pythonocc-examples Documentation

Release 0.17

Thomas Paviot and others

Dec 28, 2019

Contents

Contents:

CHAPTER 1

Introduction

pythonocc-core est une surcouche python à la bibliothèque C++ OpenCascade Community Edition (OCE).

pythonocc est un noyau CAO 3D (Conception Assistée par Ordinateur) : vous pouvez créer des géométries/topologies, selon la représentation Boundary Representation(BRep), qui décrivent les formes d'objets mécaniques qu'il est possible de fabriquer sur des machines automatisées à commande numérique.

Avant toute chose, la lecture de ce document suppose que vous ayez correctement installé/configuré pythonocc dans sa version 0.16.0. Vous devez donc préalablement vérifier que cette installation s'est bien passée.

Depuis un prompt python (par exemple ipytohn), taper la séquence d'instructions suivante

```
In [1]: import OCC
In [2]: OCC.VERSION
Out[2]: '0.16.0-dev'
```

Une erreur à cet instant signifie que l'installation n'est pas correcte. Reportez-vous à la section installation pour résoudre le problème.

1.1 A propos de ce document

Ce document propose un bref aperçu des fonctionnalités de pythonocc. Le périmètre fonctionnel de pythonocc est très étendu, ce document ne couvre qu'une petite partie de ce qu'il est possible de faire.

L'objectif de ce document est de permettre au lecteur une prise en main de la bibliothèque pour permettre une découverte approfondie en autonomie.

Dans les sections qui suivent, on présente, en les expliquant, les différents exemples qui sont disponibles dans le dossier /examples. L'étude, et a fortiori la modification, de ces exemples doivent permettre une prise en main progressive qui balaie l'ensemble des domaines suivants : la modélisation géométrique, l'exploration de la topologie, l'exécution d'algorithmes complexes (opérations booléennes, fonctions de balayage etc.), l'export vers d'autres logiciels par l'intermédiaire de formats de fichiers standardisés.

A chaque section est associée un fichier d'exemple qu'il est possible d'exécuter pour examiner le résultat produit.

CHAPTER 2

Helloworld !

2.1 Résumé

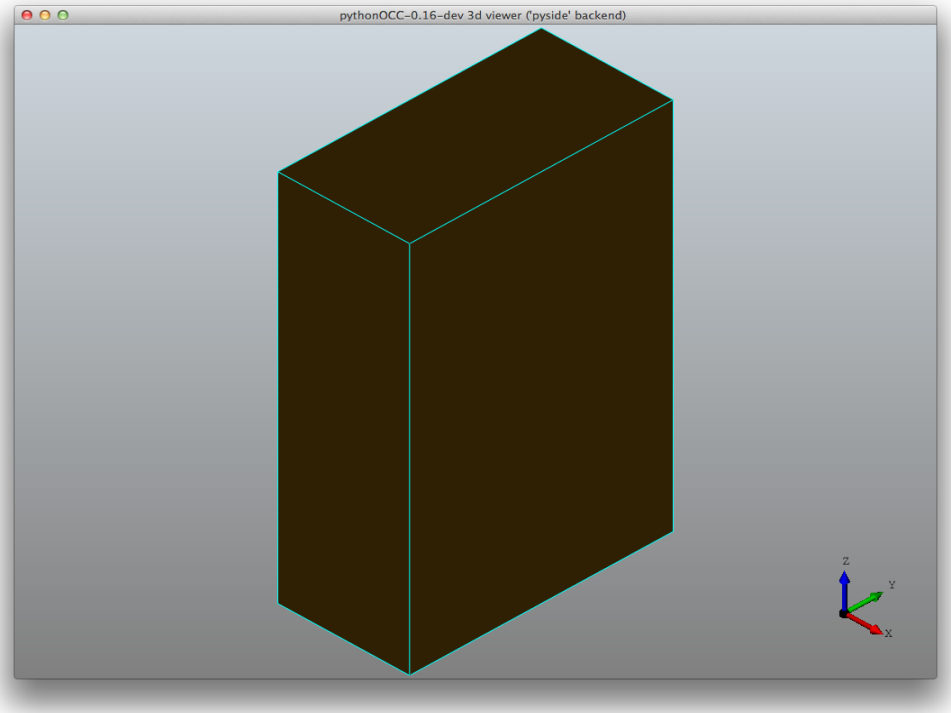
Dans cet exemple, un parallélépipède rectangle est créé et affiché dans une interface graphique rudimentaire. C'est l'exemple le plus simple qui puisse être imaginé.

2.2 Lancer l'exemple

Pour lancer le script

```
$ python core_helloworld.py
```

2.3 Résultat



2.4 Description

La première ligne permet d'importer la fonction permettant la création de l'interface graphique rudimentaire fournie avec pythonocc. On la retrouvera chaque fois que l'on souhaitera visualiser la géométrie créée.:

```
from OCC.Display.SimpleGui import init_display
```

Le module BRepPrimAPI qui offre un ensemble de classe pour créer des sphères, tores, cylindres etc. Ici, c'est la classe BRepPrimAPI_MakeBox, permettant de créer des parallélépipèdes, qui est importée:

```
from OCC.BRepPrimAPI import BRepPrimAPI_MakeBox
```

La ligne suivante initialise l'interface graphique. Par défaut, la fonction init_display recherche les gestionnaire d'interface utilisateur PyQt, PySide, et en dernier lieu wxPython:

```
display, start_display, add_menu, add_function_to_menu = init_
↪display()
```

La classe BRepPrimAPI_MakeBox est initialisée à l'aide de 3 paramètres : la largeur, la profondeur, la hauteur:

```
my_box = BRepPrimAPI_MakeBox(10., 20., 30.).Shape()
```

Puis la géométrie résultante est envoyée vers le circuit de visualisation:

```
display.DisplayShape(my_box, update=True)
```

Le paramètre *update*, initialisé à la valeur True, indique que l'affichage doit être mis à jour après l'affichage de la boîte (en particulier, cette opération implique que la boîte occupe l'ensemble de la fenêtre). Le trièdre en base à droite de la fenêtre graphique permet de s'assurer que la dimension 10.0 est suivant x, celle de 20.0 suivant y et enfin celle de 30 suivant z.

Enfin, la boucle principale du moteur d'interface graphique est lancée:

```
start_display()
```

2.5 Exercices d'application

- modifier le programme précédent pour afficher une sphère de rayon 10 en utilisant la classe BRepPrimAPI_MakeSphere
- modifier le programme précédent pour créer et afficher un cylindre de hauteur 200 et de diamètre 4 (voir à ce sujet la classe BRepPrimAPI_MakeCylinder)

CHAPTER 3

Affichage dans un navigateur Internet

3.1 Résumé

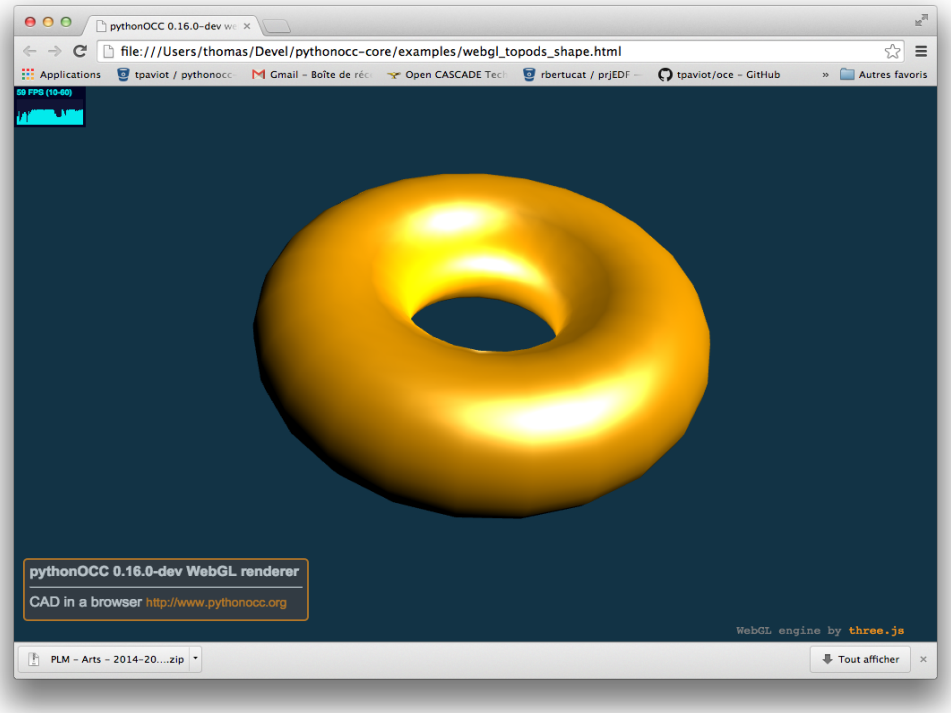
Dans cet exemple, un tore est créé et affiché dans un nabigateur internet. C'est la technologie WebGl, disponible depuis quelques mois dans les dernières versions des navigateurs Chrome, FireFox ou Safari, qui permet la visualisation interactive de formes 3D dans la navigateur. Cette technologie évite l'installation de bibliothèques graphiques (GUI) lourdes à installer et utiliser, de plus cet afichage se fait sans plugin.

3.2 Lancer l'exemple

Pour lancer le script

```
$ python core_webgl_threejs_torus.py
```

3.3 Résultat



3.4 Description

La première ligne permet d'importer la fonction permettant la création de l'interface graphique rudimentaire fournie avec pythonocc. On la retrouvera chaque fois que l'on souhaitera visualiser la géométrie créée.:

```
from OCC.Display.SimpleGui import init_display
```

Le module BRepPrimAPI qui offre un ensemble de classe pour créer des sphères, tores, cylindres etc. Ici, c'est la classe BRepPrimAPI_MakeBox, permettant de créer des parallélépipèdes, qui est importée:

```
from OCC.BRepPrimAPI import BRepPrimAPI_MakeBox
```

La ligne suivante initialise l'interface graphique. Par défaut, la fonction init_display recherche les gestionnaire d'interface utilisateur PyQt, PySide, et en dernier lieu wxPython:

```
display, start_display, add_menu, add_function_to_menu = init_
↪display()
```

La classe BRepPrimAPI_MakeBox est initialisée à l'aide de 3 paramètres : la largeur, la profondeur, la hauteur:

```
my_box = BRepPrimAPI_MakeBox(10., 20., 30.).Shape()
```

Puis la géométrie résultante est envoyée vers le circuit de visualisation:

```
display.DisplayShape(my_box, update=True)
```

Le paramètre *update*, initialisé à la valeur True, indique que l'affichage doit être mis à jour après l'affichage de la boîte (en particulier, cette opération implique que la boîte occupe l'ensemble de la fenêtre). Le trièdre en base à droite de la fenêtre graphique permet de s'assurer que la dimension 10.0 est suivant x, celle de 20.0 suivant y et enfin celle de 30 suivant z.

Enfin, la boucle principale du moteur d'interface graphique est lancée:

```
start_display()
```

3.5 Exercices d'application :

- modifier le programme précédent pour créer et afficher un cylindre de hauteur 200 et de diamètre 4 (voir à ce sujet la classe BRepPrimAPI_MakeCylinder)

CHAPTER 4

Indices and tables

- genindex
- modindex
- search