UNIVERSITÉ DE BORDEAUX

3D Printed Lithophanes

Software Engineering Project

Supervisor:

Prof. Pascal Desbarats

Prof. Fabien Baldacci

Students:

Tran Thanh Huy

Phan Thanh Dat

Dang Thanh Minh

September - 2020

Summary

The first that we want to say thank you to Professor Pascal Desbarats and Professor Fabien Baldacci that helping us during the project. This is the first report that we will introduce about existing source code that Professor Pascal gave us. After that is requirements analysis and objective, architecture and tests.

We have 3 members in our group:

- Tran Thanh Huy - huytran190194@gmail.com
- Phan Thanh Dat - phanthanhdat203@gmail.com
- Dang Thanh Minh - minhdangthanh1982@gmail.com

The goal of this project is implement an application for transforming a digital photograph into a lithophane 3D model that can be printed:

1. Processing of the input image
2. Creation of a 3D map by adding a third dimension (the graylevel of each pixel)
3. Mesh construction by joining the pixel's centers
4. Adding a shell to make the model 3D printable
5. Adding a border if asked
6. Export to STL file

Declaration

We hereby declare that this work is product of our own work. The contents of this report are original except where specific reference is made to the work of others. Besides, this report has not been submitted in whole or in part for any other university. We also confirm that all opinions, results, conclusions and recommendations are our own and may not represent the policies of opinions of University of Bordeaux.

Tran Thanh Huy
Phan Thanh Dat
Dang Thanh Minh

Krishna Kumar
September 2020

Acknowledgements

We would like to acknowledge Professor Pascal Desbarats and Professor Fabien Baldacci for taking time to support our problems during the project time and your useful instructions.

Abstract

Lithophane is an etched or molded artwork in very thin translucent porcelain that can be seen clearly only when back lit with a light source. It is a design or scene in intaglio that appears grey tone. A lithophane is a three-dimensional image with different thicknesses that cause the opacity of each of its parts to vary by projecting light on it, the thinner parts are brighter and the thicker parts are darker. The varying light source is what makes lithophane more interesting to the viewer than two-dimensional pictures.

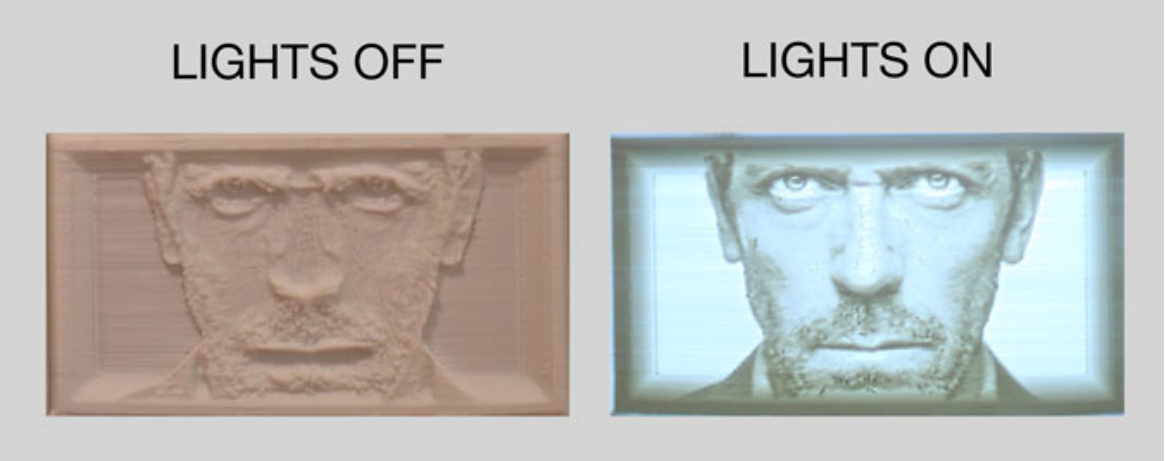

Fig. 1 A sample of lithophane

In the past, to create a lithophane, artisans would carve into a wax with small tools. Next they would create a plaster mold of the lithophane. Then from the mold a porcelain slip would be added to create a positive of the image. Finally once the porcelain has dried it is put into a kiln and fired at 2000C. But nowadays, with the common of 3D printers, it allows us to create a lithophane in a much simplier way. What we need to do is only converting a 2-dimensional digital image to a 3-dimensional design which the height illustrate the thicknesses and then let the 3D printer generate the model following that design.

The goal of this project is to implement an application for transforming a 2-dimensional digital photograph into a lithophane 3D model that can be read and printed by a 3D printer. By that we will learn about the software development process:

1. Planning
2. Analysis
3. Design
4. Implementation
5. Testing and Integration
6. Maintenance

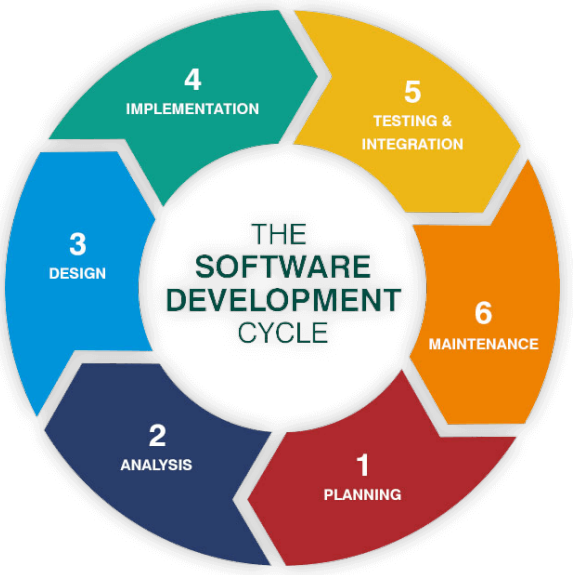

Fig. 2 Software Development Cycle

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Overview

Nowadays, Lithophane is more and more popular in the world. People look for it as a wonderful gift for children or a more exclusive alternative way to store their memory than an image. With the supporting of 3D printers, creating a lithophane becomes much simpler than ever. People may just need to search on the internet for a online image-to-lithophane converter, select their desired image and let the converter do their job to generate a 3D map for 3D printers to create a physical lithophane.
That is how the world is going at the moment, but for our project, we will study on how to convert a 2D image into a map for 3D printer. In generally, the process for that is divided into 6 steps as below:

1. Converting the 2D input image to a gray scale 2D image
2. Creation of a 3D map by adding a third dimension (the gray level of each pixel)
3. Mesh construction by joining the pixel's centers
4. Adding a shell to make the model 3D printable
5. Adding a border if asked
6. Export to STL file

1.2 Existing program analysis

1.2.1 3dp.rocks

Professor Pascal gives a similar web-based program at http://3dp.rocks/lithophane/. After investigating this page we find out some main features:

- Upload image.
- Rendering model.
- Other preferences.

1.2.1.1 Libraries

This project using many libraries as bellow:

- three.js[1] - JavaScript 3D library. The aim of the project is to create an easy to use, lightweight, 3D library with a default WebGL renderer. The library also provides Canvas 2D, SVG and CSS3D renderers in the examples.
- FileSaver.js[6] is the solution to saving files on the client-side, and is perfect for web apps that generates files on the client, However if the file is coming from the server we recommend you to first try to use Content-Disposition attachment response header as it has more cross-browser compatiblity.
- Binary STL[7] example from Paul Kaplan

1.2.1.2 Upload image

At the first view, this feature allow user to upload single or multiple images to their system. If images are uploaded successful, we will be redirect to Model page.

1.2.1.3 Rendering model

This page allows us to select 1 of 8 predefined shape[5]:

- Flat: A rectangular model with the image impressed in one face. Curve setting is ignored

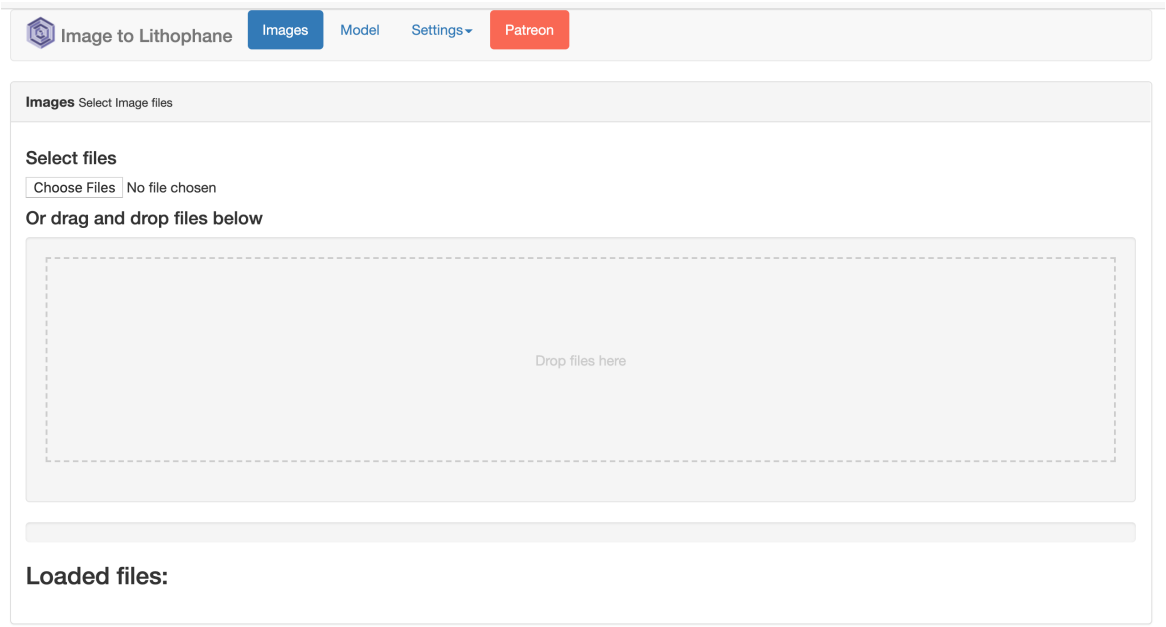

Fig. 1.1 Upload image page

- Inner Curve: A curved rectangular model with the image impressed on the inner face. Use curve setting to set the number of degrees in the arc.
- Outer Curve: A curved rectangular model with the image impressed on the outer face. Use curve setting to set the number of degrees in the arc.
- Solid Cylinder: A cylindrical model with the image impressed on the curved face.
- Rectangular Pillow: A rectangular pillow model with the image impressed on the top face
- Dome - Image on Top: A circular dome model with the image impressed as if from above the dome
- Dome - Image on Side: A circular dome model with the image impressed as if from the edge of the dome with the image wrapping the perimeter of the dome and the top of the image meeting in the centre
- Heart: A heart shaped model with the image impressed on the curved faces

They also provide Download STL feature.

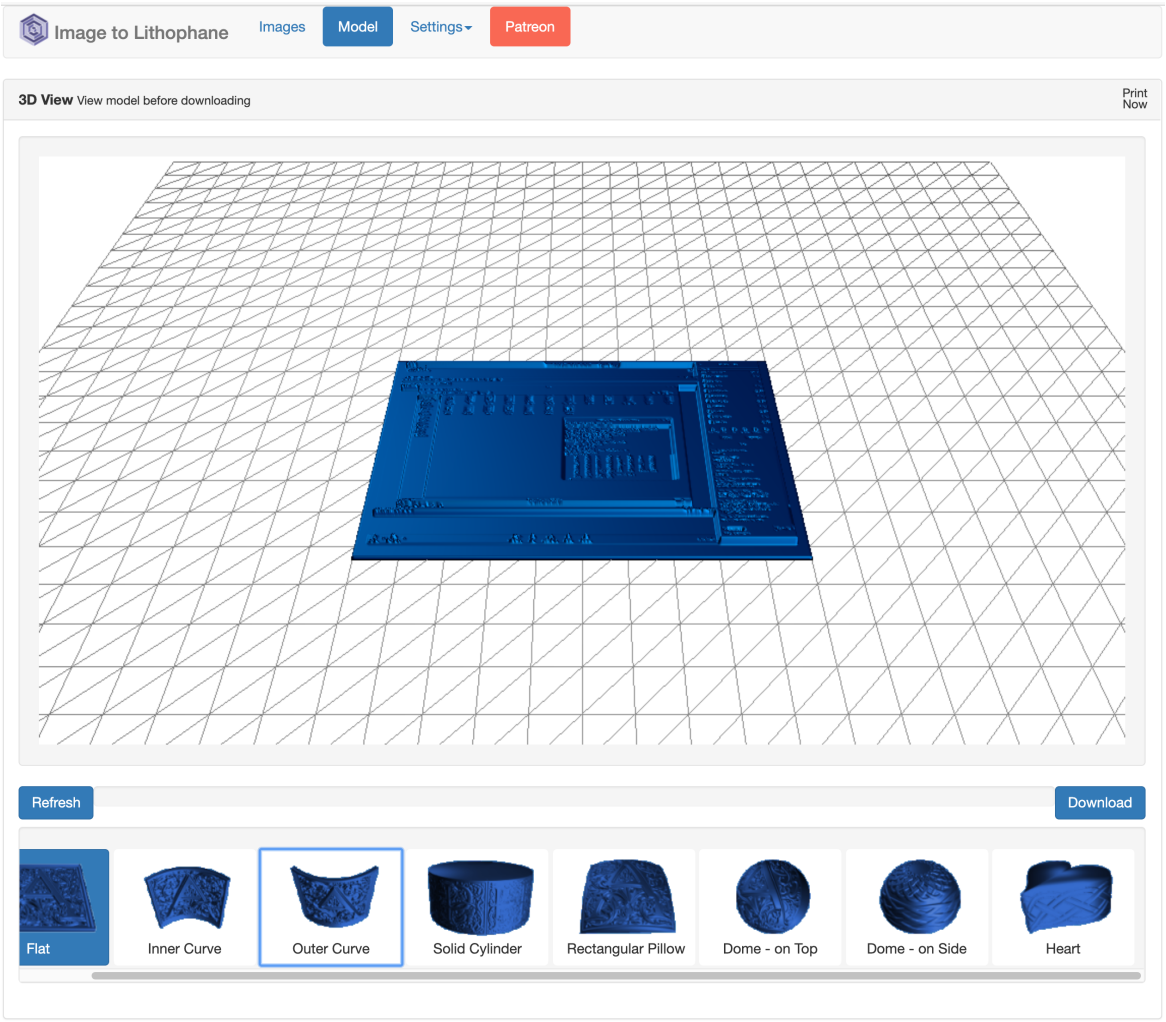

Fig. 1.2 Rendering model page

1.2.1.4 Other preferences

Beside 2 core features, program also allows us to customize many other settings[5], such as model, image, download and cache settings:

- Model settings.
  - Max Size: The largest X or Y dimension of the output lithophane - X if original image wide, Y if high.
  - Thickness: The maximum Z dimension of the output lithophane.
  - Border: The thickness of the border around the edge.
  - Thinnest layer: This is the minimum layer thickness (for the brightest pixels in the image).

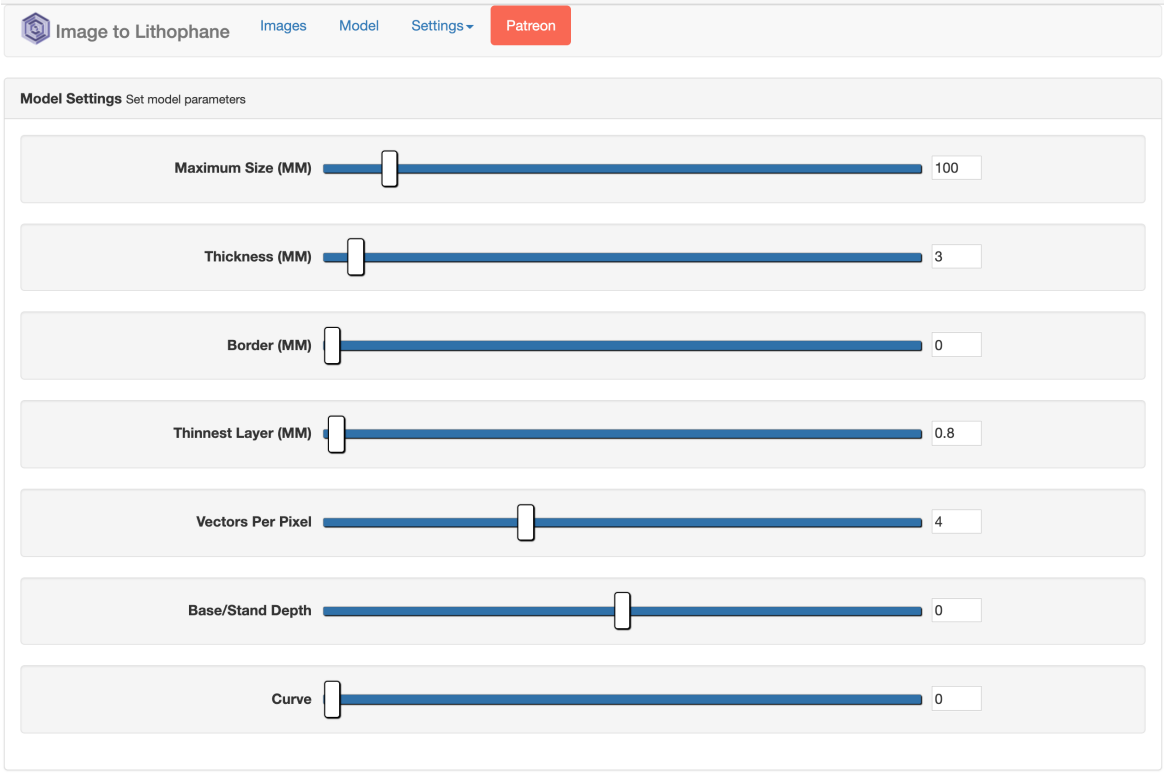

Fig. 1.3 Model settings

  - Vectors per pixel: Each of the pixels in the image is translated into a number of 3D points on the surface of the lithophane, the larger this number, the more detailed the output (and the larger the STL file/slower the processing) 2 is a good value for this you can go up to 5, but it will take time and use memory.
  - Base/Stand depth: A small stand on the base for when printing vertically. Positive number for base coming forward, negative for backward.
  - Curve: Number of degrees to curve the surface (for the curved forms).

- Image settings.
  - Positive Image: Set to Positive Image, lighter areas of the original image will be thinner in the output and the thicker if set to Negative Image.
  - Mirror Image: Set to Mirror Image, the image will be mirrored about the X axis (for rear viewing), otherwise the output will be in the same orientation as the original image.
  - Flip Image: Set to File Image, the image will be flipped about the Y axis, otherwise the output will be in the same orientation as the original image.

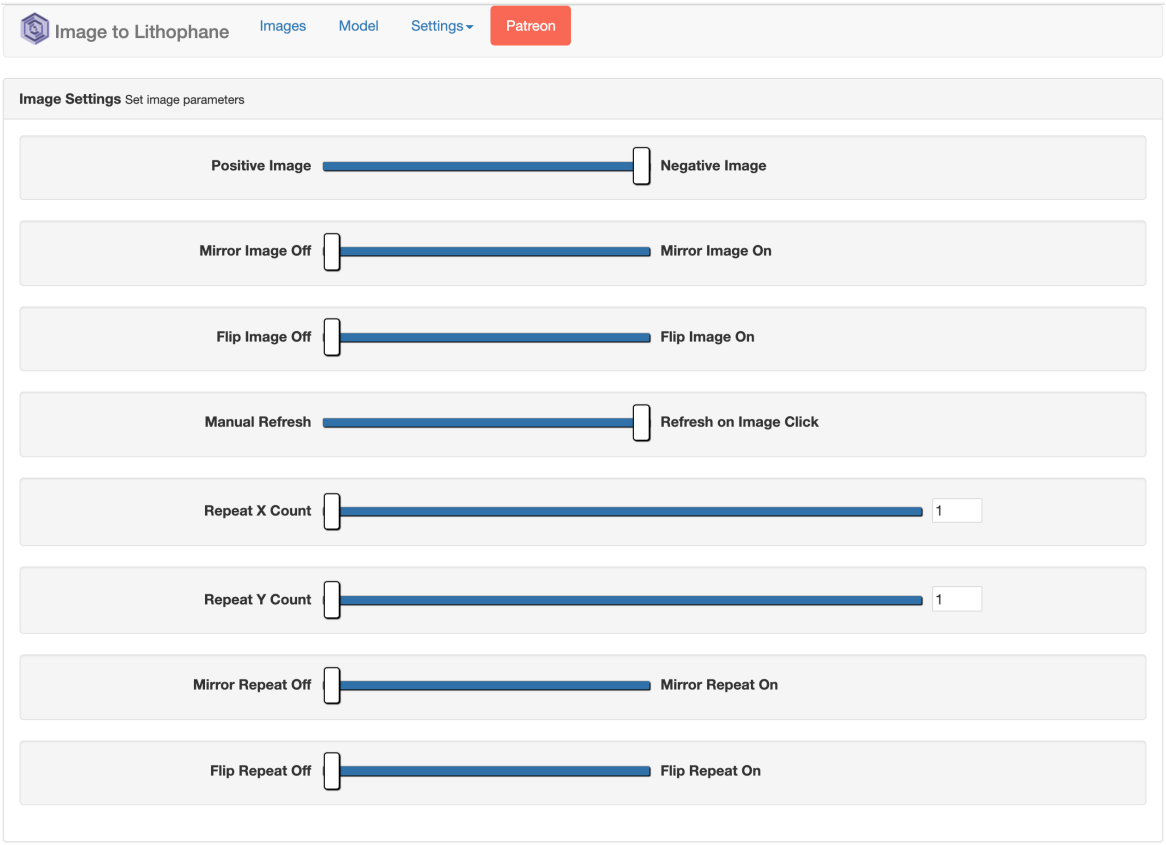

Fig. 1.4 Image settings

  - Manual Refresh: Set to Manual Refresh to only recreate the model when the 'Refresh' button is pressed, set to Automatic if you'd like the model to refresh when you select the image
  - Repeat X Count: Set to repeat the image in the X direction prior to creating the output.
  - Repeat Y Count: Set to repeat the image in the Y direction prior to creating the output.
  - Mirror Repeat: Set to alternately mirror the image when the X Repeat setting is being used.
  - Flip Repeat: Set to alternately flip the image when the Y Repeat setting is being used.

- Download settings
  - Binary STL: Set to use a binary (smaller and faster) STL file format, otherwise use ASCII where needed for compatibility with other software.

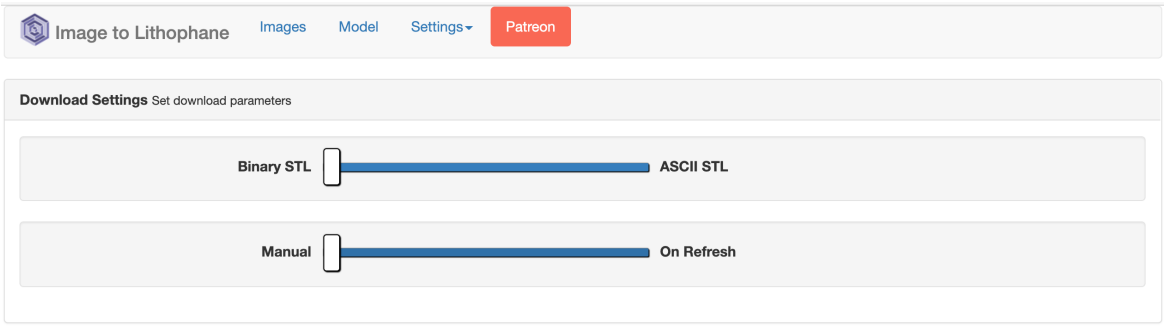

Fig. 1.5 Download settings

  - Manual Download: If set to Automatic, the STL file will download each time you refresh the model, if set to Manual, the download will only happen when you press the download button.

- Cache settings: Set to save your configuration in the browser LocalStorage each time it is changed, this way, your last state will be loaded when you restart the application.

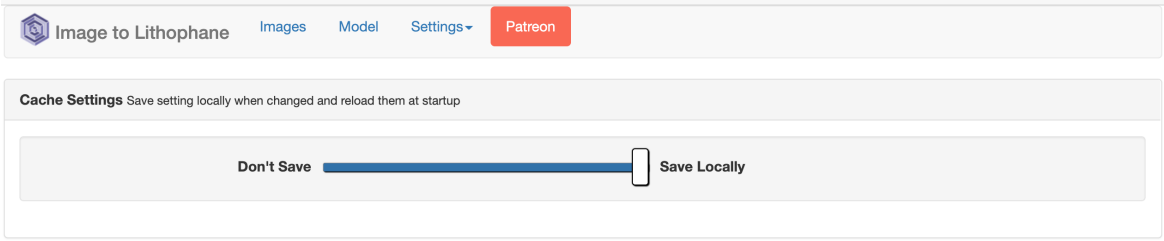

Fig. 1.6 Cache settings

1.2.2 Algorithmic 3D Modeling

A program to generate 3D models by Dr. Dirk Colbry - Michigan State University.

He write Python code on Google Colab for demo. He also provide explanation slide and soucre code from:

- Google Slide
- His source code[2] construct from 3 main parts:
  1. Reading image data into Python
  2. Generate Flat Lithophane

$$z = h(1 - \frac{p}{255}) + d$$

Fig. 1.7 An example by Dr. Dirk Colbry

where:
z - depth value for each pixel
h - hight of lithophane (thickness)
p - pixcel value(0-255)
d - default depth

3. Generate Cylinder Lithophane

1.2.3 A NodeJS utility to turn images into STL lithopanes for 3D printing

In this project, Will Turnage write a Node.js command line utility to turn images (jpg/gif/png) into STL lithopanes for 3D printing[8].His workflow is:

1. Convert the image into an array of heights representing grayscale values.

2. Convert those heights into a series of 3D rects in space and parse those rects into triangles for rendering.

3. Convert to STL or ASCII format

Chapter 2

Project Specifications

2.1 The functional requirements

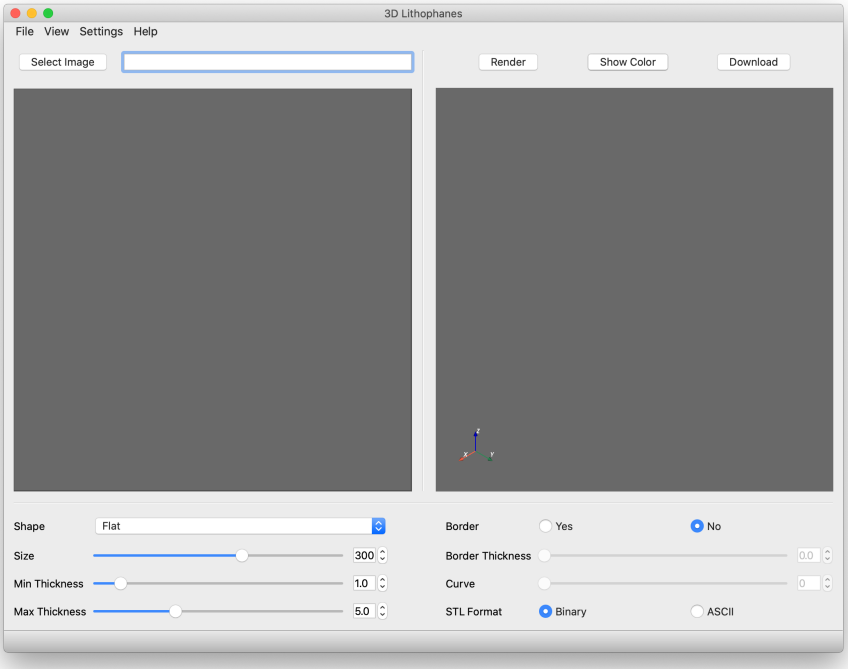

Fig. 2.1 Demo of user interface

The functional requirements to convert an image to a 3D lithophane is listed below:

- Image Upload: a function for users to select a PNG or JPG image in their local computer to convert to a lithophane. This function returns the image's direct path.

- Shape selection: a drop-down list for users to select a model's shape that they want to print the lithophane, such as plane, cylinder, curver and heart.
- Render: after select an image and configure all non-functional parameters, users click Render button to start converting the image to 3D lithophane. Render function finally returns a stl file as output, which contains a matrix whose element illustrates X, Y location and Z height.
- Show/Hide Color: to visualize the depth of pixels by color which called elevation map.
- Download STL: to download and save the stl file (with 2 format: Binary and ASCII) for later use with a 3D printer.
- Render configurations:
  - Border: allow to use border or not, also allow user to change the size of border from 0.0 to 50.0
  - Size: resize the image with the width from 1 to 500. Because we cannot prevent user input an 4K resolution image, which slow down the application.
  - Thickness: include min thickness (0.1-10.0) and max thickness (0.1-15.0) which response for hight of lithophane.
  - Curve: Curve degree (0-360) for 2 kinds of shape: Curver and Cylinder.

2.2 The non-functional requirements

2.2.1 Portability and compatibility

- This is a desktop application.
- This program can run on the difference Operating System:
  - Windows 10 and above
  - Linux (Ubuntu 16.04 and above)
  - Mac OSX 10.14 and above

2.2.2 Usability

- The user interface should be simple as possible as and friendly with the user.
  - Look and feel standards:
    - Layout is divided to 3 parts: Input, Output and Configurations and work flow go from Input to Configurations and finally show result in Output.
    - Screen element density
    - Keyboard shortcuts: Use common shortcuts by references other application
    - Colours: Only use light theme, Input and Output graphics will be highlight.
  - Localization/Internationalization requirements: Keyboards, languages, spellings,...
  - Information: Clear message in information, warning or error dialogs.

2.2.3 Reliability and Maintainability

- Each component of the project will be tested parallel to the implementation.
- Using PyUnit for testing and testing coverage over 80%
- Operational issues, exceptions and errors should be control and write to logs.
- All module should be implemented separated and independent.
- All design should be well explanation and compared with other designs.

2.2.4 Extensibility

- Application can handle new information types like add more shape or change size, thickness,...
- Can manage new or changed business entities

2.2.5 Performance

- The time of rendering must be fast for maximum width and height is 500 pixels
  - Total rendering time is under 3 seconds.
  - Converting to points cloud under 1 seconds.
  - Add facets from points cloud under 1 seconds. Maximum facets with 500x500 size is 501,000 (500*500*2 + 500*2).

Chapter 3

Theory

3.1 Overview

The process to convert a 2D digital image into a 3D lithophane map includes 4 main steps:

- Convert an image to gray scale
- Add 3rd dimension for gray level
- Construct mesh from points cloud
- Generate STL file

3.2 Convert an image to gray scale

As state before, lithophanes is a artwork panel that can be seen only with back light. Since the panel is made of one specific material, it can only illustrate one color tone. So that we have to transform 3 RGB colors into 1 color, which is the gray tone as it is the most popular for lithophanes.
There are many methods to convert a RGB image to a gray scale one. But in this project, we will study on the most 2 popular methods, which are:

- Average method
- Weighted method or luminosity method

3.2.1 Average method

Average method is the most simple one. The three RGB color parameters of a pixel will be averaged to be a gray tone value.

$$Grayscale = \frac{R+G+B}{3}$$

Fig. 3.1 A RGB image

Fig. 3.2 Gray scale image using average method

As we can see, the result of average method is not so good and quite dark. It is because the three RGB colors have different wavelength and contribution in the formation of image, so the average should be taken according to their contribution, only simply on their value on RGB scale. In this average method the contribution of each RGB color is considered the same 33%. Balancing this contribution will help improve the quality of gray scale image, which is discussed in the next section.

3.2.2 Weighted method or luminosity method

Physically, red color has a longest wavelength of all RGB colors. While green is not only the median wavelength but also gives more soothing effect to human eyes. And after some experiment, the luminosity method comes up with a new equation for gray scale converting, which is:

$$Grayscale = (0.3 * R) + (0.59 * G) + (0.11 * B)$$

Result of this method is brighter compare to the average method one.

Fig. 3.3 Gray scale image using weighted method

3.3 Add 3rd dimension for gray level

In this section, we will study on how to convert the gray scale level into thickness, which value will be added as the 3rd dimension z in points coordinator together with x and y.
In gray scale, each gray value varies from 0 (black) to 255 (white) but for the lithophanes black color should be the thickest and white should be thinnest, means that black color has the highest gray value and white is the smallest. Thus, we have to invert the gray scale then multiply to the desired thickness of the lithophane. The result then will be added an offset which is the minimum thickness of the lithophane to get the thickness of a pixel.

3.4 Construct mesh from points cloud

We've now got a 3D points cloud, in which each point represents a pixel coordinates x, y and thickness z on the lithophane. Basically, it is enough to construct a lithophane, but for faster and smoother 3D model visualization, we now create triangle surfaces by connecting any 3

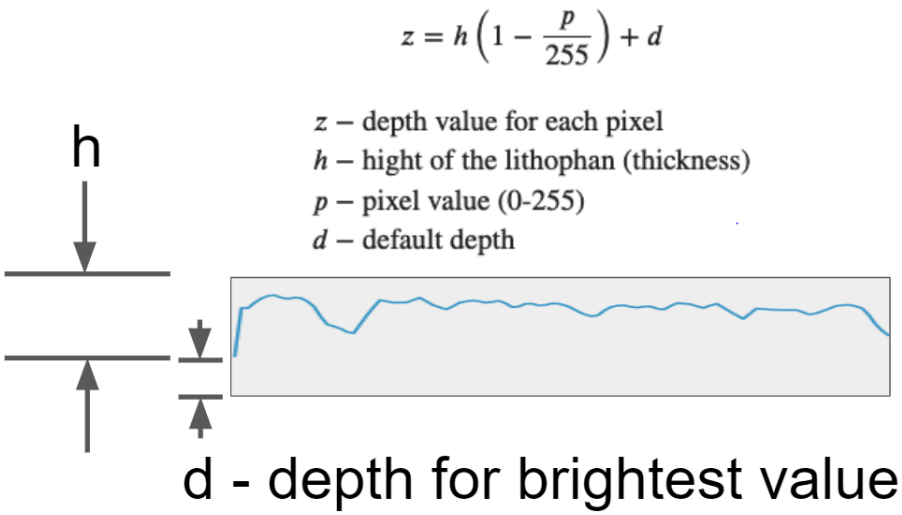

Fig. 3.4 Equation to convert gray scale to thickness

points from the cloud. Those triangle surfaces are called 3D mesh.

When visualize a 3D model from points cloud, the model is a combination of discrete points, so it is not so friendly for users' eyes. But 3D meshes improves that disadvantage.

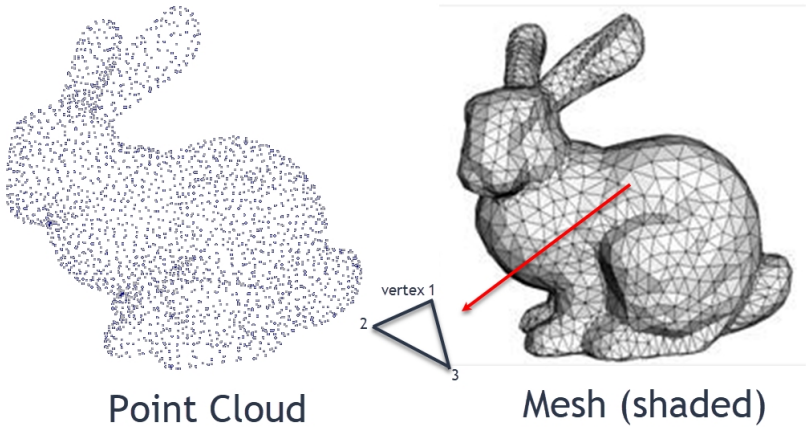

Fig. 3.5 Points Cloud and 3D Mesh

3.5 Generate STL file

Finally, we will need to save the result of lithophane conversion progress in a file type that is supported by 3D printers.

There are many 3D printers supported file type in the world.The most common and universal file formats for 3D printing are STL and VRML. STL stands for “stereolithography” – it is a 3D rendering that contains only a single color. This is typically the file format we would use with desktop 3D printers. VRML (“vermal”, .WRL file extension) stands for “Virtual Reality Modeling Language” – it is a newer digital 3D file type that also includes color, so it can be used on desktop 3D printers with more than one extruder (i.e. two more nozzles that each can print with a different color plastic), or with full-color binder jetting technology.

Additive Manufacturing File Format (.AMF) is a new XML-based open standard for 3D printing. Unlike STL, it contains support for color. They can also be compressed to about half the size of a compressed STL file. AMF is not widely used at present, but in future we would like to add this an option for uploading and downloading files to and from the NIH 3D Print Exchange.

Another file format input for 3D printers in GCode. This file contains detailed instructions for a 3D printer to follow for each slice, like the starting point for each layer and the "route" that the nozzle or print head will follow in laying down the material

Since lithopane is single-color and free of routing instruction for 3D printers, STL file format should be the best one to save the result of lithiphane conversion progress.

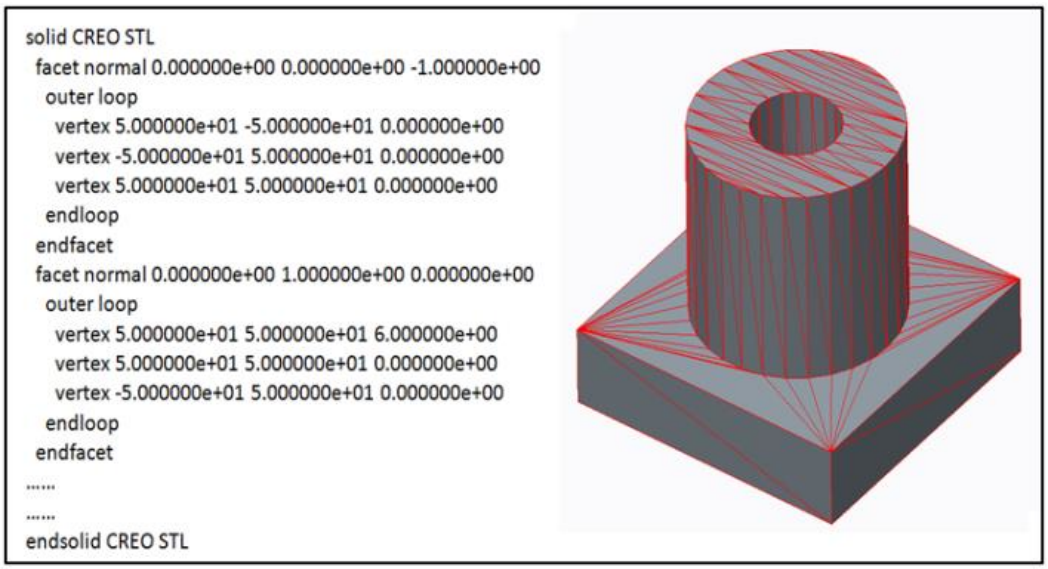

Fig. 3.6 An example of STL file format

Chapter 4

Architecture

4.1 Overview

This chapter provides a overview and explains the architecture of 3D Lithophanes Application. This chapter defines the architecture, the use cases supported by the system, relationship between classes and flow of system when user interact with system.

4.2 Class Diagram

Depend on previous analysis in chapter 2, we design a desktop application have 3 main classes:

- UIApp: which is an user interface for user to interact with the Lithophane converter application.
- Lithophane: a main system which is responsible for logical processing to convert a 2D image to 3D lithophane model.
- Config: Manage all configuration to customize the output result.
- STL Generator: export STL file which is the result of this application, as well as the input for 3D printers to construct the lithophane.

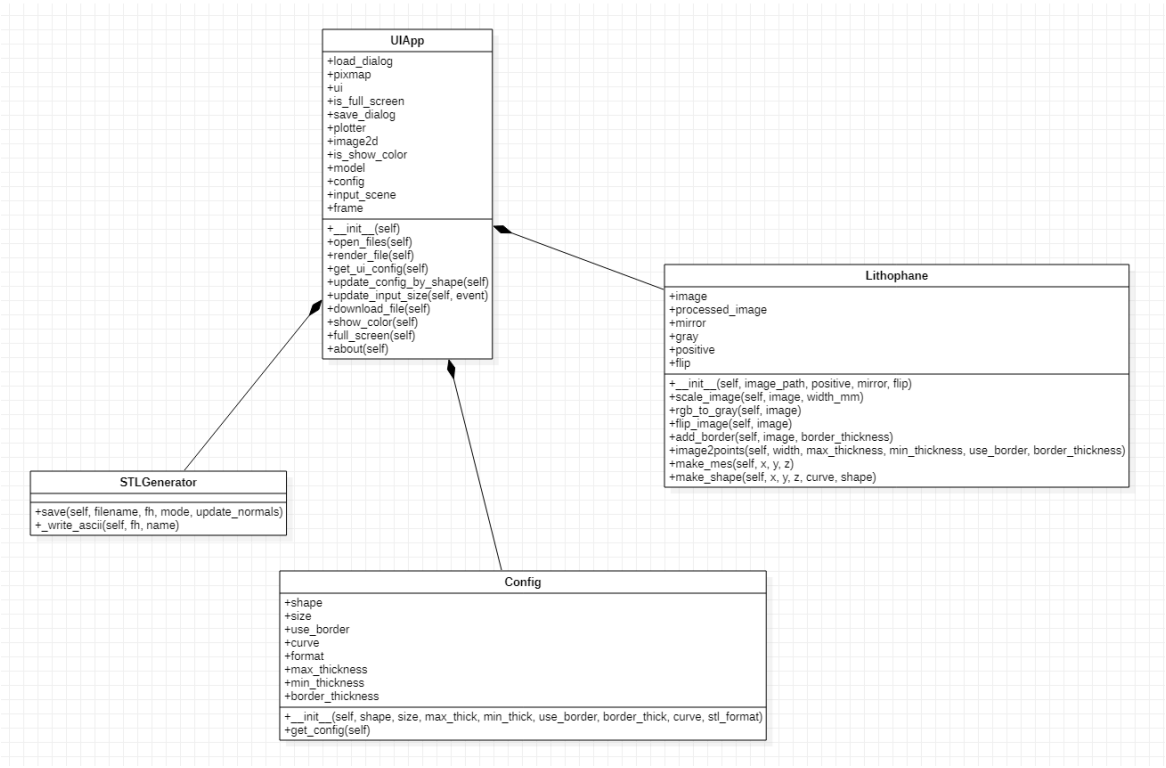

Fig. 4.1 Class Diagram

4.3 Use Case Realizations

This section will describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.[3]

There are 3 main use cases that are "Select Image", "Render Lithophane" and "Download STL". Beside that, some steps are optional such as configuration modification, border enabling, STL file type selection, entering full screen, minimizing window, application exit, about information. Similarly, you have to select a 2D image then render a 3D model before download it as STL file.

In most cases, we use a sequence diagram to illustrate use-case realizations i.e. to show how objects interact to perform the behavior of all or part of a use case. One or more sequence diagrams may illustrate the object interactions which enact a use case. A typical organization is to have one sequence diagram for the main flow of events and one sequence diagram for each independent sub-flow of the use case.[3]

In this application, we visualise flow of system (UI and Converter) follow by user action.

4.3.1 Select Image

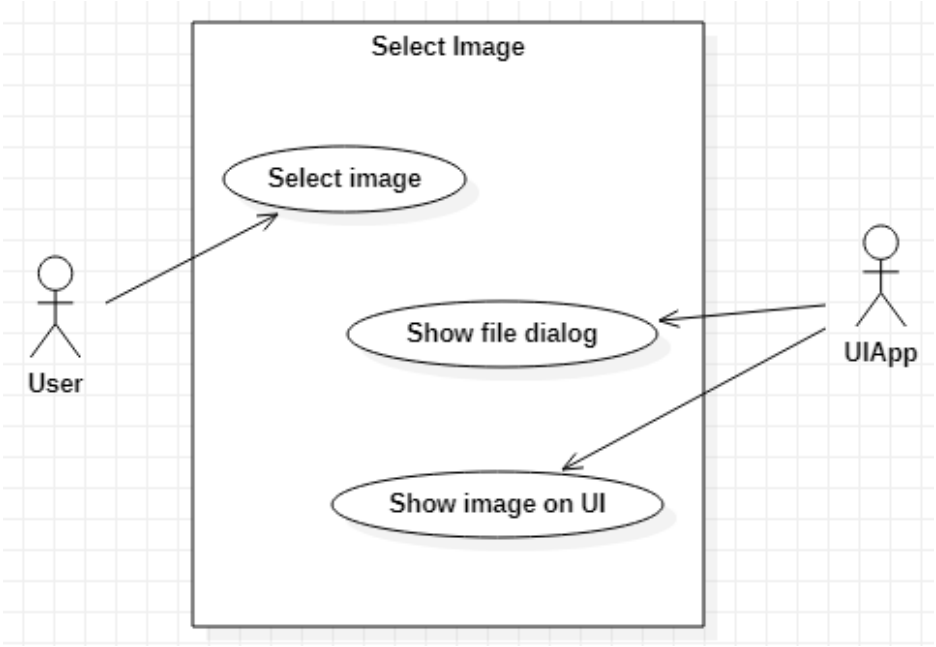

Fig. 4.2 Use case diagram of select image

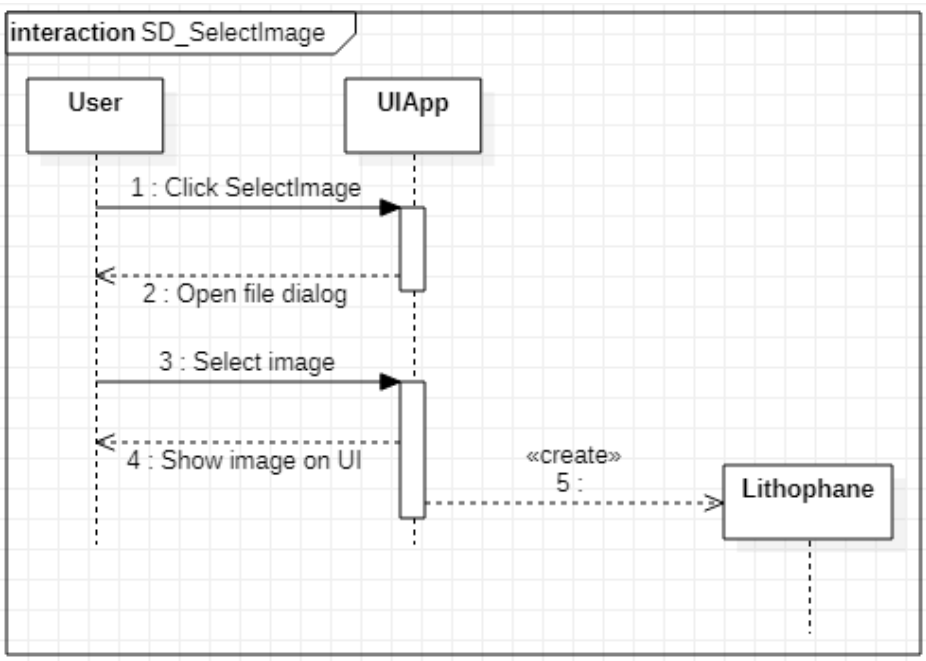

Fig. 4.3 Sequence of select image

4.3.2 Render 3D model

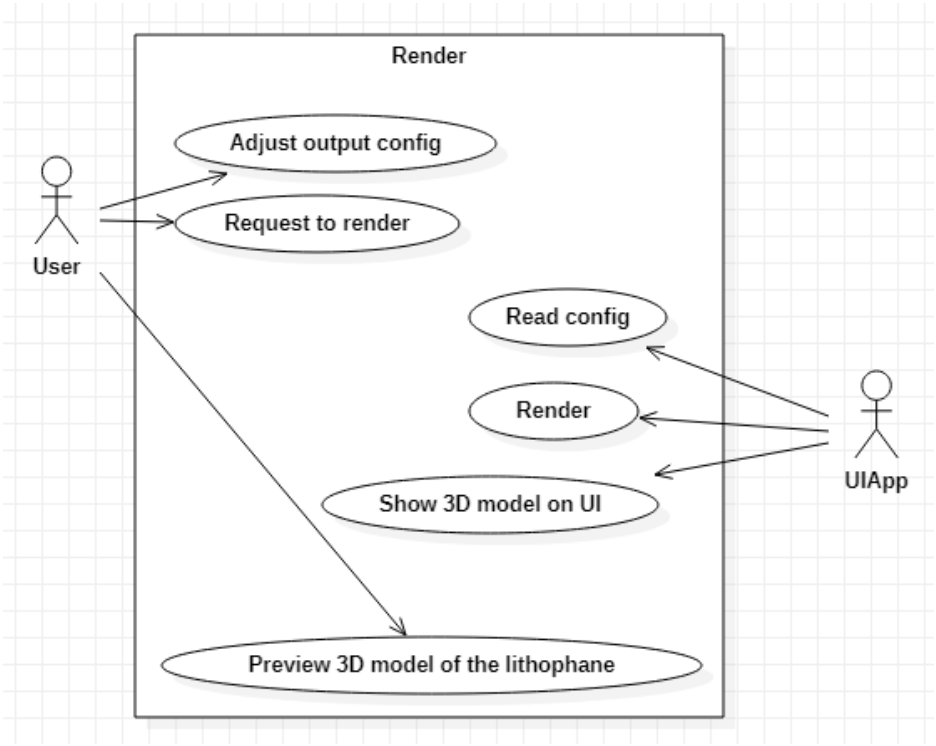

Fig. 4.4 Use case diagram of render 3D model

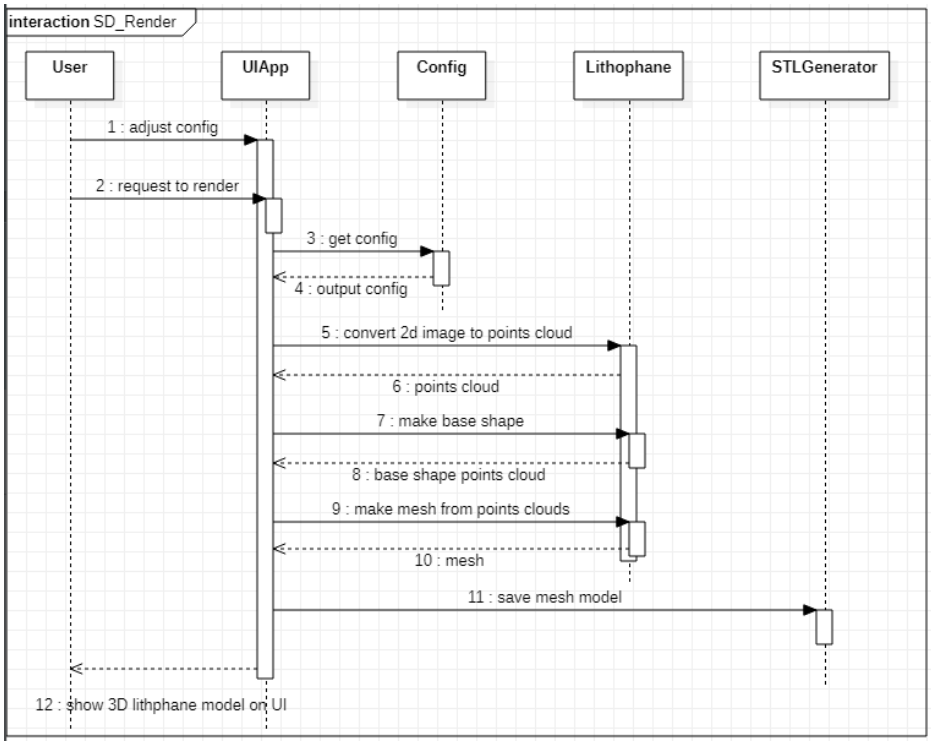

Fig. 4.5 Sequence of render 3D model

4.3.3 Render 3D model with color

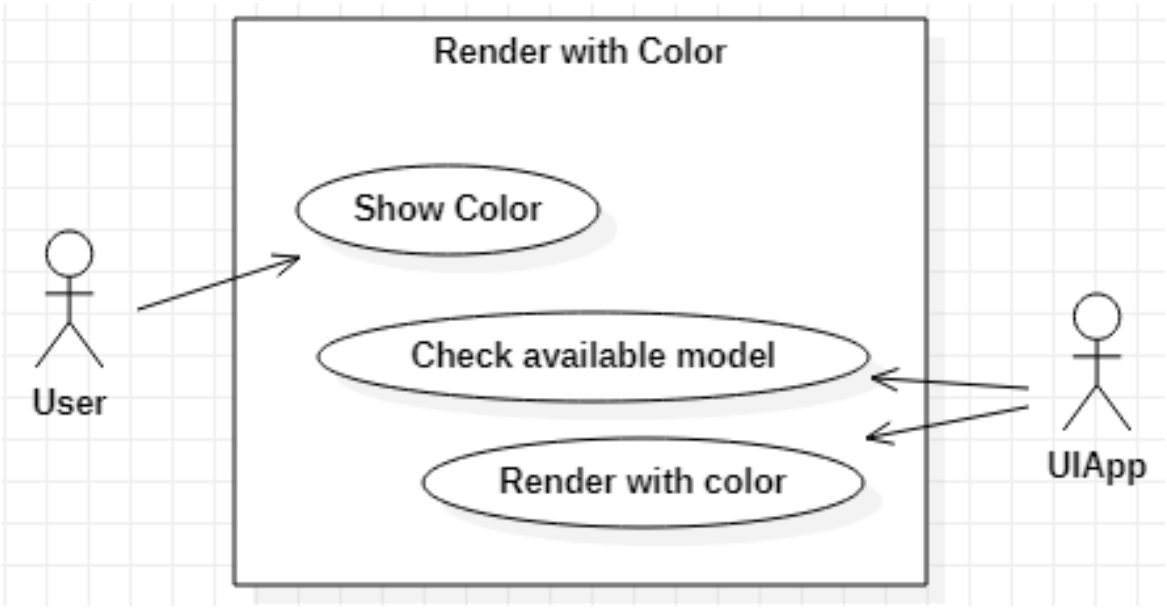

Fig. 4.6 Use case diagram of render 3D model with color

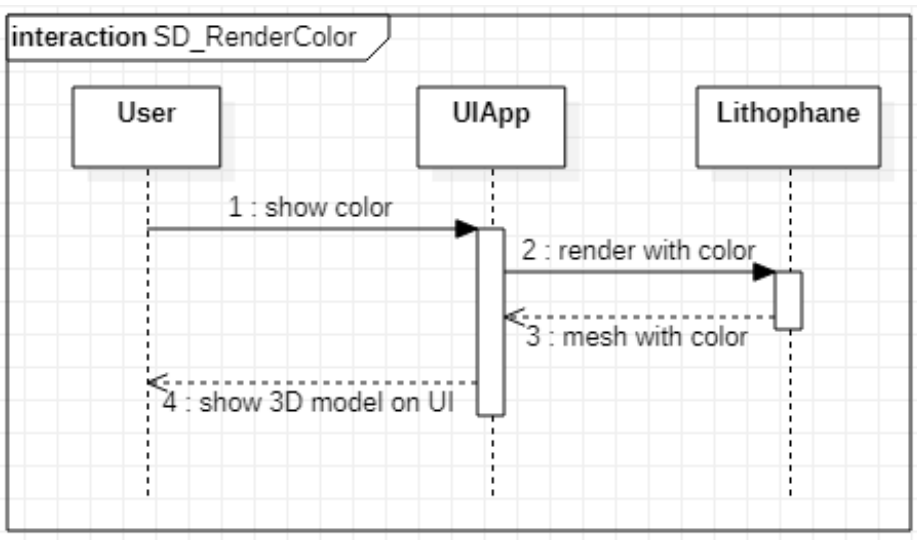

Fig. 4.7 Sequence of render 3D model with color

4.3.4 Download STL file

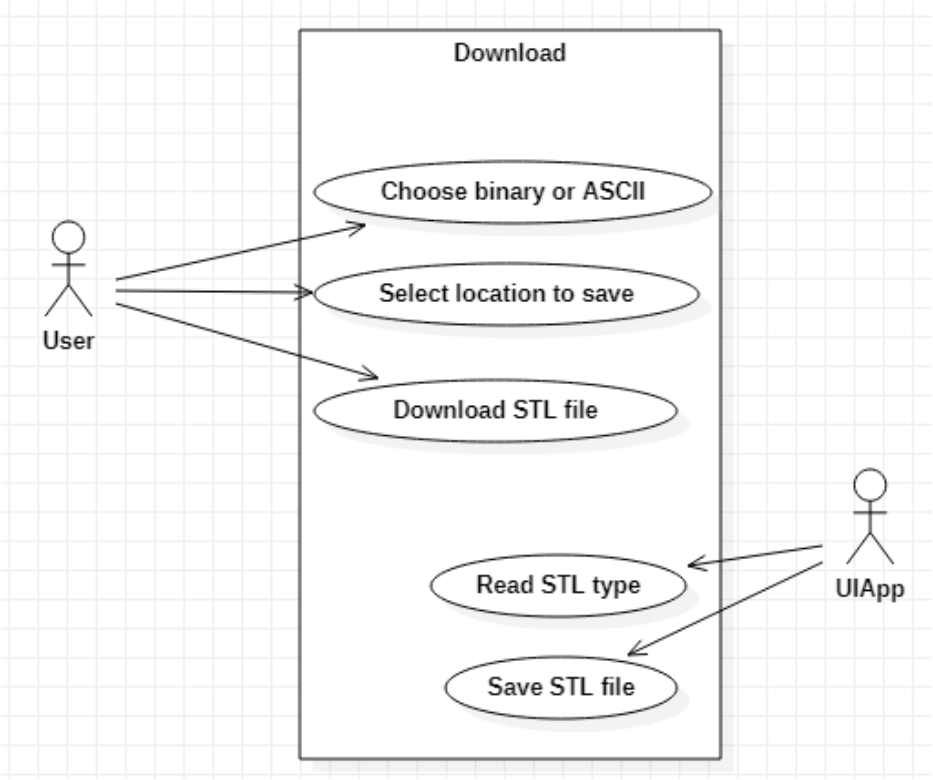

Fig. 4.8 Use case diagram of download STL file

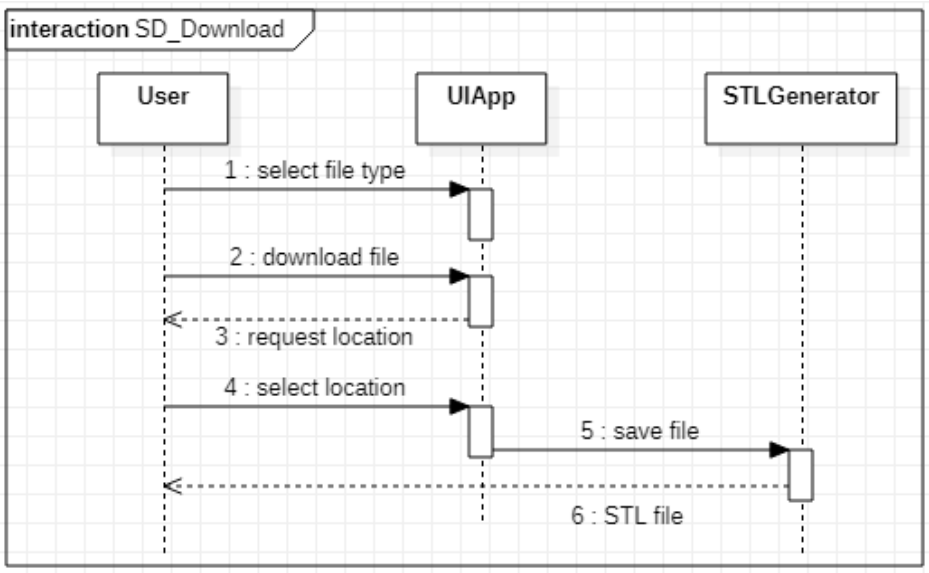

Fig. 4.9 Sequence of download STL file

4.4 User Interface

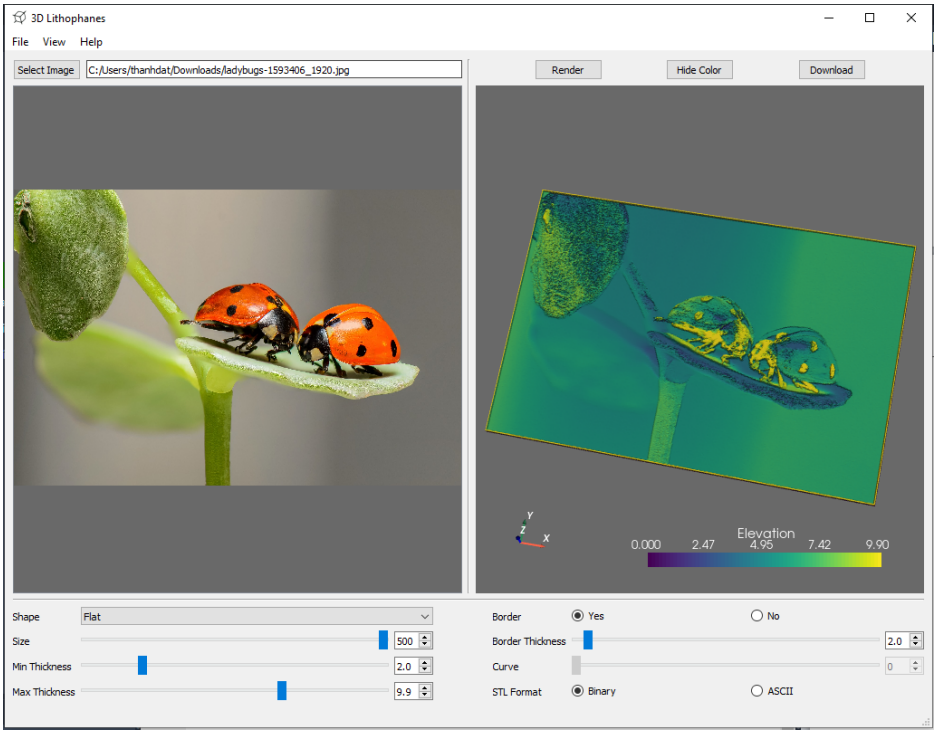

Fig. 4.10 User Interface

Chapter 5

Implementation and Deployment

In this chaper we will describle the actual implementation, necessary tools and how to deployment the application.

5.1 Source code

We use GitHub to store and tracking coding progress at: https://github.com/calemolech/lithophanes.

- There are 3 branches: *master*, *dev* and *dev-ui*
- This repository also have some features like CI/CD to auto intergrate and deployment. We also build a binary (run-able) version for MacOS and Linux. Releases link

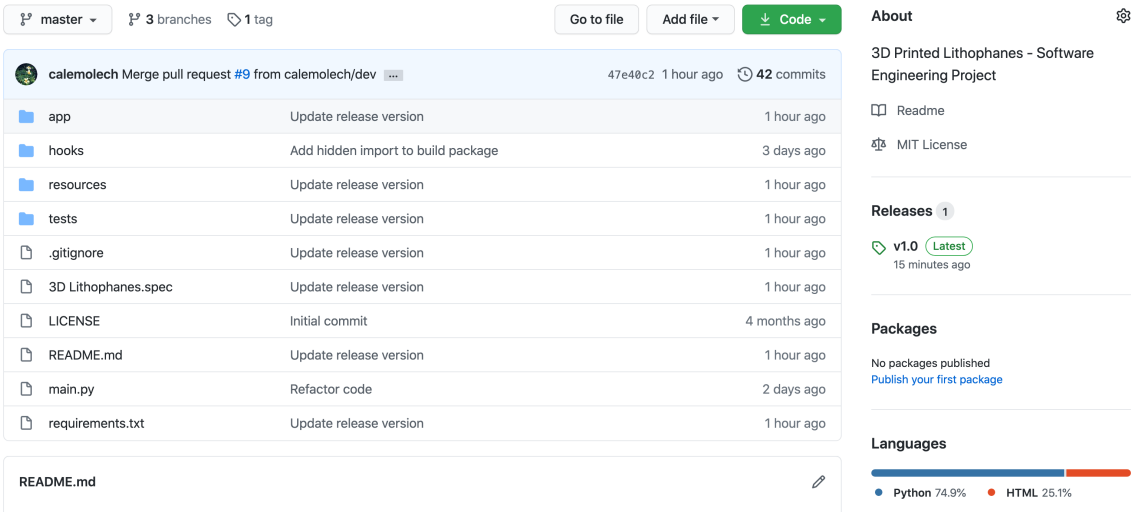

Fig. 5.1 Github Repository

5.2 Tools and Frameworks

In this project, we also use some common tools

- IDE: PyCharm
    1. PyCharm is used widely in developer community in the world
    2. Pycharm is free and also provide license for student account.
    3. PyCharm support a ton of plugins which increase development efficiency.
- GUI: Qt Designer
- Source control: Git
- Deployment: PyInstaller
- Document, Report: LaTex on Overleaf

About frameworks and libraries:

- GUI: We use PyQt5 to create GUI and interact with backend.
- Libraries:
    - OpenCV and Pillow for 2D image processing.
    - Vtk and PyVista to visualize 3D output.
    - HTML Runner to export HTML testing result.

5.3 Deployment

5.3.1 General

As many other project in Python, to run our application you should:

1. Install the Virtual Environment (virtualenv) package: ***pip install virtualenv***
2. Create Virtual Environment: ***virtualenv venv***
3. Activate Virtual Environment: MacOS/Linux: ***source venvbinactivate*** or Windows: ***venv***
   Scripts
   activate

```
altgraph==0.17
appdirs==1.4.4
distlib==0.3.1
filelock==3.0.12
html-testRunner==1.2.1
imageio==2.9.0
importlib-metadata==1.7.0
Jinja2==2.11.2
macholib==1.14
MarkupSafe==1.1.1
meshio==4.2.0
numpy==1.19.1
numpy-stl==2.11.2
opencv-python-headless==4.4.0.42
Pillow==7.2.0
protobuf==3.13.0
pyinstaller==4.0
pyinstaller-hooks-contrib==2020.7
PyQt5==5.15.1
PyQt5-sip==12.8.1
python-utils==2.4.0
pyvista==0.26.0
pyvistaqt==0.2.0
scooby==0.5.6
six==1.15.0
virtualenv==20.0.31
vtk==9.0.1
zipp==3.1.0
```

Fig. 5.2 All requirement packages

4. Install requirement packages: ***pip install -r requirements.txt***

5. Run application: ***python main.py***

5.3.2 Binary

In this section, we talk about how to packaging application.

We use the package called PyInstaller - A tool freezes (packages) Python applications into stand-alone executables, under Windows, GNULinux, Mac OS X, FreeBSD, Solaris and AIX. PyInstaller support many operating system and ease to use.[4]

In order to run in end-user without install any thing we must create an executive file. However, this project is image processing in Python so there are a lot Python libraries, which leading

to the size of application must be very big (about 190MB).

This is command to build application:
pyinstaller –onefile –additional-hooks-dir=hooks –name="3D Lithophane" –noconsole –icon=".resourcesiconsapp.icns" –runtime-tmpdir="tmp main.py

ID	Arguments	Description
1	–onefile	Create a one-file bundled executable.
2	–additional-hooks-dir	An additional path to search for hooks. This option can be used multiple times.
3	–name	Name to assign to the bundled app and spec file (default: first script's basename)
4	–noconsole	Windows and Mac OS X: do not provide a console window for standard i/o. On Mac OS X this also triggers building an OS X .app bundle. On Windows this option will be set if the first script is a '.pyw' file. This option is ignored in *NIX systems.
5	–icon	Path to icon
6	–runtime-tmpdir	Where to extract libraries and support files in onefile-mode. If this option is given, the bootloader will ignore any temp-folder location defined by the run-time OS.

Table 5.1 Arguments table

The result will be generated in 2 folder *dist* and *build*. You can copy file in *dist* to other computer and run. However, we are un-identify developer so in MacOS, you must allow to run this application.

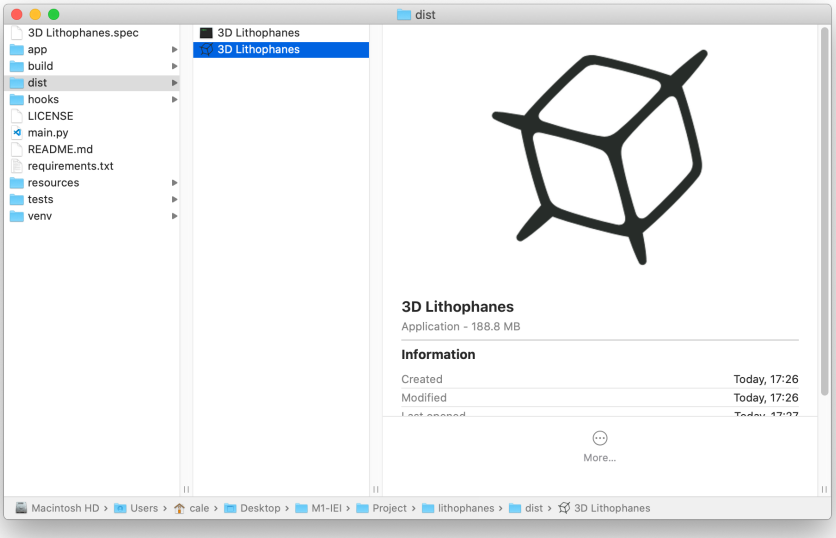

Fig. 5.3 *dist* folder

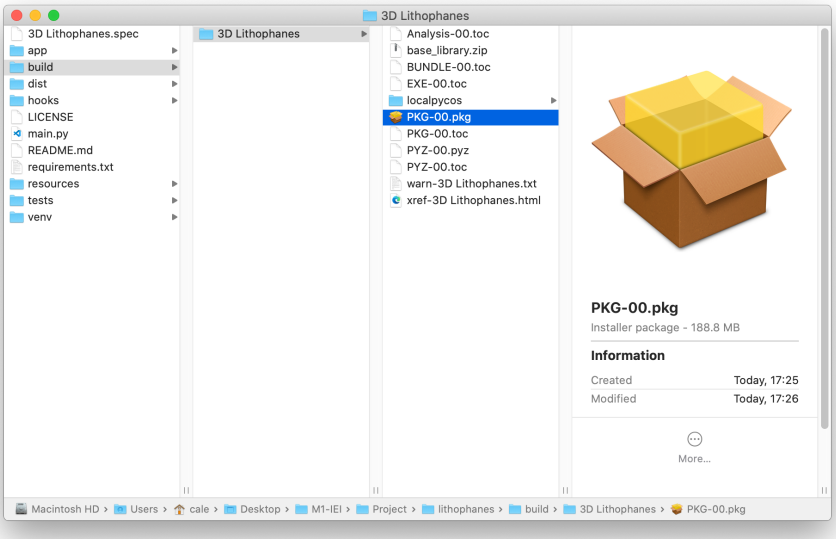

Fig. 5.4 *build* folder

Chapter 6

Tests and Results

6.1 Overview

There are two common testing framework is Python's built-in Unittest (PyUnit) framework and Pytest. Both of them are powerful framework, but we choose PyUnit because these following reasons:

- Fixtures are simple and easy to use.
- Support to work with many extensions.
- Generating HTML tests result.

Although Unittest has very bad UX like: camelCase naming while Python using snake_case, no corlor output. However, it is not effect to the test case and result, we also use the extension like HTML Test Runner to export result to HTML format.

6.2 Unit Test

- All function must be tested by unit test.
- All class must be tested.

6.3 Testing scenarios

6.3.1 UI

- Can user press button "Select image" and load image?

- Can user change the shape: Flat, Cylinder, Curve, Heart?
- Can user change settings?
  - Input, output size
  - Min, max thickness
  - Turn on/off border
  - Border thickness
  - Curve settings
- Can user download the STL file?

6.3.2 Functional Testing

- Can user load and show correct image?
- Can user load 3D image after rendering?
- Can user get correct shape after change shape setting?
- Can user get correct output when changing settings?
- Can user download correct STL file from 3D viewer?

6.3.3 Non-Functional Testing

- Do system delay, not responding when rendering?
- Is rendering time bellow 5 seconds?
- Can system run on many Operating System:
  - Windows 10 and above
  - Linux (Ubuntu 16.04 and above)
  - Mac OSX 10.14 and above
- How much time the app starts up?
- How easy is it for maintenance?
- How much memory increases while using the app?

6.4 Test Result

6.4.1 Test coverage and unit test result

- There are 33 tests are created and all of them are passed.
- Coverage in general is 92% lines of code. In detail, app source code have coverage rate is 77%, this score is low because it is very difficult to check output of PyQt like Dialog, Message Box or simulate output of button.

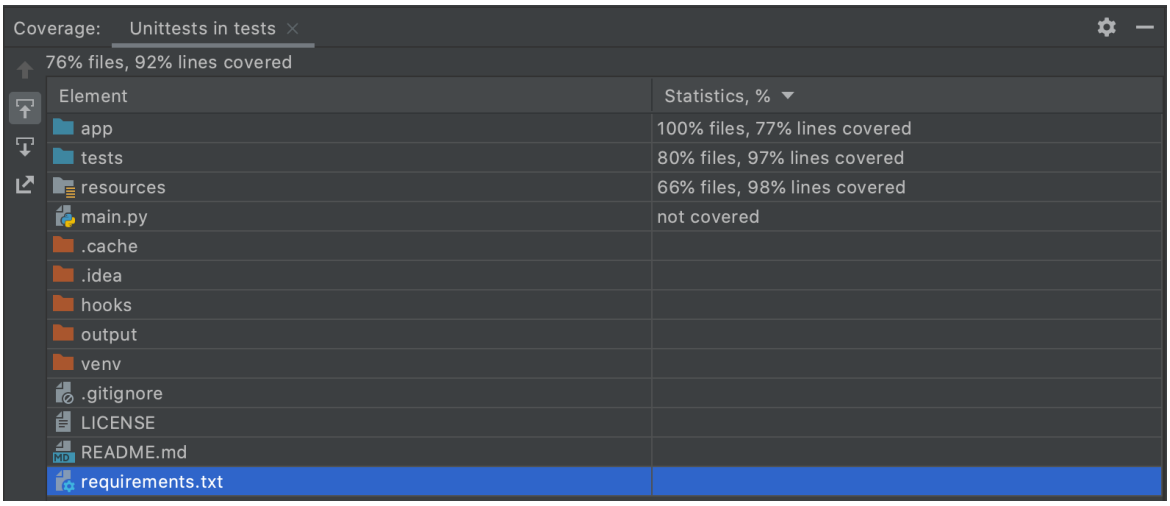

Fig. 6.1 Coverage of whole project - Generated by PyCharm

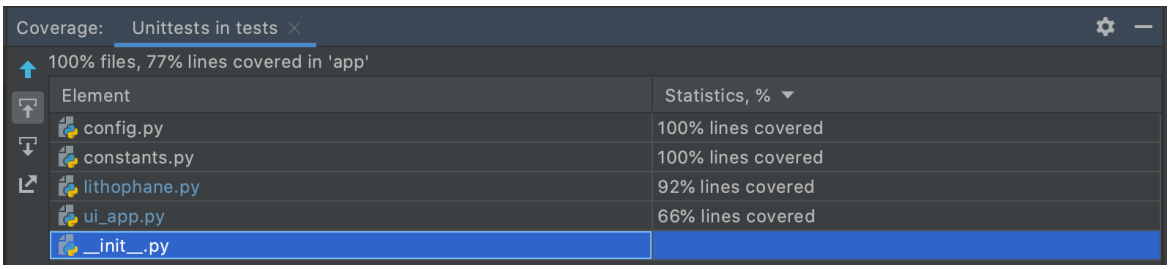

Fig. 6.2 Coverage of whole application - Generated by PyCharm

Report Unittest UI

Start Time: 2020-09-14 20:36:42

Duration: 3.24 s

Summary: Total: 24, Pass: 24

__main__.TestUi	Status
test_border_radio	Pass
test_border_thick_change	Pass
test_border_thick_out_range	Pass
test_curve_change	Pass
test_curve_out_range	Pass
test_format_radio	Pass
test_full_screen	Pass
test_get_ui_config	Pass
test_init_ui	Pass
test_init_value	Pass
test_max_thick_change	Pass
test_max_thick_out_range	Pass
test_min_thick_change	Pass
test_min_thick_out_range	Pass
test_render	Pass
test_select_image	Pass
test_shape_change	Pass
test_show_normal	Pass
test_size_change	Pass
test_size_out_range	Pass
test_update_config_by_shape_curve	Pass
test_update_config_by_shape_cylinder	Pass
test_update_config_by_shape_flat	Pass
test_update_config_by_shape_heart	Pass

Total: 24, Pass: 24 -- Duration: 3.24 s

Fig. 6.3 Result of UI tests

Report Unittest Config

Start Time: 2020-09-14 15:12:15

Duration: 0 ms

Summary: Total: 2, Pass: 2

__main__.TestConfig	Status
test_default_config	Pass
test_get_config	Pass

Total: 2, Pass: 2 -- Duration: 0 ms

Fig. 6.4 Result of Config tests

Report Unittest Lithophane

Start Time: 2020-09-14 20:49:05

Duration: 3.81 s

Summary: Total: 5, Pass: 5

__main__.TestLithophane	Status
test_add_border	Pass
test_flip	Pass
test_image_to_points	Pass
test_rgb_to_gray	Pass
test_scale_image	Pass

Total: 5, Pass: 5 -- Duration: 3.81 s

Fig. 6.5 Result of Lithophane tests

Report Unittest Lithophane Mesh

Start Time: 2020-09-14 19:48:01

Duration: 221 ms

Summary: Total: 4, Pass: 4

__main__.TestLithophane	Status
test_make_mesh	Pass
test_make_shape_curve	Pass
test_make_shape_cylinder	Pass
test_make_shape_heart	Pass

Total: 4, Pass: 4 -- Duration: 221 ms

Fig. 6.6 Result of Lithophane (create shape and mesh) tests

6.4.2 Testing scenarios result

- UI

ID	Description	Status	Update date
1	Select and Load image?	Passed	14/09/2020
2	Change shape: Flat, Cylinder, Curve, Heart	Passed	14/09/2020
	Configurations		
3	Change input size?	Passed	14/09/2020
4	Change output size?	Passed	14/09/2020
5	Change minimum thickness?	Passed	14/09/2020
6	Change maximum thickness?	Passed	14/09/2020
7	Turn border on or off?	Passed	14/09/2020
8	Change border size?	Passed	14/09/2020
9	Change the curve?	Passed	14/09/2020

Table 6.1 UI Testing

- Functional Testing:

ID	Description	Status	Update date
1	Can user load and show correct image?	Passed	14/09/2020
2	Can user load and show correct image?	Passed	14/09/2020
3	Can user load 3D image after rendering?	Passed	14/09/2020
4	Can user get correct shape after change shape setting?	Passed	14/09/2020
5	Can user get correct output when changing settings?	Passed	14/09/2020
6	Can user download correct STL file from 3D viewer?	Passed	14/09/2020

Table 6.2 Functional Testing

- Non-Functional Testing:

6.5 Conclusion

The system fully satisfied the initial requirements from client. It can be used to convert image from 2D to 3D Lithophane with a lot of preferences.
Furthermore, during the project, the developing team managed to develop their own skills, and satisfied with the final result. In order to achieve current result, the team especially focused on teamwork (each member has a individual task and role based on their skills and

ID	Description	Status	Update date
1	Do system delay, not responding when rendering?	Passed	14/09/2020
2	Is rendering time bellow 5 seconds?	Passed (3s)	14/09/2020
3	Can system run on Windows 10 and above?	Passed	14/09/2020
4	Can system run on Linux (Ubuntu 16.04 and above)?	Passed	14/09/2020
5	Can system run on Mac OSX 10.14 and above	Passed	14/09/2020
6	How much time the app starts up?	Passed (<1s)	14/09/2020
7	How easy is it for maintenance?	Passed (OK)	14/09/2020
8	How much memory increases while using the app?	Not Measured	14/09/2020

Table 6.3 Non-Functional Testing

experiences, but they also cover others to ensure the rate of progress. The decisions were also based on discussion between team's members, or communication with supervisors.
As a result, each member not only improve skills and experiences but also learn to manage time thus leading to the successful project.

References

[1] Cabello, R. (2010-2020). Javascript 3d library. https://github.com/mrdoob/three.js/.

[2] Colbry, D. (2020). Python lithophane stl generator. https://github.com/colbrydi/Lithophane.

[3] Corporation, R. S. (1987-2001). Software architecture document. https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/process/artifact/ar_sadoc.htm.

[4] David Cortesi, based on structure by Giovanni Bajo & William Caban, b. o. G. M. m. (2020). Pyinstaller manual. https://pyinstaller.readthedocs.io/en/stable.

[5] Durbin, M. (2015). 3dp.rock - lithophane: Utility for converting images to lithophanes for 3d printing. http://3dp.rocks/lithophane/.

[6] Grey, E. (2011-2020). An html5 saveas() filesaver implementation. https://github.com/eligrey/FileSaver.js.

[7] Kaplan, P. (2014-2020). Writing binary and ascii stl files in javascript. https://gist.github.com/paulkaplan/6d5f0ab2c7e8fdc68a61.

[8] Turnage, W. (2015). Node.js command line utility to turn images (jpg/gif/png) into stl lithopanes for 3d printing. https://github.com/wubbahed/lithophane.