



The YDEOS Project

Yacht Design and Engineering Open Source
v. 2020

Project info

September 22nd, 2020

Copyright © 2020-2021 The YDEOS project
Authors: Guillaume Florent (2020)

This work is licensed under a
Creative Commons Attribution-NonCommercial-NoDerivs 3.0Unported License.

Contents

- 1. Introduction.....4
 - 1.1. Vision.....4
 - 1.2. Scope.....4
 - 1.3. Governance.....4
- 2. Architecture.....5
 - 2.1. Technology.....5
 - 2.2. Decisions.....5
 - 2.2.1. Functional Core, Imperative Shell.....5
 - 2.2.2. Pipes and filters.....5
 - 2.2.3. Side effects.....5
 - 2.2.4. Dependencies.....5
 - 2.3. Delivery.....5
- A. Project checklist.....6
 - A.1. Structure.....6
 - A.2. Code.....6
 - A.3. Testing.....6
 - A.4. CI.....6

Conventions:

Some `code` or `symbol` in a sentence

library name

code

Column header #1	Column header #2	Column header #3	Column header #4
Table content			

<http://www.anylink.com>

1. Introduction

1.1. Vision

The YDEOS project is a collection of software libraries and executables that help with yacht and ship design and engineering.

1.2. Scope

The following topics are intended to be covered:

- units conversions
- forces modeling
- hydrodynamics and resistance estimates using textbook and publication formulae and regressions
- aerodynamic sails and windage models
- upright and heeled/trimmed hydrostatics
- 2D foil sections generation
- sailing yacht velocity prediction
- sailing yacht polars handling and utilities
- offsets from CAD
- 2D structured meshing
- ship and yacht related CFD cases generators
- automated sail shape analysis

1.3. Governance

The governance model for 2020 / 2021 is a BDFL model
(https://en.wikipedia.org/wiki/Benevolent_dictator_for_life)

2. Architecture

2.1. Technology

The initial developments of the YDEOS project will use Python 3 and its scientific stack (Numpy, Scipy, etc ...). The scientific ecosystem + ease of development combination of Python is unrivaled; yet, some parts of the project may later have to be re-written using other technologies for performance reasons.

2.2. Decisions

2.2.1. Functional Core, Imperative Shell

Functional Core, Imperative Shell: development will follow this philosophy with domains modeled as functionally as possible.

Shell (Command Line Interface, Graphical User Interface, web entry points) may be developed in imperative style, with an optional Service Layer if orchestration logic starts to creep into domain logic or into the shell developments.

2.2.2. Pipes and filters

Keep in mind that, to make the YDEOS project globally more useful, the output of executables should be designed as the input of others. Create common formats between projects whenever it makes sense.

2.2.3. Side effects

Side effects coding should be separated into 'what' and 'how' parts.

2.2.4. Dependencies

Each project should be as self-contained as possible. Whenever possible, there should be no dependency on any other internal project.

Depending on the Python scientific stack is OK. Depending on PythonOCC for geometry stuff is OK.

2.3. Delivery

Implement CI/CD principles to create:

- an install script for Linux and Windows for the whole YDEOS project software
- a Docker image that contains the whole YDEOS project software

CI should be handled at each internal project level and at the YDEOS project level (end-to-end scenarii)

CD will be handled at the YDEOS project level.

A. Project checklist

A.1. Structure

/<project>

.gitignore

requirements.txt

LICENSE

README.rst

setup.py

/bin

/docs

/tests

/<same name as project>

__init__.py (with text art and version, author, license info)

A.2. Code

Type hints on all functions

PEP8 with a 120 characters line length limit

Numpy style docstrings

A.3. Testing

Unit tests using pytest

A.4. CI

Travis CI

Code quality