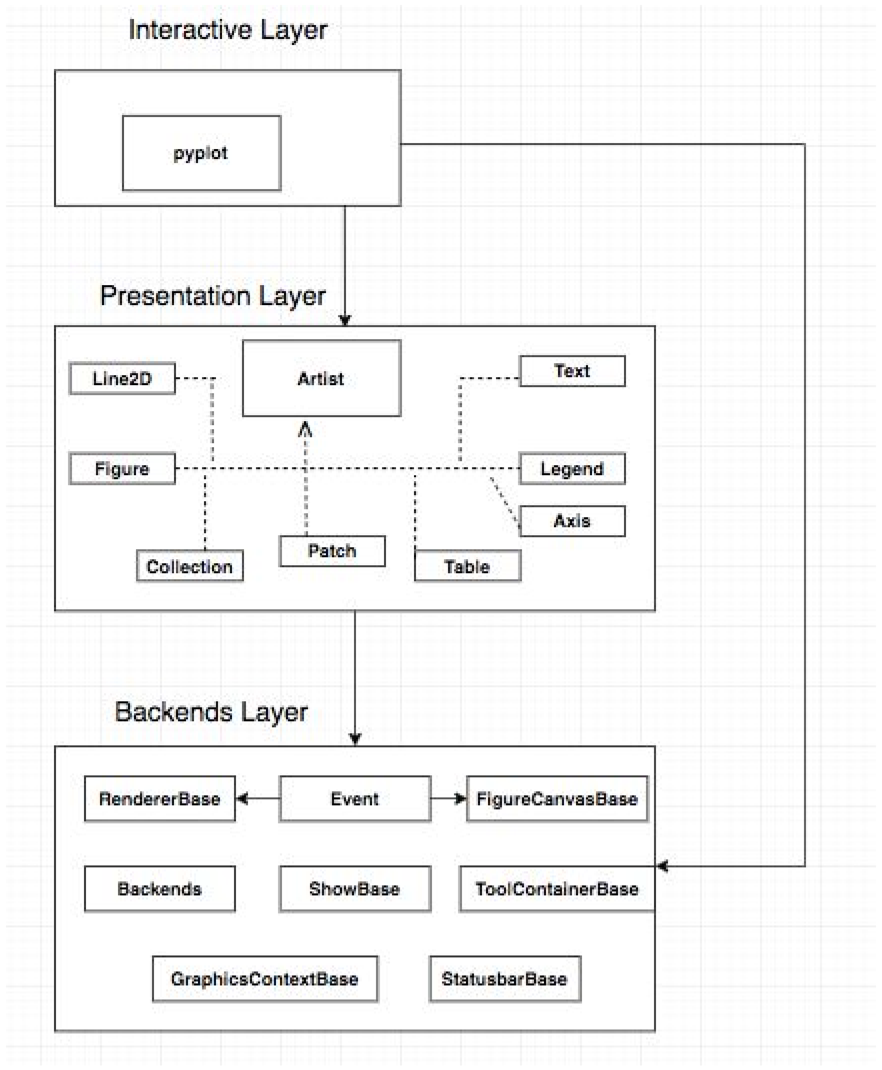
The Quick and the Studious

Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu

Overall Architecture

Matplotlib

Matplotlib is a 3-tier architecture, meaning that it consists of layers of computation (i.e. presentation layer, interactive layer, backend layer). These layers provide the scalability advantage because each layer can be scaled independently (since each layer is separated). This independence means that there is a low degree of *coupling* between layers. Below is a high-level overview of Matplotlib:



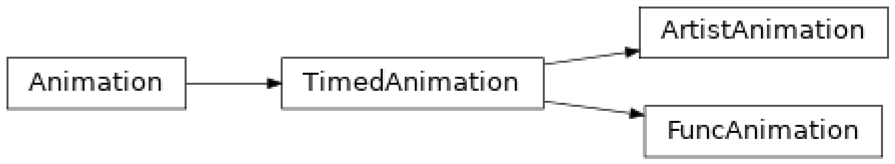
The Quick and the Studious

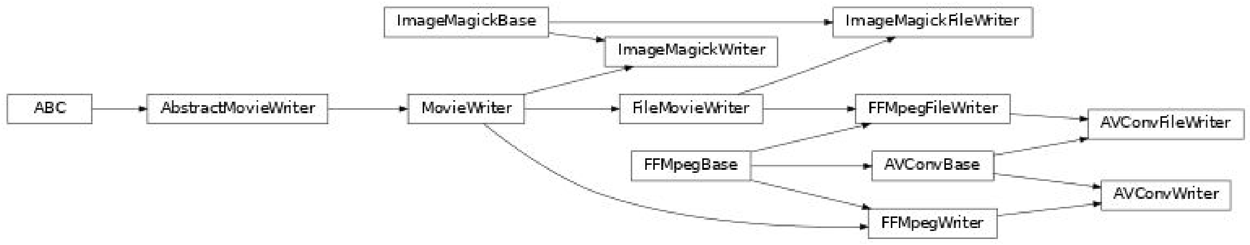
Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu

Below is a more detailed overview of Matplotlib:

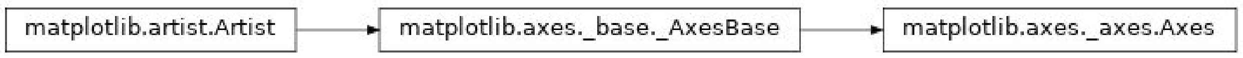
Important classes

- matplotlib.RcParams
- matplotlib.afm.AFM → object
- matplotlib.afm.CharMetrics → tuple
- matplotlib.afm.CompositePart → tuple
- Matplotlib.animation

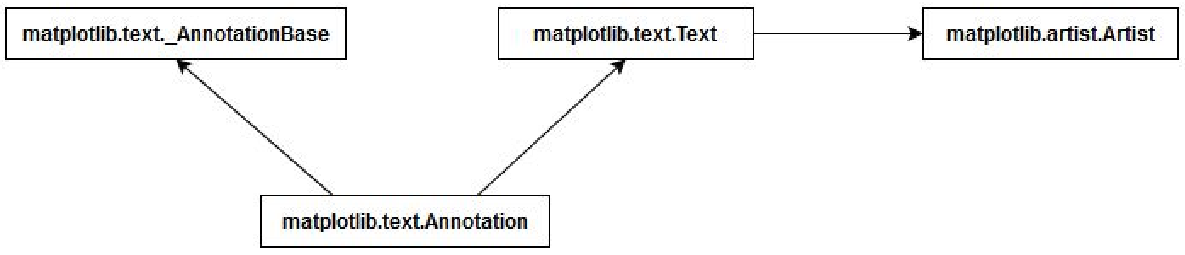




- matplotlib.artist.Artist, abstract, design at end of doc
- matplotlib.axes.Axes → matplotlib.axes._base._AxesBase



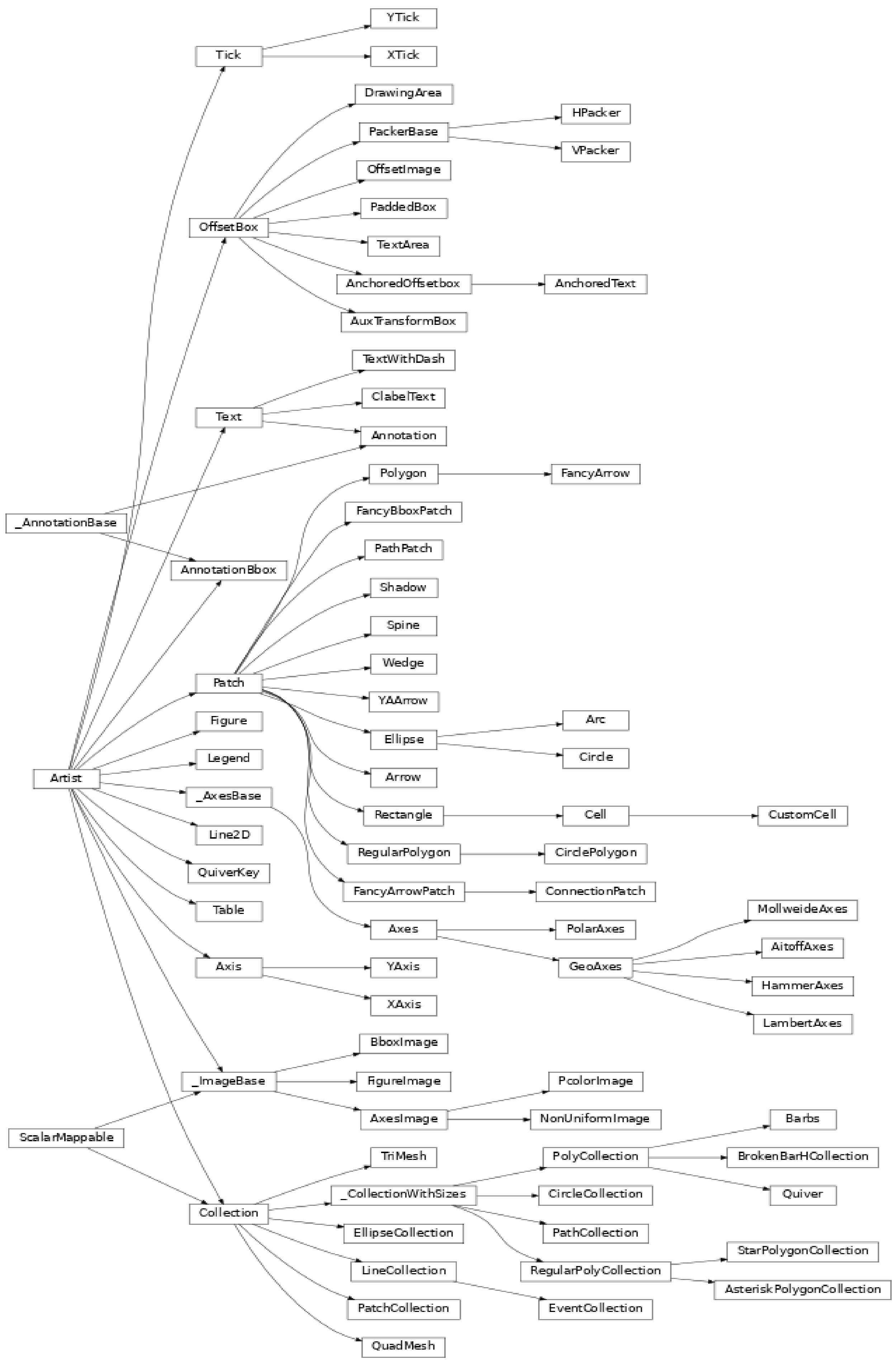
- matplotlib.axis.Axis + 3 more: XAxis, YAxis, Ticker
- Matplotlib.backend_bases: just has abstract base classes, no uml but can just use generated uml to see stuff



- matplotlib.text.Text is the super class that inherits from Artist used to draw text on a plot
- matplotlib.text.Annotation is one example of how you would use create an annotation (at position xy) with text

The Quick and the Studious

Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu



The Quick and the Studious

Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu

External libraries used

Matplotlib uses the NumPy library to manage the numbers and to perform calculations like converting between polar and cartesian coordinates (seen in /projections/polar.py). For example, it uses the ndarray (n-dimensional array) in NumPy to create and manage the data in a plot as can be seen in Matplotlib.pyplot, Matplotlib.transform, Matplotlib.axis. It also uses NumPy.arange() often to create evenly spaced intervals which is particularly useful for creating evenly spaced tick marks on a graph's axis.

Matplotlib uses the layered architecture pattern. Below are the 3 layers that make up Matplotlib.

Presentation Layer

The presentation layer is represented by the Artist class. This class processes data and manages all the drawing, styling, and plotting via an object-oriented API. In general, all visible elements that can be seen on a plot is a subclass of Artist. The Artist class itself is an abstract class and contains most of the functionalities that come with any visible object created by Matplotlib. All subclasses of Artist knows how to use a Renderer to draw onto the Canvas, the Renderer then knows how to draw on the Canvas, and the Canvas is where things are drawn.

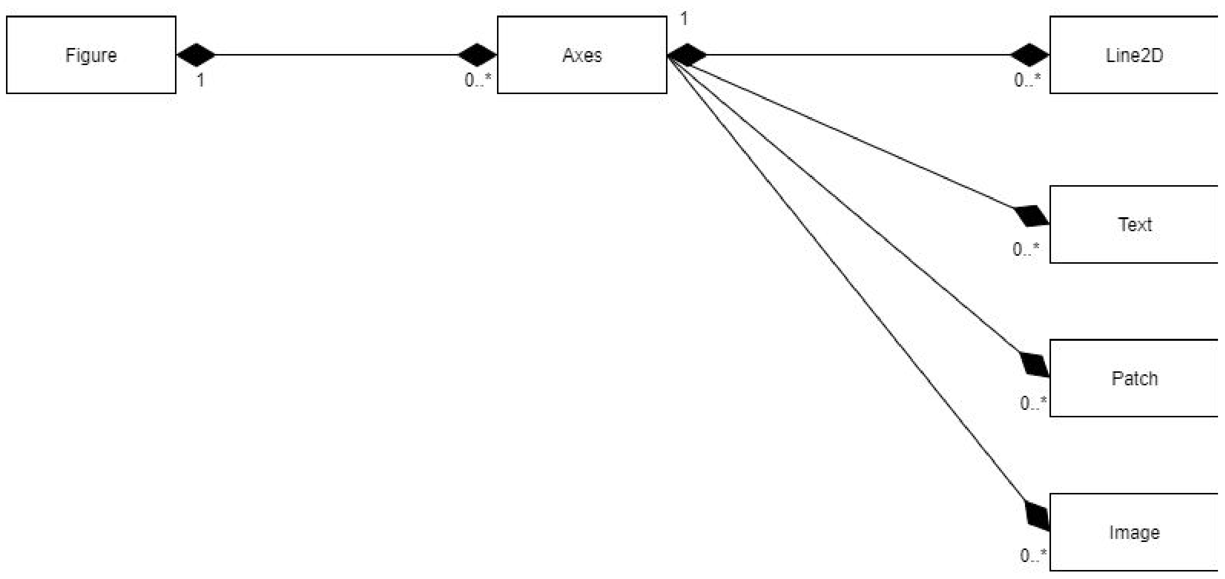
Artist objects can be grouped into two main types, primitives and containers. A primitive type Artist are objects that we actually draw such as the lines (Line2D class), rectangles (Rectangle class), and text (Text class). A container type Artist is the area where we draw other objects into such as the x and y axis of a graph (Axes class) and figures (Figure class). Typically, a user will not directly create an Artist object. Instead, it is the Axes object which acts as a wrapper for plotting functions (ex. plot(), text(), hist(), imshow()) and will return the common primitives (ex. Line2D, Text, Rectangle, Image). Once an instance of an Artist object has been created, a user can freely modify its properties. This is usually done by directly modifying the primitive itself (i.e. by changing its color and width), or by changing the property of the container (i.e. changing the axis scale from linear to logarithmic).

For instance, a user would create a Figure through pyplot's APIs and proceed to add subplots to that figure by defining axis in the figure, the number of columns or rows in the figure, or by drawing lines in the figure. The relationship of the Figure object to other Artist objects can be seen in the UML diagram below. Every class shown on the diagram is a subclass of Artist. Recall that a Figure is a container type of Artist object where you can draw other Artist objects into. In this case, a user would define the axes of this figure and then proceed to add in lines, text, or images as wanted. A Figure is then just an aggregation of Axes, which itself is a container that is responsible for determining the shape, background, and border of the items within. Some of these items as seen in the diagram below are lines, text, patch, and images.

It is worth noting that the classes in this layer have a high level of cohesion. Each object has a clear purpose and does only what it was designed to do. This design has the advantage that each object is highly reusable and that changing any one component requires relatively few changes in others.

The Quick and the Studious

Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu



Link to matplotlib documentation on Artist class
https://matplotlib.org/3.1.3/tutorials/intermediate/artists.html

Interactive Layer

The interactive layer is constituted largely by the PyPlot module. The PyPlot module in matplotlib is an interactive command-line style way of drawing plots. This is the ideal module to use to draw simple plots programmatically. However, this is achieved by abstracting away a lot of the Presentation Layer which means that it is not as flexible as the object-oriented API provided by the Artist class.

Similar to what a user would do in the object-oriented API, the PyPlot module provides simplified static methods for users to create a figure with the pyplot.plot() function. A user would then modify that plot by subsequently calling other static methods such as pyplot.axis() to change the numbers displayed on the xy-axis, or pyplot.xlabel() to change the x-axis label in the plot. It is worth noting that this module is not recommended for complex plots.

Link to matplotlib documentation on the PyPlot module
https://matplotlib.org/3.1.3/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py

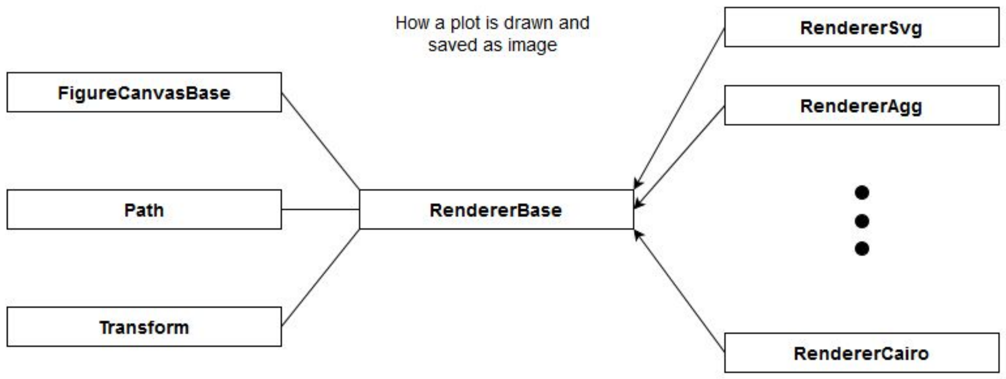
Backend Layer

The backend layer is composed of everything in the matplotlib/backends directory, the backend_bases, backend_tools, and backend_managers modules. This layer contains all the tools necessary for matplotlib to render an image, save it to a file, or embedded it into another application.

The RendererBase is a base class that defines all the functions for how an image is to be drawn and displayed on the screen. Several of the Renderer classes that inherit from this base class have to do with the image's specific format (e.g. RendererSvg, RendererAgg, RendererCairo). The diagram below shows an example of how the Renderer classes interact with the Artist class to create an image.

The Quick and the Studious

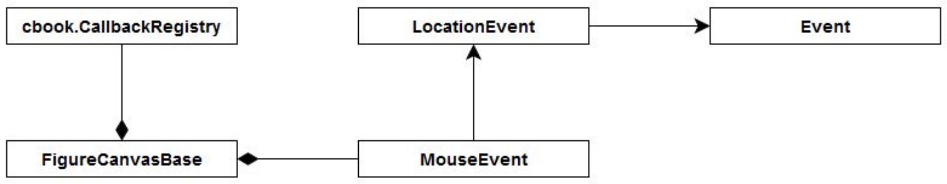
Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu



The Renderer takes a Path object that denotes the line that is to be drawn on the canvas, a Transform object which represents the geometric transformation that is used to determine the final position of all elements drawn on the canvas, and a FigureCanvas object which is the canvas to which a Figure object would be rendered into.

In addition to writing images to files, the backend also enables matplotlib to be embedded into various other applications. For example, the backends_gtk3 module defines all methods that utilizes the GTK rendering engine to render images based on the Artist objects.

Another major component to the backend is the Event class. In general, an event is an object that is created when a user interacts with matplotlib. Once an event has been created, there will be an appropriate event handler which in most cases is a callback that is there to handle the event. For instance, as seen in the UML diagram below. When a user clicks on the screen, a MouseEvent object will be created in the FigureCanvas class. The FigureCanvas class will first look up the callback function associated with the event that is stored in a CallbackRegistry object. Then, it will attempt to process the event based on the parameters generated by the mouse click (i.e. the location that was clicked and the mouse button that was clicked). In addition to pre-defined events that comes with matplotlib, users can also create their own event and connect it to their generated plot.



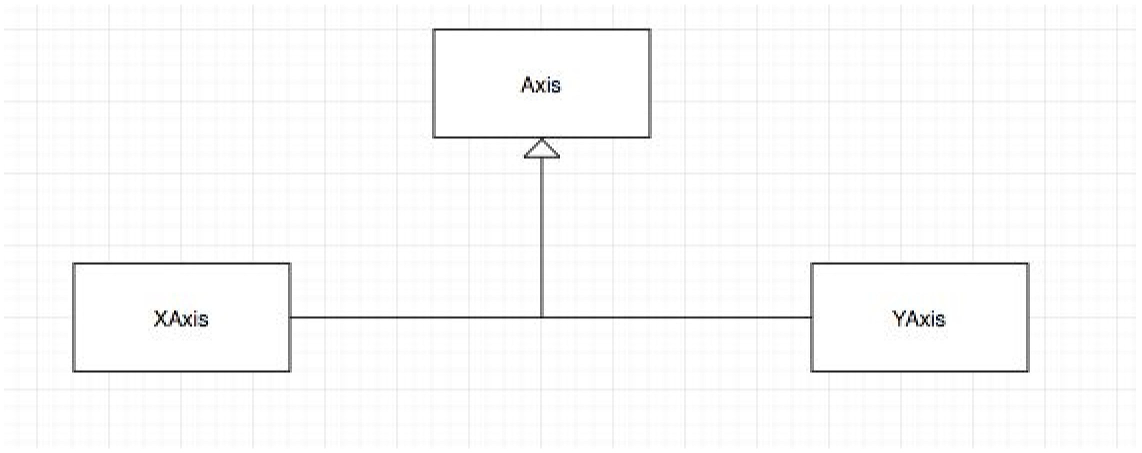
Link to matplotlib documentation on backend_bases and event handling
https://matplotlib.org/3.1.1/api/backend_bases_api.html
https://matplotlib.org/3.1.1/users/event_handling.html

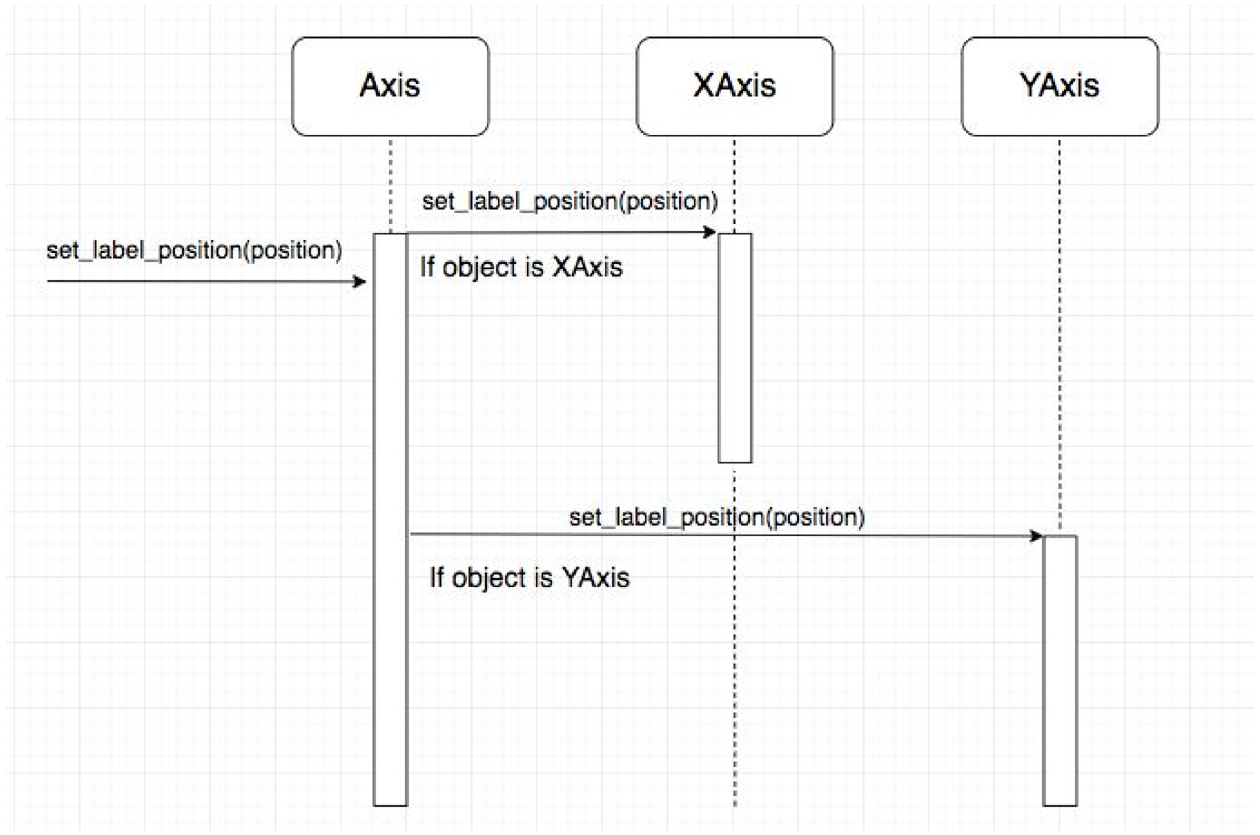
The Quick and the Studious

Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu

<u>Design Patterns</u>

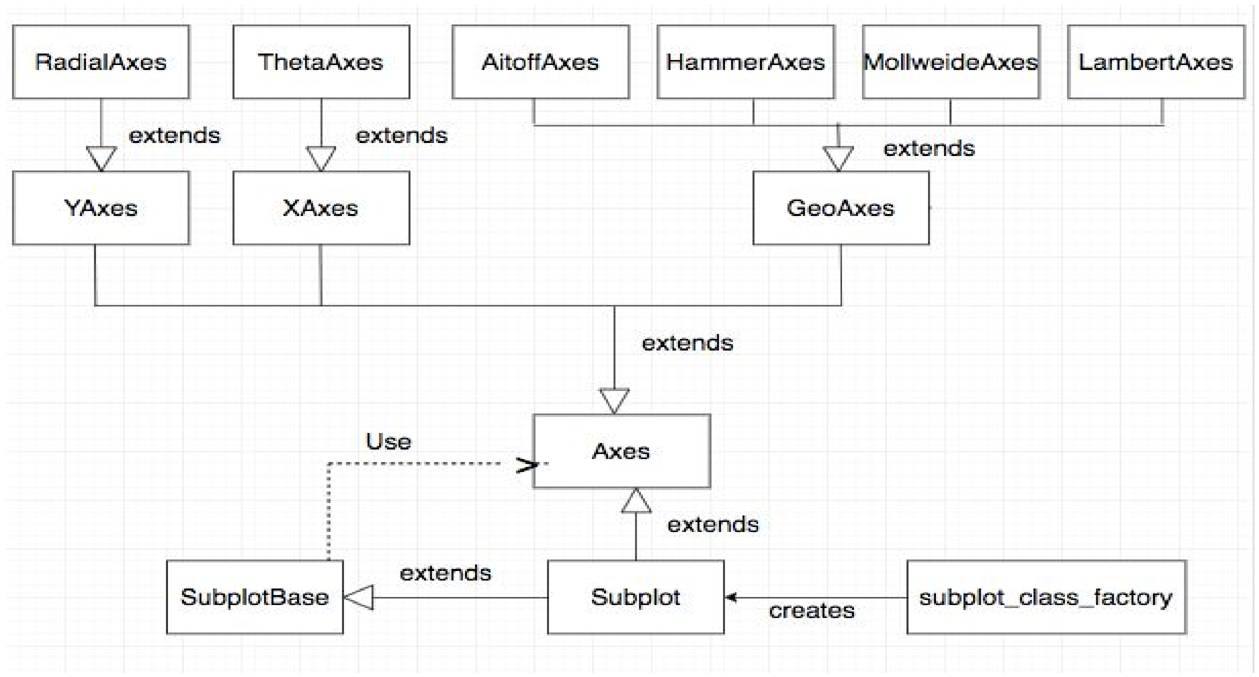
1) **Strategy Design Pattern**



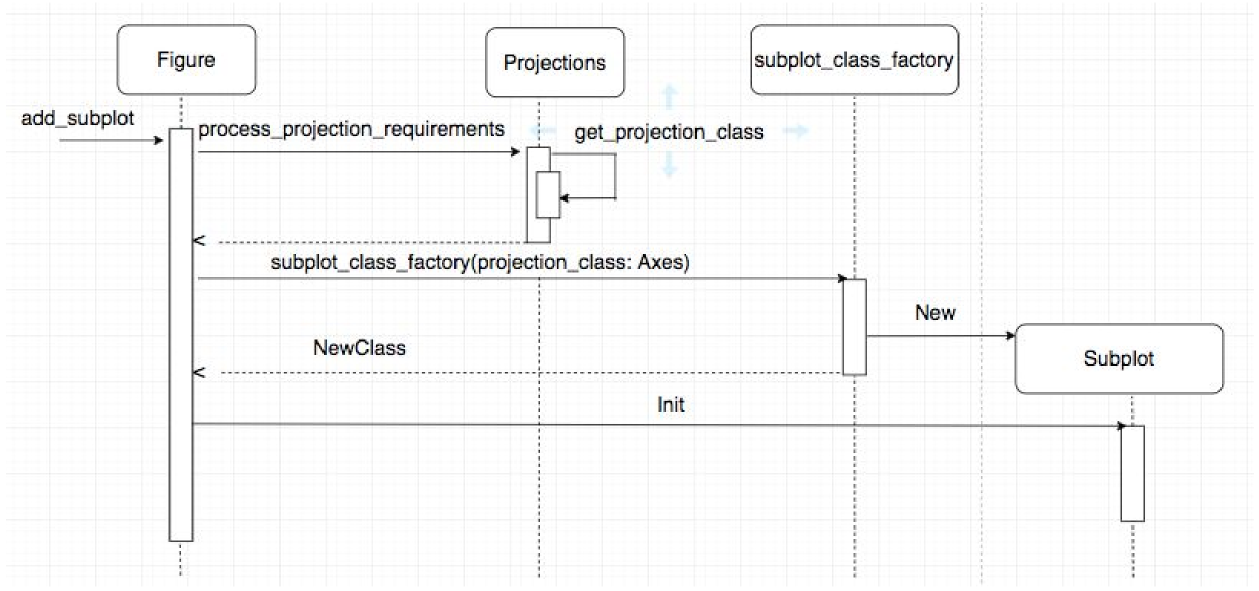

Axis
XAxis
YAxis
set_label_position(position)
set_label_position(position)
If object is XAxis
set_label_position(position)
If object is YAxis

When set_label_position is called on an XAxis object, the set_label_position of XAxis is called (similar logic for YAxis). There are many other examples.

2) **Factory Design Pattern**



subplot_class_factory takes in an Axes and creates a new class that inherits from SubplotBase and the given axes_class (which is assumed to be a subclass of Axes). This allows a Subplot of any type of axes to be created without having to create a new Subplot class for every Axes. There are many types of Axes and each have their own transform algorithms (eg. some invertible, some not), which will make them look different on the subplots. Once a Subplot is created, it is returned so that the calling function can instantiate it.



The Quick and the Studious

Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu

Software Development Process

Waterfall Process

For our project, we are opting to use the waterfall development process. The waterfall method emphasizes a linear one direction development approach. Using a series of logical progression of steps in the Software Development Life Cycle. One step must be finished before moving on to the next step, this is fine as we will not have changing requirements. There are in general 6 steps in the waterfall methodology: Requirements, Analysis, Design, Coding, Testing, Operations. However, we will not be using all of these steps.

- Requirements: We don't need to define our requirements because it should already be given to us in the issues list. As we are fixing bugs, the requirement would simply be to resolve the issue. Furthermore, the requirements won't change throughout the development process. However, for adding new features, we will define requirements.
- Analysis: In this step, we will understand the structure of the application and where the issue comes from. We will use UML diagrams and reverse engineering tools to help us understand the code source. For adding features, we will analyze the structure to decide on practical features.
- Design: Once the source of the issue is identified, we will start designing how to approach and resolve the issue. This is mainly planning our code before the concrete implementation.
- Coding: In this step we will start the implementation based on the previous two steps. The coding will also be done in a repeating loop with the next step testing, until the bugs are fixed, features are working, and the code behaves properly.
- Testing: After some iteration of the code is completed, we will perform testing to verify if its working as expected. We will go back and forth between coding and testing until everything works.
- Maintenance: We will not need this step as after we fix the bug, our part of the project is considered done. We haven't decided on how the feature will be maintained. The maintenance will be left to the owner of the original project.

The Quick and the Studious

Chris Ling, Faris Ally, Harrison Fok, Vili Milner, Winston Zhu

Why we chose Waterfall

There are several reasons why we chose to follow the Waterfall process. First, we realized that since there is a list of issues that need to be fixed, the requirements are already set for us, and they are not likely to change while we are developing. An Agile process is more effective when requirements are uncertain and likely to change, but since the requirements are set, we do not anticipate any changes, and feel that Waterfall would be a more effective process. In addition to this, the Waterfall model is ideal for us as we would be able to have a design phase which will allow us to figure out how we can implement features and bug fixes. We thought that a planning phase would be critical for this project as it will allow us as a team to identify what parts of the code that would need to be changed and how our changes might affect the project as a whole. Since we are fairly new to this software, we will need some time to identify exactly how to fix bugs and design features, as we will need to thoroughly examine the section of the code related to the bug or feature. This planning time lets us as a team identify any dependencies in the code, any problems we might run into, and how we can most effectively work on the code. This will allow us as a team to effectively divide work, since we will have a good idea of how difficult each task is, and we will gain a better understanding of the project. Lastly, the team would like to experiment with a new development process, as we have all used Agile in the past, we felt it would be a good idea to get experience with a new software development process that is very commonly used in the industry.

Why not Agile?

Initially, our team wanted to use an Agile development process. Upon further discussion and consideration, we thought that it might be better for us to use Waterfall. In addition to the explanation above regarding why Waterfall is the best process for us, there are several reasons why an Agile process would not suit our needs for this project. First of all, An Agile process is most effective when a team can work together in the same space and devote most of their time to the project, so that meetings can occur and team members can interact with one another. Due to the nature of us being students, we will rarely have time for actual face-to-face interaction, and most interaction will be done online. Due to this, an Agile process would not have been as effective in this situation. Waterfall will allow us to work effectively while not having to meet as frequently, while still having the whole team be on top of what is going on. Also, using an Agile process requires the timeline to be more flexible, as naturally, Agile has the team prioritize working increments of software rather than setting strict deadlines. For this project, our deadlines for our deliverables are very strict, and we need to be able to ensure that we have working code by the deadlines. Using Waterfall will ensure that we can plan successfully to meet our deadlines. Lastly, Agile is most effective when we can communicate with the customer and get feedback regularly. Since we are working on an open-source project, it is uncertain how often we will get feedback from the product owners (if any at all). Since there are preset requirements and no guarantee of customer feedback, Agile would not be effective here. For these reasons, we decided that Waterfall would be superior to Agile for this project.