Volmdlr Documentation

Release 0.2.5.dev26-dirty

DessIA Technologies

Mar 02, 2021

Contents

1 Getting started 3

2 Tutorial 5

3 Primitives2D 7

4 Primitives3D 9

5 Developpements 11

6 Features 13

7 Galery 15

8 Indices and tables 21

Index 23

Volmdlr is a volume modeler, which is used as a CAD plateform.

It is simple to understand and operate. With it, you can create a lot of 3D Models easily. Check the follow examples to see what you can do with Volmdlr.

Volmdlr uses Babylonjs as a display.

CHAPTER 1

Getting started

1.1 Install

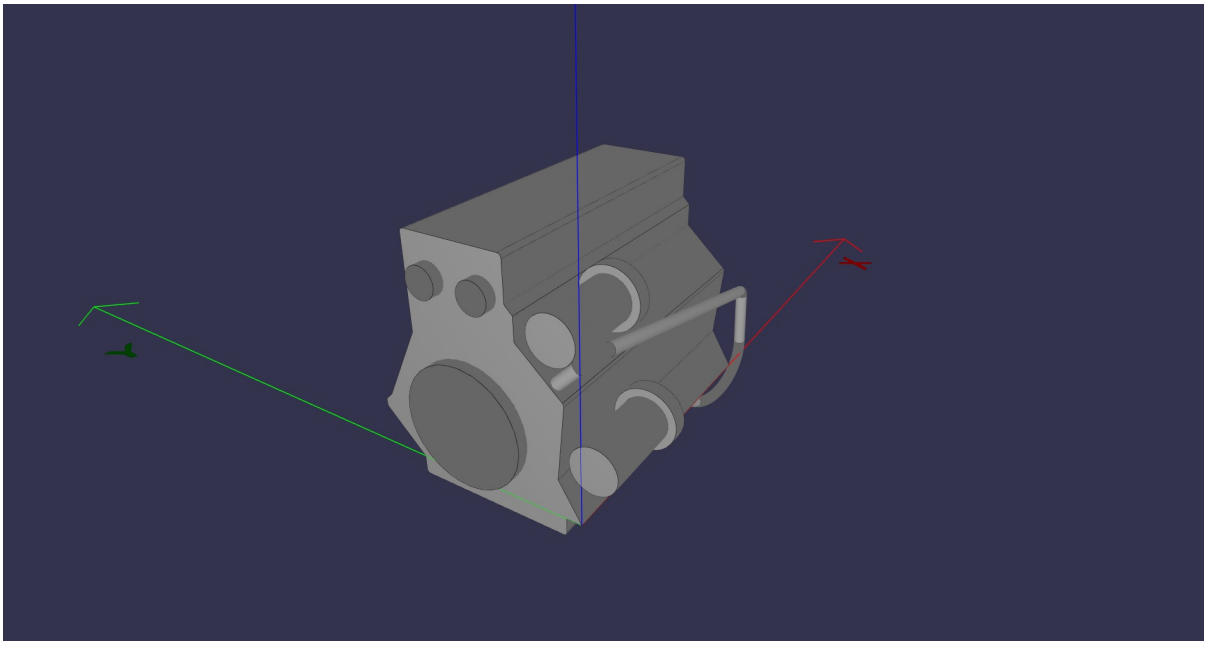
```
pip(3) install Volmdlr
```

1.2 Important

Before using Volmdlr, be sure to have a compiler C/C++ (not necessary on Linux). N.B : With Windows you have to download one and allows it to read Python's code.

1.3 CAD

The following picture is here to show you about the power of Volmdlr.



https://github.com/Dessia-tech/volmdlr/blob/distancewire/scripts/Tutorial/CasTest.py

This engine is made from ExtrudedProfile and RoundedLineSegment2D. With CylindricalFace3D and ToroidalFace3D too. You can see Sweep as presented before. If you want to load it, run the code above, and if you are curious, check on Primitives3D how to create Faces.

CHAPTER 2

Tutorial

This guide can help you using tools developed with the Volume Modeler Volmdlr.

2.1 Models

In volmdlr, you can create Models as a face contained in a plane, Cylinder (CylindricalFace3D), a Tore (ToroidalFace3D) and more (BSpline, ConicalFace, SphericalFace WIP). Please refers to primitives-label in 3D advanced primitives to know how it works.

To show the final result, we use Babylonjs.

2.2 VolumeModel

A model is set instanciating a VolumeModel:

Prior to that, primitives have to be instanciated. See primitives-label.

2.3 FreeCAD binding (Optional)

Once a VolumeModel is instanciated, one can call the FreeCADExport method:

2.4 Sweep

'Sweep' creates pipes with the following tool :

To understand how Sweep works, see primitives3D-label in 3D advanced primitives.

2.5 Shell3D

If you want to create a simple Face3D or more, you can create them thanks to our tools see primitives3D-label in 3D advanced primitives.

Example

```
>>> You can create a Shell3D :
      import volmdlr as vm
      (create faces with our examples of each Face3D)
      shell = vm.Shell3D([face1, face2, etc..])
      volumodel = vm.VolumeModel([shell])
      volumodel.babylonjs() (to show it in babylonjs interface)
```

2.6 More Tutorials

The scripts folder contains some examples of the capabilities of volmdlr:

https://github.com/Dessia-tech/volmdlr/tree/master/scripts

CHAPTER 3

Primitives2D

3.1 2D core primitives

class volmdlr.**Vector2D**

frame_mapping
side = 'old' or 'new'

norm

Returns norm of vector

normalize
normalize the vector modifying its coordinates in place

translation

Parameters **offset** – an other Vector2D

class volmdlr.**Point2D**

class volmdlr.**Basis2D**
Defines a 2D basis :param u:Vector2D: first vector of the basis :param v:Vector2D: second vector of the basis

normalize
normalize the basis modifying its coordinates in place

class volmdlr.**Frame2D**
Defines a 2D basis :param origin:Point2D: origin of the basis :param u:Vector2D: first vector of the basis :param v:Vector2D: second vector of the basis

3.2 2D advanced primitives

CHAPTER 4

Primitives3D

4.1 3D core Primitives

class volmdlr.**Vector3D**

deterministic_unit_normal_vector
Returns a deterministic normal vector

frame_mapping
side = 'old' or 'new'

normalize
normalize the vector modifying its coordinates

random_unit_normal_vector
Returns a random normal vector

rotation
rotation of angle around axis. Used Rodrigues Formula:

https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula

x_rotation
rotation of angle around X axis.

y_rotation
rotation of angle around Y axis.

z_rotation
rotation of angle around Z axis.

class volmdlr.**Point3D**

class volmdlr.**Basis3D**
Defines a 3D basis

Parameters

- **u:Vector3D** – first vector of the basis
- **v:Vector3D** – second vector of the basis
- **w:Vector3D** – third vector of the basis

classmethod from_two_vectors(*vector1: volmdlr.core_compiled.Vector3D*, *vector2: volmdlr.core_compiled.Vector3D*) → volmdlr.core_compiled.Basis3D

Create a basis with first vector1 adimensionned, as u, v is the vector2 substracted of u component, w is the cross product of u and v

normalize

normalize the basis modifying its coordinates in place

class volmdlr.**Frame3D**

Defines a 3D frame :param origin:Point3D: origin of the basis :param u:Vector3D: first vector of the basis :param v:Vector3D: second vector of the basis :param w:Vector3D: third vector of the basis

new_coordinates

You have to give coordinates in the global landmark

old_coordinates

You have to give coordinates in the local landmark

rotation

Rotate the center as a point and vectors as directions (calling Basis)

4.2 Faces3D

https://github.com/Dessia-tech/volmdlr/blob/distancewire/scripts/Tutorial/PlaneFace.py

https://github.com/Dessia-tech/volmdlr/blob/distancewire/scripts/Tutorial/Cylinder.py

https://github.com/Dessia-tech/volmdlr/blob/distancewire/scripts/Tutorial/Tore.py

4.3 3D advanced primitives

CHAPTER 5

Developpements

5.1 Roadmap

- BSplines in 2D
- NURBS in 3D
- Computation of basis geometry in 3D primitives, task given to freecad thanks to the binding.
- Finite elements models with a binding to CodeAster

5.2 Release notes

5.2.1 v0.1

- Documentation
- Basis and Frames
- adapt radius in roundedlines
- various fixes

CHAPTER 6

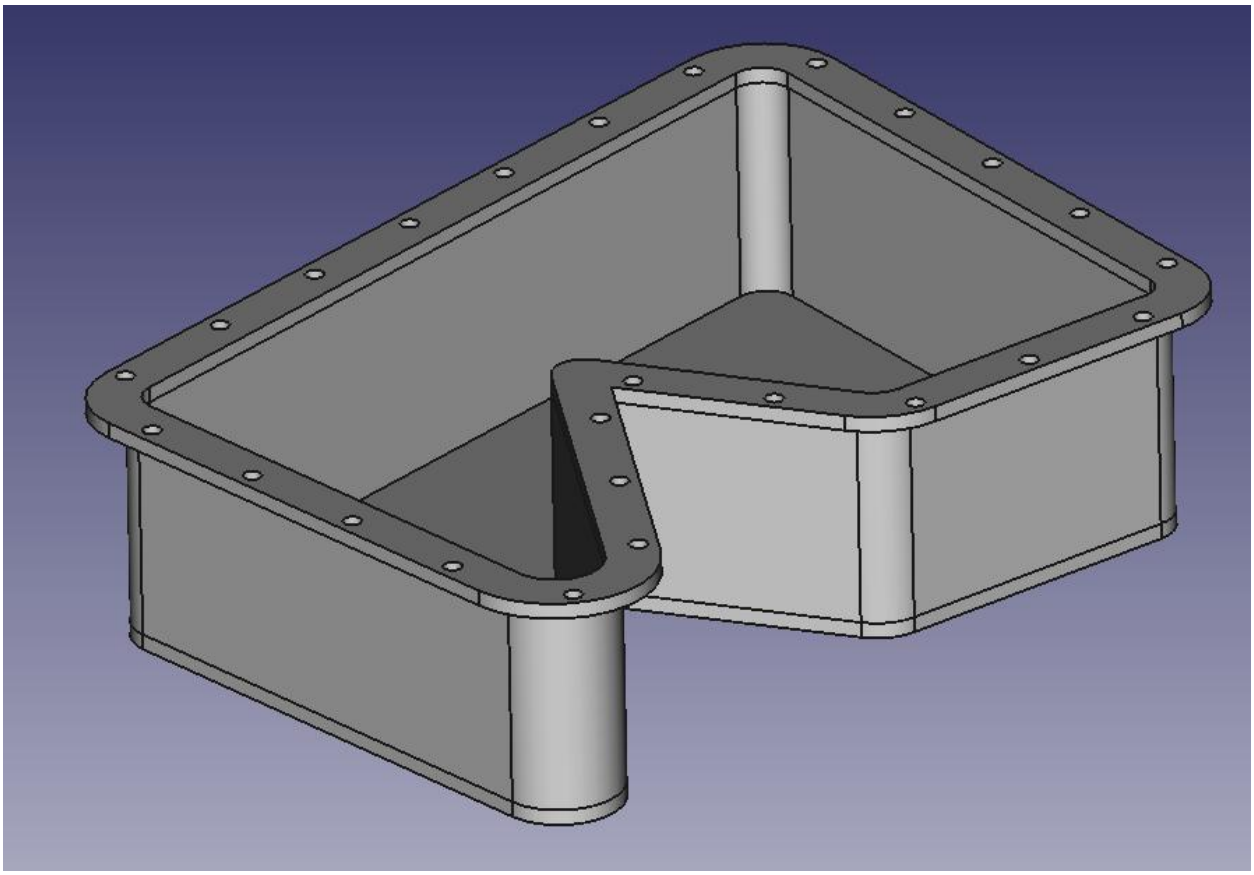
Features

- A geometrical description of basis objects and primitives built on top: lines, points contours...
- Computational tools for creating the geometry and to analyse it (area, second moment area, intersections, closest point, distance)
- FreeCAD binding for exporting in .fcstd, .step, .stl
- Volmdlr is able to read .step files (WIP)

CHAPTER 7

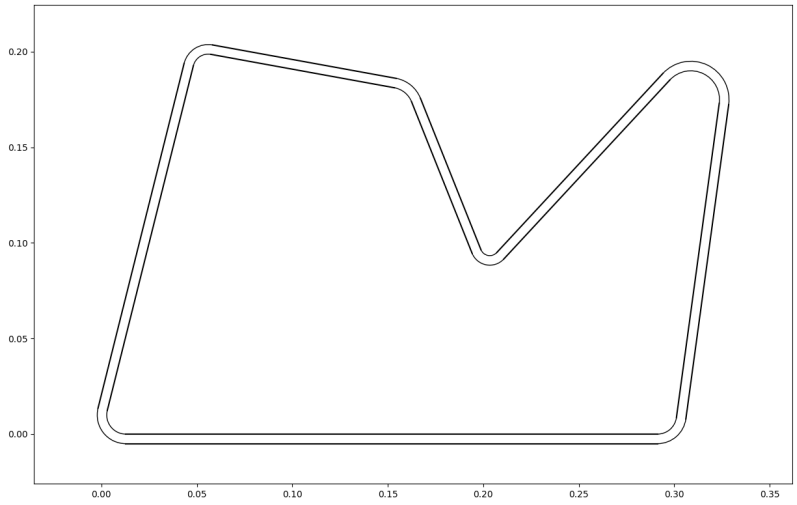
Galery

7.1 Casing



https://github.com/Dessia-tech/volmdlr/blob/master/scripts/casing.py

A casing is defined by a 2Dcontour formed with the primitive RoundedLineSegment2D. This contour is offset by the casing width.

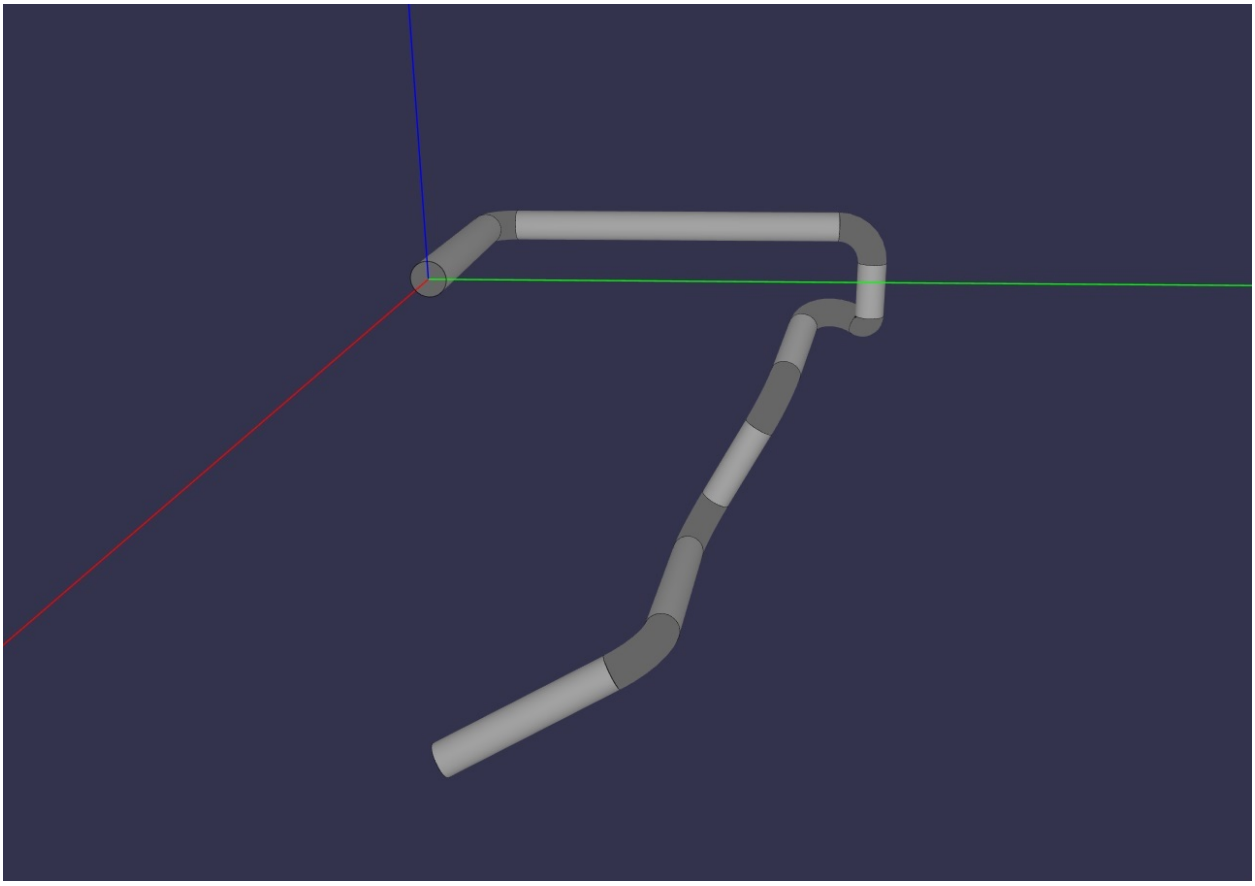


These contours are extruded to form the side shell. A bottom is formed from an extrusion.

Screw holes are placed at equal curvilign distance of the belt.

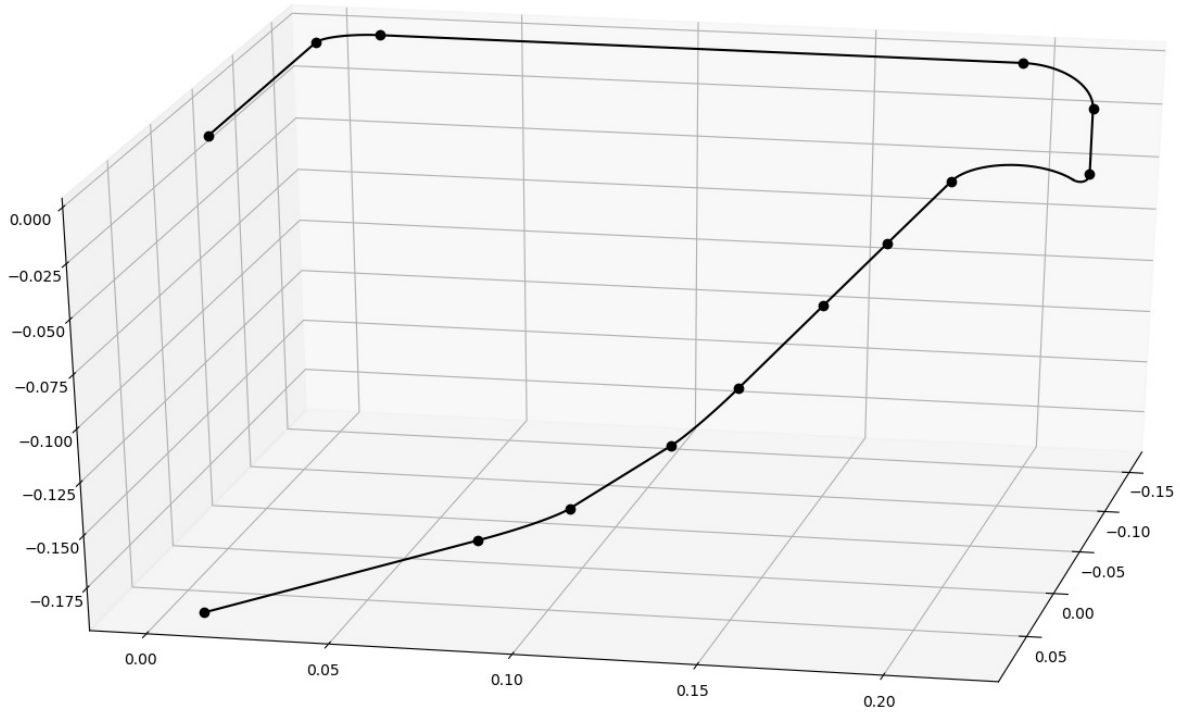
All the primitives are fused together in a single volume.

7.2 Sweep



https://github.com/Dessia-tech/volmdlr/blob/master/scripts/sweep.py

A Sweep is pipes, created with Circle2D/Arc2D which is contained in a Contour2D. You have to create the neutral fiber, i.e., the pipe's road, with the primitive RoundedLineSegment3D.

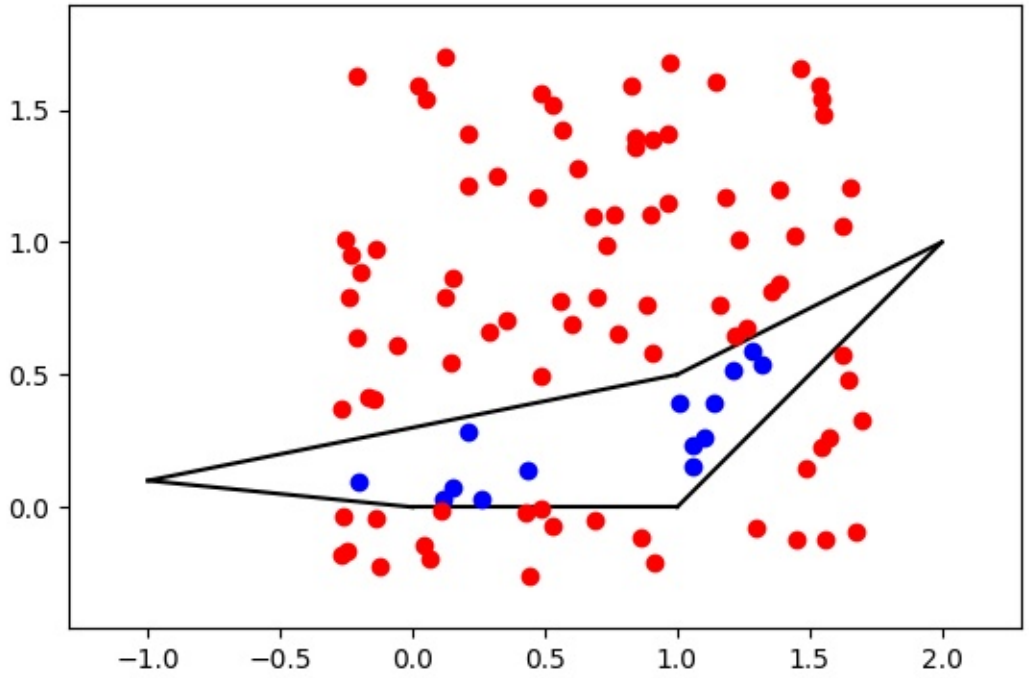


7.3 Polygon

https://github.com/Dessia-tech/volmdlr/blob/master/scripts/polygon2D.py

A polygon is defined out of points. Random points are sampled and the tested whether they are inside or outside of the polygon. They are plotted with the Matplotlib binding MPLPlot with custom style:

- red if they are outside,
- blue if they are inside


1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index

B

Basis2D (*class in volmdlr*), 7
Basis3D (*class in volmdlr*), 9

D

deterministic_unit_normal_vector (*volmdlr.Vector3D attribute*), 9

F

Frame2D (*class in volmdlr*), 7
Frame3D (*class in volmdlr*), 10
frame_mapping (*volmdlr.Vector2D attribute*), 7
frame_mapping (*volmdlr.Vector3D attribute*), 9
from_two_vectors() (*volmdlr.Basis3D class method*), 10

N

new_coordinates (*volmdlr.Frame3D attribute*), 10
norm (*volmdlr.Vector2D attribute*), 7
normalize (*volmdlr.Basis2D attribute*), 7
normalize (*volmdlr.Basis3D attribute*), 10
normalize (*volmdlr.Vector2D attribute*), 7
normalize (*volmdlr.Vector3D attribute*), 9

O

old_coordinates (*volmdlr.Frame3D attribute*), 10

P

Point2D (*class in volmdlr*), 7
Point3D (*class in volmdlr*), 9

R

random_unit_normal_vector (*volmdlr.Vector3D attribute*), 9
rotation (*volmdlr.Frame3D attribute*), 10
rotation (*volmdlr.Vector3D attribute*), 9

T

translation (*volmdlr.Vector2D attribute*), 7

V

Vector2D (*class in volmdlr*), 7
Vector3D (*class in volmdlr*), 9

X

x_rotation (*volmdlr.Vector3D attribute*), 9

Y

y_rotation (*volmdlr.Vector3D attribute*), 9

Z

z_rotation (*volmdlr.Vector3D attribute*), 9