TXLWriter Documentation

Release 1.0.0

Esteban Marin

May 17, 2016

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

This document describes the usage and technical reference of the python program *TXLWizard* written by Esteban Marin (estebanmarin@gmx.ch).

1.1 What does it do?

The *TXLWizard* provides routines for generating TXL files (.txl) for the preparation of E-Beam lithography masks using python code. The TXL files can be processed with BEAMER. See the following links:

- http://genisys-gmbh.com/web/products/beamer.html
- http://cad035.psi.ch/LB_index.html
- http://cad035.psi.ch/LBDoc/BEAMER_Manual.pdf

The generated TXL files are also converted to HTML / SVG for presentation in any modern browser or vector graphics application.

Moreover, a command line interface *TXLConverter* provides conversion of existing TXL files to HTML / SVG (See Section *TXLConverter*).

1.2 Technical Information

The "TXLWizard" is written in python and will run in Python version 2.7+ and 3.1+.

In order to use it, the *TXLWizard* package must be available as
a python package, i.e. either it must be copied to

```
Path_to_my_python_installation/site-packages/
```

or to the path where your script is located.

Alternatively, you can also prepend the following command to your python script:

```
sys.path.append('path to the folder containing TXLWizard')
```

1.3 Example SVG Output

An example output can be seen here:

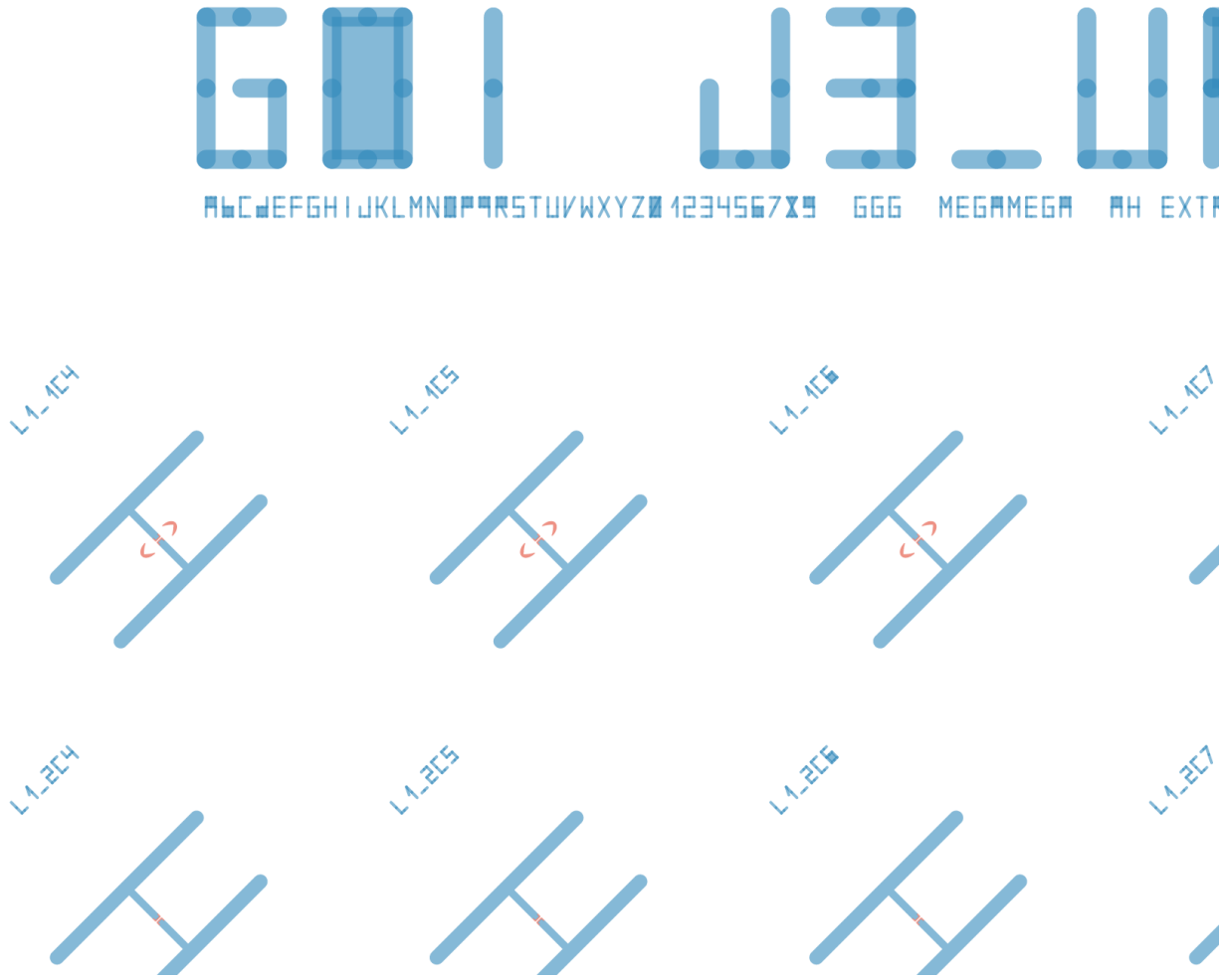

Fig. 1.1: Example SVG output for a mask

CHAPTER

TWO

TXLWIZARD EXAMPLE

The following code demonstrates a simple example usage of the *TXLWizard* for generating TXL files with python code.

The code can be found in the file `Content/Example_Simple.py`.

The resulting image is shown in Figure *Simple Example: Generated Mask*.

A more advanced example is shown in Section *TXLWizard Advanced Example*

2.1 Code

```
####################
# Import Libraries #
####################

# Import TXLWriter, the main class for generating TXL Output
import TXLWizard.TXLWriter

# Import Pre-Defined Shapes / Structures wrapped in functions
import TXLWizard.ShapeLibrary.EndpointDetectionWindows
import TXLWizard.ShapeLibrary.Label

# Import math module for calculations
import math


################################
# Sample / Structure Parameters #
################################

# Define all sample parameters
SampleParameters = {
    'Width': 8e3,
    'Height': 8e3,
    'Label': 'Simple Demo',
}

# Define all structure parameters
StructureParameters = {
    'Circle': {
        'Radius': 50,
        'Layer': 3
    },
```

```
    'CircleArray': {
        'Columns': 6,
        'Rows': 5,
        'ArrayXOffset': 500,
        'ArrayYOffset': -500,
        'ArrayOrigin': [0.75e3, 3e3],
        'Label': 'R{:d}C{:d}',
    }
}


########################
# Initialize TXLWriter #
########################
TXLWriter = TXLWizard.TXLWriter.TXLWriter(
    GridWidth=SampleParameters['Width'],
    GridHeight=SampleParameters['Height']
)

####################
# Define Structures #
####################

## Sample Label ##

# Give the sample a nice label
SampleLabelObject = TXLWizard.ShapeLibrary.Label.GetLabel(
    TXLWriter,
    SampleParameters['Label'],
    OriginPoint=[
        0.5e3, 1. * SampleParameters['Height'] / 2. - 500
    ],
    FontSize=150,
    StrokeWidth=20,
    RoundCaps=True, # Set to False to improve e-Beam performance
    Layer=1
)


## Endpoint Detection ##

# Use Pre-Defined Endpoint Detection Windows
TXLWizard.ShapeLibrary.EndpointDetectionWindows.GetEndpointDetectionWindows(
    TXLWriter, Layer=1)

## User Structure: Circle ##

# Create Definition Structure for Circle that will be reused
CircleStructure = TXLWriter.AddDefinitionStructure('Circle')
CircleStructure.AddPattern('Circle',
    Center=[0, 0],
    Radius=StructureParameters['Circle']['Radius'],
    Layer=StructureParameters['Circle']['Layer']

)

# Create array of the definition structure above
CircleArray = TXLWriter.AddContentStructure('CircleArray')
```

```
CircleArray.AddPattern('Array',
    ReferencedStructureID=CircleStructure.ID,
    OriginPoint=StructureParameters['CircleArray']['ArrayOrigin'],
    PositionDelta1=[
        StructureParameters['CircleArray']['ArrayXOffset'], 0
    ],
    PositionDelta2=[
        0, StructureParameters['CircleArray']['ArrayYOffset']
    ],
    Repetitions1=StructureParameters['CircleArray']['Columns'],
    Repetitions2=StructureParameters['CircleArray']['Rows']
)



#########################
# Generate Output Files #
#########################

# Note: The suffix (.txl, .html, .svg) will be appended automatically
TXLWriter.GenerateFiles('Masks/Example_Simple')

```

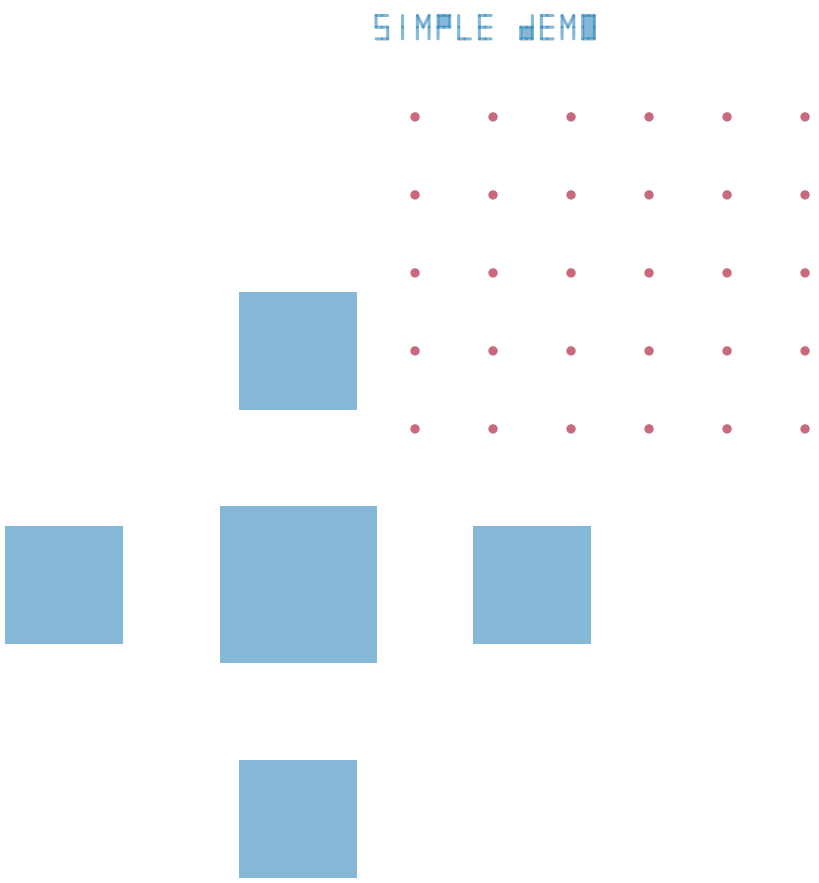

Fig. 2.1: Simple Example: Generated Mask

CHAPTER

THREE

TXLCONVERTER

blub

CHAPTER

FOUR

PYTHON MODULE REFERENCE

Module *TXLWriter* contains the *TXLWizard.TXLWriter.TXLWriter* class

class TXLWizard.TXLWriter.**TXLWriter**(***kwargs*)

Controller class for generating TXL / SVG / HTML output.

Here we can add structures (definitions and content) which will be rendered in the output. Optionally a coordinate system grid is drawn.

Parameters

- **ShowGrid** (*bool, optional*) – Show the coordinate system grid or not.

  Defaults to True

- **GridWidth** (*int, optional*) – Full width of the coordinate system grid in um.

  Defaults to 800

- **GridHeight** (*int, optional*) – Full height of the coordinate system grid in um.

  Defaults to 800

- **GridSpacing** (*int, optional*) – Coordinate Sytem Grid Spacing in um.

  Defaults to 100

- **SubGridSpacing** (*int, optional*) – Coordinate System Sub-Grid Spacing in um.

  Defaults to 10

AddContentStructure(*Index*, ***kwargs*)

Add content structure. A content structure can hold patterns that will render in the output.

A structure corresponds to the "STRUCT" command in the TXL file format.

Parameters

- **Index** (*str*) – Unique identification of the structure. Must be used when referencing to this structure.
- **kwargs** (*dict*) – keyword arguments passed to the structure constructor

Returns

Return type TXLWizard.Patterns.Structure.Structure structure instance

AddDefinitionStructure(*Index*, ***kwargs*)

Add definition structure. A definition structure can be referenced by a content structure.

A structure corresponds to the "STRUCT" command in the TXL file format.

Parameters

- **Index** (*str*) – Unique identification of the structure. Must be used when referencing to this structure.
- **kwargs** (*dict*) – keyword arguments passed to the structure constructor

Returns

Return type TXLWizard.Patterns.Structure.Structure structure instance

AddHelperStructure(*Index*, ***kwargs*)

Add helper structure. Helper structures are only visible in the HTML / SVG Output.

A structure corresponds to the "STRUCT" command in the TXL file format.

Parameters

- **Index** (*str*) – Unique identification of the structure. Must be used when referencing to this structure.
- **kwargs** (*dict*) – keyword arguments passed to the structure constructor

Returns

Return type TXLWizard.Patterns.Structure.Structure structure instance

GenerateFiles(*Filename*, *TXL=True*, *SVG=True*, *HTML=True*)

Generate the output files (.txl, .svg, .html).

Parameters

- **Filename** (*str*) – Path / Filename without extension. The corresponding path will be created if it does not exist
- **TXL** (*Optional[bool]*) – Enable TXL Output
- **SVG** (*Optional[bool]*) – Enable SVG Output
- **HTML** (*Optional[bool]*) – Enable HTML Output

CHAPTER

FIVE

TXLWIZARD ADVANCED EXAMPLE

The following code demonstrates an advanced example usage of the *TXLWizard* for generating TXL files with python code.

The code can be found in the file `Content/Example_Advanced.py`.

The resulting image is shown in Figure *Simple Example: Generated Mask*.

5.1 Code

```
####################
# Import Libraries #
####################

# Import TXLWriter, the main class for generating TXL Output
import TXLWizard.TXLWriter

# Import Pre-Defined Shapes / Structures wrapped in functions
import TXLWizard.ShapeLibrary.EndpointDetectionWindows
import TXLWizard.ShapeLibrary.Markers
import TXLWizard.ShapeLibrary.Label
import TXLWizard.ShapeLibrary.CornerCube

# Import math module for calculations
import math


################################
# Sample / Structure Parameters #
################################

# Define all sample parameters
SampleParameters = {
    'Width': 8e3,
    'Height': 8e3,
    'Label': 'GOI Demo CornerCube',
}

# Define all structure parameters
StructureParameters = {
    'CornerCube': {
        'BridgeLength':8,
        'ParabolaFocus': 9,
        'XCutoff': 9,
```

```
        'AirGapX': 3,
        'AirGapY': 1,
        'LabelXOffset': 0,
        'LabelYOffset': 50,
        'Label': 'R{:d}C{:d}', # {:d} will be replaced
                               # by str.format() with the corresponding row / column
        'Layer': 2
    },
    'Circle': {
        'Radius': 5,
        'Layer': 3
    },
    'CornerCubeArray': {
        'Columns': 6,
        'Rows': 5,
        'ArrayXOffset': 500,
        'ArrayYOffset': -500,
        'ArrayOrigin': [0.75e3, 3e3]
    }
}


########################
# Initialize TXLWriter #
########################
TXLWriter = TXLWizard.TXLWriter.TXLWriter(
    GridWidth=SampleParameters['Width'],
    GridHeight=SampleParameters['Height']
)

#####################
# Define Structures #
#####################

## Sample Label ##

# Give the sample a nice label...
SampleLabelObject = TXLWizard.ShapeLibrary.Label.GetLabel(
    TXLWriter,
    SampleParameters['Label'],
    OriginPoint=[
        0.5e3, 1. * SampleParameters['Height'] / 2. - 500
    ],
    FontSize=150,
    StrokeWidth=20,
    RoundCaps=True,# Set to False to improve e-Beam performance
    Layer=1
)
# ...and some other information
Alphabet = TXLWizard.ShapeLibrary.Label.GetLabel(
    TXLWriter,
    'abcdefghijklmnopqrstuvwxyz0123456789 megamega ggg ah extraaaa rischaaaar',
    OriginPoint=[
        0.5e3, 1. * SampleParameters['Height'] / 2. - 600
    ],
    FontSize=50,
    StrokeWidth=3,
    RoundCaps=True, # Set to False to improve e-Beam performance
```

```
    Layer=1
)

## Endpoint Detection ##

# Use Pre-Defined Endpoint Detection Windows
TXLWizard.ShapeLibrary.EndpointDetectionWindows.GetEndpointDetectionWindows(
    TXLWriter, Layer=1)

## Alignment Markers ##

# Use Pre-Defined Alignment Markers
TXLWizard.ShapeLibrary.Markers.GetMarkers(
    TXLWriter, Layer=1)


## User Structure: Corner Cube ##

# Create Definition Structure for Corner Cube that will be reused
CornerCubeDefinition = TXLWizard.ShapeLibrary.CornerCube.GetCornerCube(
    TXLWriter,
    ParabolaFocus=StructureParameters['CornerCube']['ParabolaFocus'],
    XCutoff=StructureParameters['CornerCube']['XCutoff'],
    AirGapX=StructureParameters['CornerCube']['AirGapX'],
    AirGapY=StructureParameters['CornerCube']['AirGapY'],
    Layer=StructureParameters['CornerCube']['Layer']
)

# Create Definition Structure for combination of cornercube and additional circle
FullCornerCubeNoRotation = TXLWriter.AddDefinitionStructure('FullCornerCubeNoRotation')
FullCornerCubeNoRotation.AddPattern('Reference',
    ReferencedStructureID=CornerCubeDefinition.ID,
    OriginPoint=[1. * StructureParameters['CornerCube']['BridgeLength'] / 2., 0]
)
FullCornerCubeNoRotation.AddPattern('Circle',
    Center=[0, 0],
    Radius=StructureParameters['Circle']['Radius'],
    Layer=StructureParameters['Circle']['Layer']
)

# Create definition structure with rotation of entire referenced structure
FullCornerCube = TXLWriter.AddDefinitionStructure('FullCornerCube',
                                                  RotationAngle=45)
FullCornerCube.AddPattern('Reference',
    ReferencedStructureID=FullCornerCubeNoRotation.ID,
    OriginPoint=[0, 0]
)

# Create array of the definition structure above
CornerCubeArrayFine = TXLWriter.AddContentStructure('CornerCubeArrayFine')
CornerCubeArrayFine.AddPattern('Array',
    ReferencedStructureID=FullCornerCube.ID,
    OriginPoint=StructureParameters['CornerCubeArray']['ArrayOrigin'],
    PositionDelta1=[
        StructureParameters['CornerCubeArray']['ArrayXOffset'], 0
    ],
    PositionDelta2=[
        0, StructureParameters['CornerCubeArray']['ArrayYOffset']
```

```
    ],
    Repetitions1=StructureParameters['CornerCubeArray']['Columns'],
    Repetitions2=StructureParameters['CornerCubeArray']['Rows']
)


# Add Labels to each array element
for Row in range(1, StructureParameters['CornerCubeArray']['Rows'] + 1):
    for Column in range(1, StructureParameters['CornerCubeArray']['Columns'] + 1):
        RowColumnCountLabel = TXLWizard.ShapeLibrary.Label.GetLabel(
            TXLWriter,
            StructureParameters['CornerCube']['Label'].format(Row, Column),
            OriginPoint=[
                StructureParameters['CornerCubeArray']['ArrayOrigin'][0]
                + StructureParameters['CornerCubeArray']['ArrayXOffset']
                * (Column - 1) + StructureParameters['CornerCube']['LabelXOffset'],
                StructureParameters['CornerCubeArray']['ArrayOrigin'][1]
                + StructureParameters['CornerCubeArray']['ArrayYOffset']
                * (Row - 1) + StructureParameters['CornerCube']['LabelYOffset']],
            FontSize=16,
            StrokeWidth=3,
            RoundCaps=True,# Set to False to improve e-Beam performance
            Layer=1,
            RotationAngle=45
        )


#########################
# Generate Output Files #
#########################

# Note: The suffix (.txl, .html, .svg) will be appended automatically
TXLWriter.GenerateFiles('Masks/Example_Advanced')

```

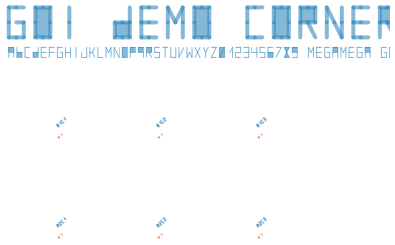

Fig. 5.1: Simple Example: Generated Mask

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

t