# User guide for using the python framework to generate models in FreeCAD

A. F. D. Morgan

November 14, 2017

# Contents

# Chapter 1

# Introduction

This framework is designed to make the programatic control of FreeCAD easier with the aim of using the STL files generated as an input to EM modelling software. Geometries are defined in python and can be parameterised, allowing for parameter sweeps. The parameters used to generate a particular model are also stored which aids later analysis.

One of the advantages of this approach is that the models can be visualised and errors quickly identified *before* being passed on to EM simulation thus saving time overall.

# Chapter 2

# Using an existing file

Use the python interpreter built into FreeCAD.

```
python pillbox_cavity.py C:\temp
```

will run the code which generates the pillbox cavity models as defined in `pillbox_cavity.py` and places the results in `C:\temp\pillbox_cavity`. in this folder will be a sub folder containing the base model, also sub folders containing additional models which form part of user defined parameter sweeps (also defined in `pillbox_cavity.py`) Inside each of these sub folders is a native FreeCAD file containing the model. Also a text file describing the parameter settings used to generate this model. There is also a folder caller binary and one named ascii. These contain STL files of the various components defined in `pillbox_cavity.py`. Usually a separate component is defined if it will be a different material in the model.

# Chapter 3

# Creating a new file

This is an example of a simple input file (`pillbox_cavity.py`). All model files share the same basic structure however things like the imports, input parameters, and parts directory will likely need adjustment in addition to the model definition section itself.

```python
from freecad_elements import make_beampipe, make_circular_aperture,
                             ModelException, parameter_sweep, base_model
from sys import argv
import os

# baseline model parameters
INPUT_PARAMETERS = {'cavity_radius': 20, 'cavity_length': 20,
                    'pipe_radius': 10, 'pipe_length': 80}

# The model name is the name of the file.
# The output path is a user defined location.
MODEL_NAME, OUTPUT_PATH = argv

def pillbox_cavity_model(input_parameters):
    """ Generates the geometry for the pillbox cavity in FreeCAD.
        Also writes out the geometry as STL files and writes a
        "sidecar" text file containing the input parameters used.

         Args:
            input_parameters (dict): Dictionary of input parameter names and values
        """

    try:
        # The model is defined in this section. Different components can be defined
        # and later added to the parts dictionary.
        wire1, face1 = make_circular_aperture(input_parameters['pipe_radius'])
        wire2, face2 = make_circular_aperture(input_parameters['cavity_radius'])
        beampipe1 = make_beampipe(face1, input_parameters['pipe_length'],
                                  (-input_parameters['pipe_length'] / 2.
                                   - input_parameters['cavity_length'] / 2., 0, 0)
                                  )
        beampipe3 = make_beampipe(face1, input_parameters['pipe_length'],
                                  (input_parameters['pipe_length'] / 2.
                                   + input_parameters['cavity_length'] / 2., 0, 0)
                                  )
        beampipe2 = make_beampipe(face2, input_parameters['cavity_length'])
        fin1 = beampipe1.fuse(beampipe2)
        fin2 = fin1.fuse(beampipe3)
    except Exception as e:
        # This allows errors in the model to be separated from other code errors.
        raise ModelException(e)

    # An entry in the parts dictionary corresponds to an STL file.
    # This is useful for parts of differing materials.
    parts = {'all': fin2}
```

```
46      return parts, os.path.splitext(os.path.basename(MODEL_NAME))[0]
47
48  # Generate the base model.
49  base_model(pillbox_cavity_model, INPUT_PARAMETERS, OUTPUT_PATH, accuracy=10)
50  # Generate additional models to form a parameter sweep.
51  parameter_sweep(pillbox_cavity_model, INPUT_PARAMETERS, OUTPUT_PATH,
52                  'cavity_radius', [10, 30, 40, 50])
```

# Chapter 4

# Helper functions

In `freecad_elements.py` various helper functions are defined. These are mainly to do with creating beam pipes with various shapes, and tapers between various apertures. Their basic use can be seen in the example code in chapter 3.