

---

**Robotic Artist:Walter**  
**G400 Computer Science, CS39440**  
**Design Specification**

---

Author: Matthew Howard  
Candidate Number: 150035512  
User ID: Mah60  
Supervisor Name: Helen Miles  
Supervisor ID: Hem23  
Last Modified: 25/04/2018  
Version: 0.2  
Status: Work In Progress(WIP)

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Version Control</b>	<b>2</b>
<b>3</b>	<b>UML Diagrams</b>	<b>3</b>
3.1	Class Diagram . . . . .	3
3.2	Sequence Diagrams . . . . .	4
<b>4</b>	<b>Graphical Unit Interface(GUI) Design</b>	<b>6</b>
<b>5</b>	<b>Versions</b>	<b>9</b>

## 1 Introduction

This document is going to go look at the overall design of the product as a whole. What the structure of the product is and how the maintains the integrity of the design.

The Design specifications will talk about version control and the different UML diagrams that can be drawn from the structure of the product that has been developed. This involves UML diagrams for sequences and class layouts.

## 2 Version Control

Version control is maintained in this project through a Github Repository. As Github stores, different versions of the code that have been committed to the repository as the project has gone on. Github also allows for branches to be made in which you can try out the code and suggest it to implement the master version of the product stored on git hub.

Github has been used in this project to maintain versions of the code and could be used to backtrack the code if there are any major errors or faults in the project.

Another that there has been version control within the code and documentation is at the top of every module and end of everything document there is a version history part. This talks about what has been done to modify the document in a clear way and the date this was done. An example of this can be seen in Figure 1.

```
***
Summary => threading to capture video footage.

Description => creates a thread that will continuesly capture images at
               the rate of the fps given. This will then update the image
               to display a video on the gui.

Author => Matthew Howard (mah60).
Version => 0.1 - 12/03/2018 - create the basic set up for the VideoCapture.
               with the init method and run method.
               0.2 - 15/03/2018 - added te static variable stopped, to be able to
               stop the feed in other classes when asked.
               0.3 - 01/04/2018 - this class now has the ui being passed to it.
               This then allows us to continusly update the ui images on
               run time. Also has a image_update method passed to it.
               0.4 - 15/04/2018 - changed code to support python2 instead of
               python3. This required me to change from threading to
               QThread. As well, the thread now sends a signal to the
               main gui thread and will take a thread from the main gui
               to update the other images when this thread is complete.
               http://euanfreeman.co.uk/pyqt-qixmap-and-threads/ is
               the link that I used to help develop the signal connection.
               last seen: 15/04/2018
               0.5 - 20/04/2018 - removed __del__ code. As it had no code inside.
***
```

Figure 1: Demonstrates the Version control that is being used in the video\_capture.py Module. This includes the version number, description and date of change.

### 3 UML Diagrams

In this section of the Design specifications, the document will talk about the framework of the code. How the code is structured and runs on real-time events that occur. This will be done by using UML diagrams.

#### 3.1 Class Diagram

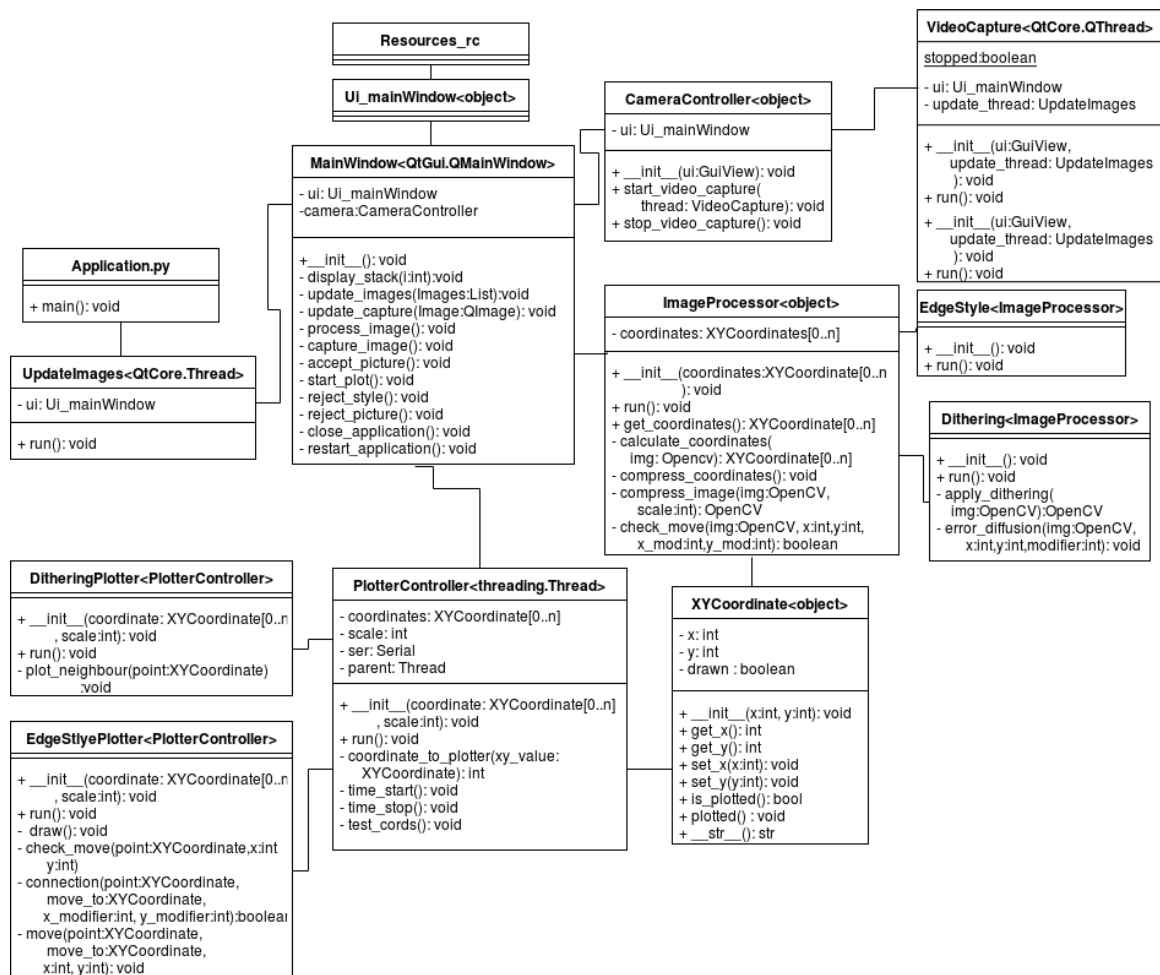


Figure 2: Displays the UML class diagram. This is the structure of all the classes in the products code.

Figure 2 describes the structure of the code that has been developed. The structure of the code has been developed around the MainWindow class. This is where the code for controlling the graphical unit interface(GUI) is. This code is connected to the Ui.Mainwindow, which is a set of code that is developed from the QT Designer software. This allowed the overall framework for the GUI to be developed using various tools within the software.

After this there are three major sections to the class diagram(Figure 2). These are the Camera, Image Processor, and the plotter.

The CameraController controls the inputs and outputs from the camera. The video capture is the main thing that this involves. This is done by creating a thread that continuously sends frames to the GUI to be updated. Once the method `stop_video_capture` is called, then the video capture is stopped and the picture is taken from the frame that the thread was on at that time. These sequence of events can be seen in Figure 3. The thread is stopped by there being a static variable within the VideoCapture class called 'stopped' this variable is a boolean. Which is changed to True when the video thread should be stopped and to False when the capture is running.

The next sections of the class diagram are the ImageProcessor class's, this will be a parent to all the styles that are developed for this program. The code that is generated for this class will be used over all the individual ImageProcessors. The main requirements of each style will require is a partner PlotterController to plot the final output and an output of coordinates to be plotted. This is where the XYCoordinate class is used to create a generic structure to translate data between the two classes.

Finally, with have the PlotterController which is a parent class to all plotters for a specific style. This class will contain code that is generic to all styles plotting routines. An example of this is the `coordinate_to_plotter()` method which will translate coordinate values to plotter values. The overview of the plotter requirements is that it must send serial commands in HPGL to the plotter, to plot out the specified style for that plotter controller.

### 3.2 Sequence Diagrams

Figures 3, 4 and 5 describe the sequence of events that are taken towards plotting a Dithering style.

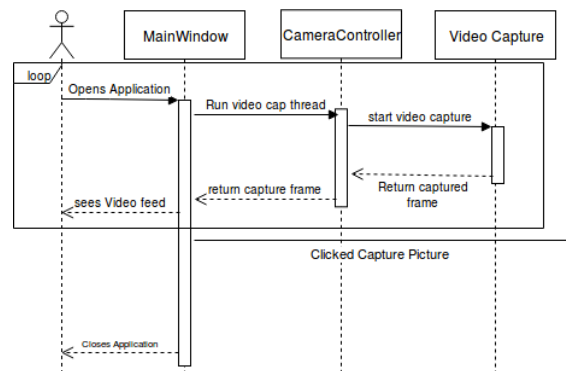


Figure 3: Displays the sequence is taken to display the video feed at the start of the application. This sequence is that when the application is run the MainWindow will go into a continuous loop of talking to the Camera and VideoCapture class to get frames to display on the GUI. When the Capture Image button is pressed by the user, then the GUI will move onto the next page asking if the captured image is suitable for what the user wants.

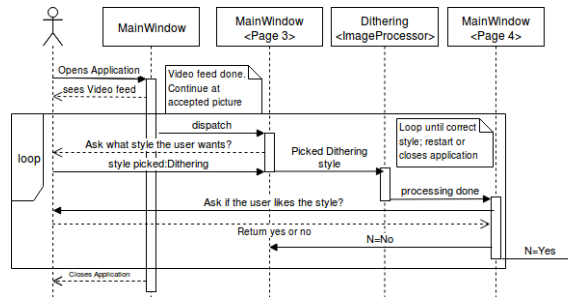


Figure 4: Displays the sequence is taken to process an Image. This is the sequence of events in Figure 3 and the Image has been accepted to be processed. The user will be taken to page 3 of the GUI. This is the style selection page. After one style has been selected the image will be processed for that specified style. In this case, it is the Dithering style. So the program will run the method `run()` in the Image Processor `Dithering.py`. After, this the finalized style will be displayed on page 4. Then there are two sequences of the event's the user can accept the style and start the plot or reject the style and go back to page 3.

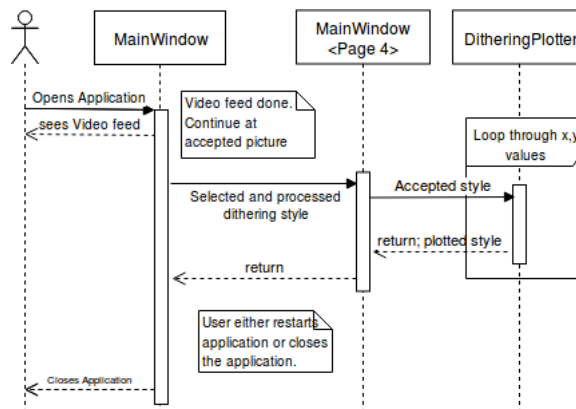


Figure 5: Displays the sequence is taken to plot the style that was processing through given style. This involves when the style has been accepted on page 4, as continued from Figure 4. Then the coordinates are gathered from that style in the `MainWindow` class, then the thread for plotting that specified style begins. This involves sending multiple HPGL commands through serial communication to the plotter. After this task is complete and all coordinates that were made are complete then the application then is restarted from the beginning or the application is closed.

## 4 Graphical Unit Interface(GUI) Design

This section will discuss the design that was developed for the project. The design was developed for python 2.7 using the library PYQT 4. The overall development was done using the software called 'QT Designer', this allowed the project to save time as the GUI didn't take as much time to develop and was shortened dramatically. The project being developed can be seen in Figure 6.

Figures 7, 8, 9 and 10 show the design of the overall GUI. From pages one to four in respective order. These designs were all made with QT Designer and then actions and code were added to the MainWindow class that can be seen in Figure 2.

The design also has a description of what the project is about on the left-hand side of the GUI. This is a label that takes text from the description.txt file that can be found in the code folder.

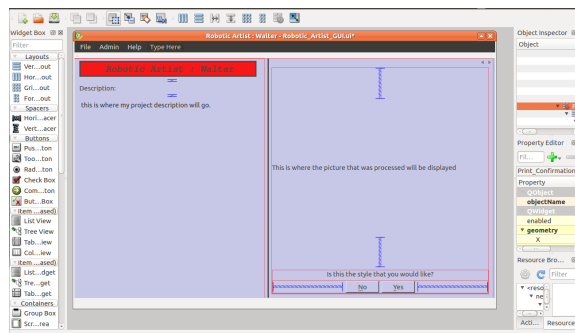


Figure 6: This is an example of how the Qt designer software looked when developing the GUI for the project

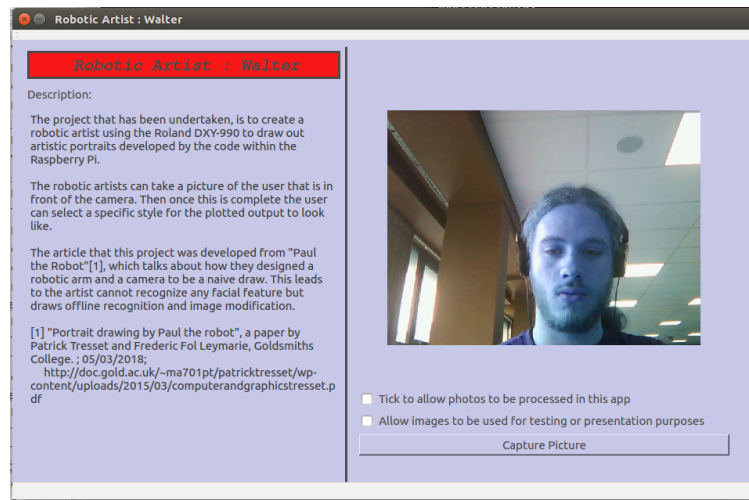


Figure 7: Page 1: Video capture. This page was designed to contain the video capture from the camera. The design of the page is to have a label that a QImage can be added to. During runtime, the image will continuously be updated through a QThread. The other parts of the page are two checkboxes and a button at the bottom of the page, which stops the video capture and take the frame it was on as a picture. This routine can be seen in Figure 3.

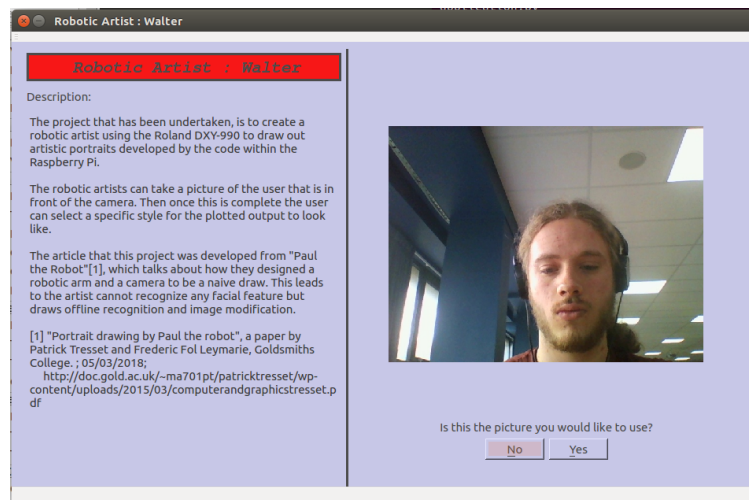


Figure 8: Page 2: Picture Acceptence. This page has a label that will contain the captured frame. Then there are a set of buttons that ask yes or no to whether the picture taken is correct.



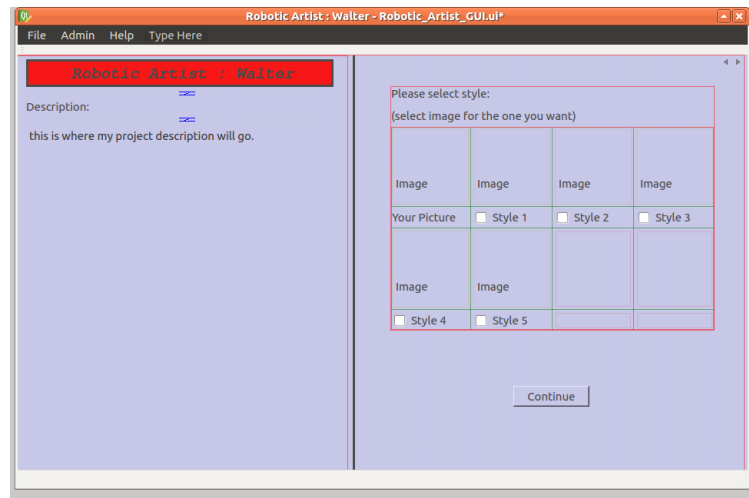


Figure 9: Page 3: Style selection. This page is all about selecting the individual style that you want. There is a grid of H and V box panes with labels inside them that display either text or images. These make a grid that displays the styles image and the name of each of the styles. In this section, you are only allowed to select one style. Then once you have selected that style the program will process the image from the style that you selected. At the moment the GUI has a limit of 5 styles to pick from when they are all made.

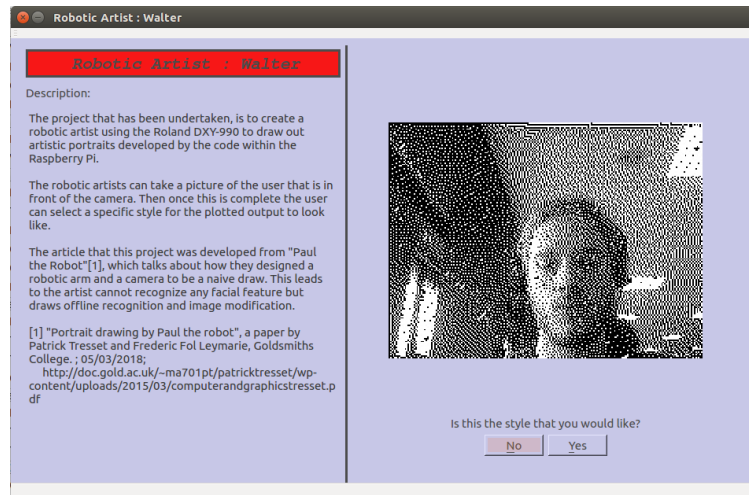


Figure 10: Page 4: Style acceptance. This page has a label that contains the stylized image and then a button at the end of the page that allows the user to accept or reject the style.

## 5 Versions

Version	Description	Date Modified
0.1	created basic structure for introduction and version control. Added sequence diagrams and class diagram.	23/04/2018
0.2	wrote the content for the UML diagram. As well, added to the section to the GUI design section.	25/04/2018