS donc n'hésitez pas à me faire des retours!

Merci