Matplotlib

Deliverable 5: Feature Implementation

By: Team Legend (Team 4)

Feature 8283: Custom Ordering of Legend

Section of Code Modified

- Class Artist (matplotlib/lib/matplotlib/artist.py)
  - Line 110
  - Line 915-933
  - Line 985
- Class Container (matplotlib/lib/matplotlib/container.py)
  - Line 21
  - Line 31
  - Line 70-87
- Class Axes (matplotlib/lib/matplotlib/axes/_axes.py)
  - Line 259-266
  - Line 276

Feature Documentation

User Guide

Custom ordering of legend entries using `lorder` parameter in `plot()` method

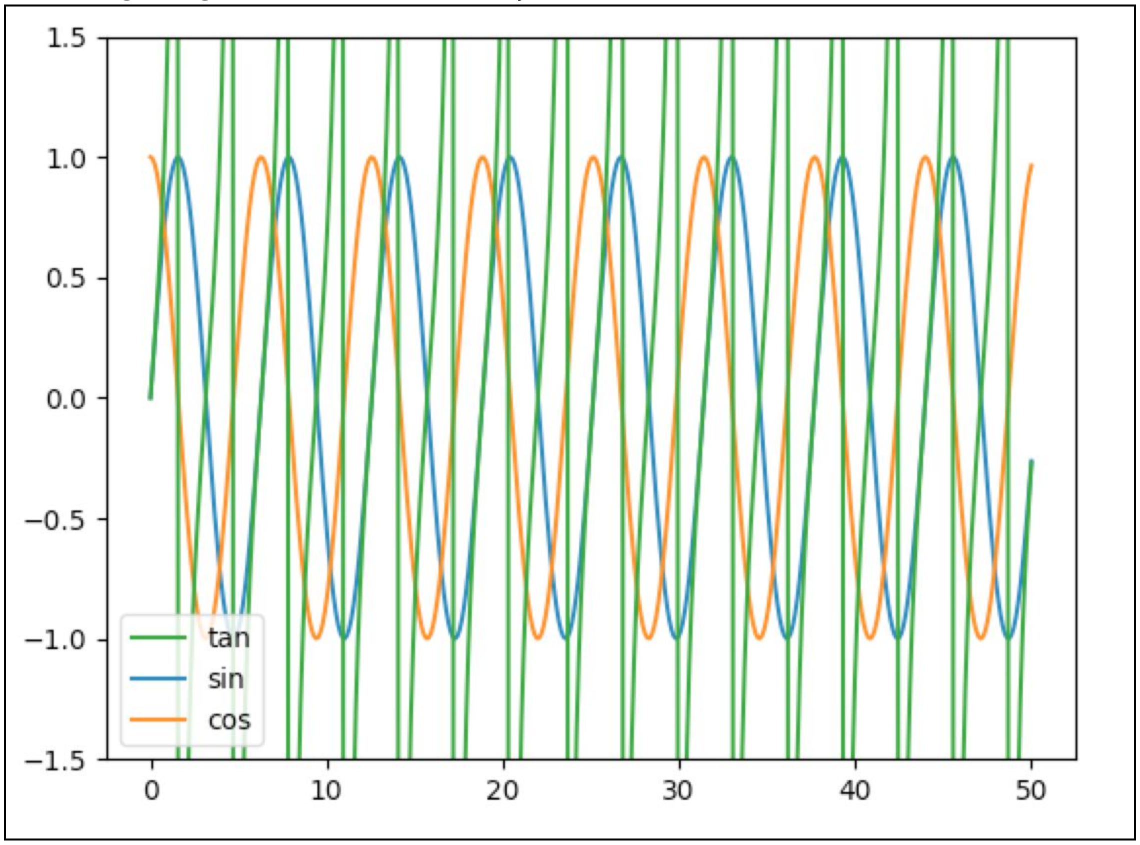
Legend label: The text which describes the handle represented by the key.
Legend lorder: A positive integer that is assigned as ordering priority to the legend entries

When plotting without using the `lorder` parameter the ordering of the legend entries will be random. However, by using lorder legend entries can be ordered manually. To do so assign the priority when initiating the figure as show in the example below.

```
x = np.linspace (0,50,400)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)

pyplot.plot(x, y1, lorder=2, label="sin")
pyplot.plot(x, y2, lorder=3, label="cos")
pyplot.plot(x, y3, lorder=1, label="tan")
pyplot.legend()
pyplot.ylim(-1.5, 1.5)
pyplot.show()
```

The resulting image has `tan` as first entry followed as `sin` and then `cos.`



The value of the `lorder` are relative therefore it can be assign any positive integer. For instance the below code will produce as result.

```
pyplot.plot(x, y1, lorder=200, label="sin")
pyplot.plot(x, y2, lorder=59999, label="cos")
pyplot.plot(x, y3, lorder=1, label="tan")
```

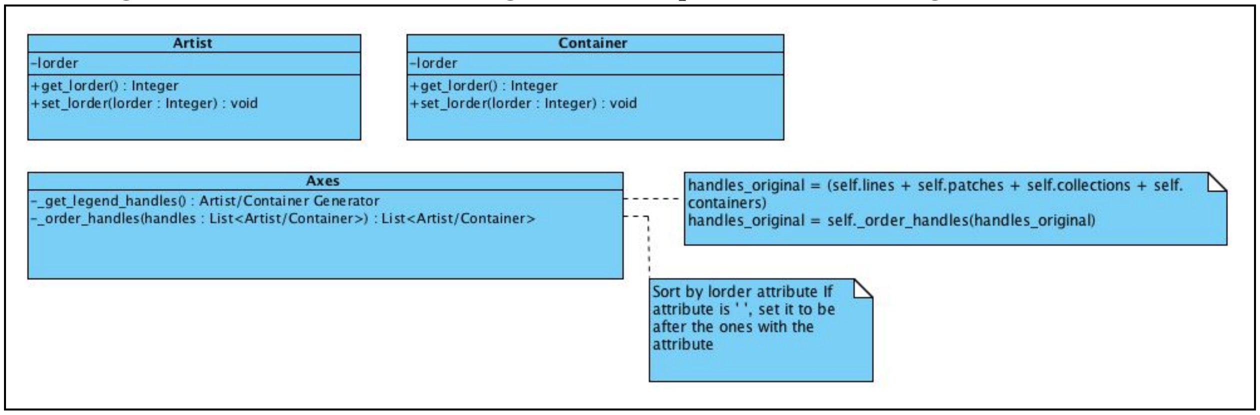
Design Documentation

To implement Feature 8283, we have added an additional attribute `lorder` in both the Artist class (located in artist.py) and the Container class (located in container.py). This attribute is an Integer that represent the order of an Artist/Container object in the legend. In the Axes class (located in _axes.py), we have added a function `_order_handles(List <Artist/Container> handles)` that sorts a list of Artist/Container and then returns the list. We have modified function `_get_legend_handles()` to call `_order_handles` that sorts `handles_orginal` (a List <Artist/Container>) according to the value of `lorder` in each element. If the value of `lorder` in every element of handles_original is `‘’`, then `_order_handles` does not make any changes to `handles_original`.

With our feature implemented into the Matplotlib code base, users still have the option to create Artist/Container objects and to order entries in the legend the same way they did before. In other words, our feature would not break any previously written code. For the creation of an Artist/Container object, the user now have the choice to input a value for the lorder attribute. The modification we have made in function _get_legend_handles will modify the default entry order by sorting them according to the `lorder` attribute.

UML Diagram

The diagram below is the UML diagram that represents our design.


Artist
-lorder
+get_lorder() : Integer
+set_lorder(lorder : Integer) : void
Container
-lorder
+get_lorder() : Integer
+set_lorder(lorder : Integer) : void
Axes
-_get_legend_handles() : Artist/Container Generator
-_order_handles(handles : List<Artist/Container>) : List<Artist/Container>
handles_original = (self.lines + self.patches + self.collections + self.containers)
handles_original = self._order_handles(handles_original)
Sort by lorder attribute If attribute is ' ', set it to be after the ones with the attribute

Changes from Deliverable 4

In Deliverable 5, our team have generally followed the design of this feature in Deliverable 4. However there are two minor changes. Originally, we default value for attribute `lorder` was `None`. In Deliverable 5, we change changed `None` to the empty string `‘’`. Another minor change we made is in the method _get_legend_handles(), originally, we planned to add the line:

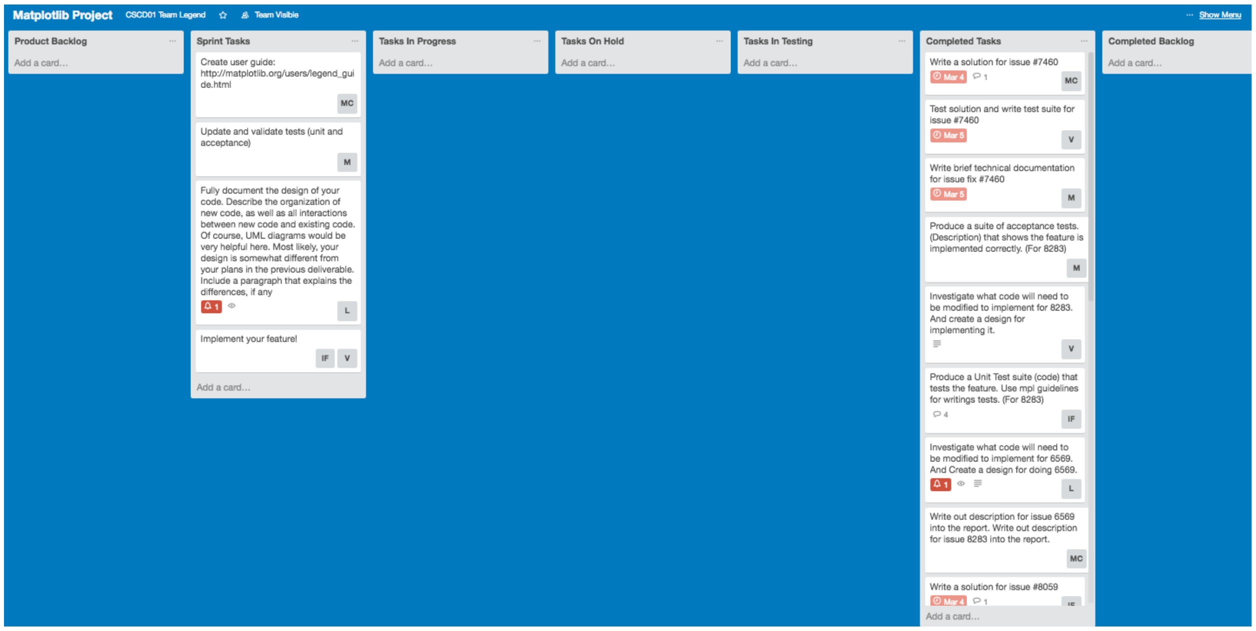
- `handles_origina l= _order_handles(handles_original)`

In Deliverable 5, this line was changed to

- `handles_origina l= self._order_handles(handles_original)`

Development Process - Screenshots of Taskboard

The diagram below is the screenshot of our Trello Taskboard at the beginning of this sprint



Matplotlib Project · CSCD01 Team Legend · Team Visible · Show Menu

Product Backlog

Add a card…

Sprint Tasks

Create user guide: http://matplotlib.org/users/legend_guide.html — MC

Update and validate tests (unit and acceptance) — M

Fully document the design of your code. Describe the organization of new code, as well as all interactions between new code and existing code. Of course, UML diagrams would be very helpful here. Most likely, your design is somewhat different from your plans in the previous deliverable. Include a paragraph that explains the differences, if any — 1 — L

Implement your feature! — IF V

Add a card…

Tasks In Progress

Add a card…

Tasks On Hold

Add a card…

Tasks In Testing

Add a card…

Completed Tasks

Write a solution for issue #7460 — Mar 4 — 1 — MC

Test solution and write test suite for issue #7460 — Mar 5 — V

Write brief technical documentation for issue fix #7460 — Mar 5 — M

Produce a suite of acceptance tests. (Description) that shows the feature is implemented correctly. (For 8283) — M

Investigate what code will need to be modified to implement for 8283. And create a design for implementing it. — V

Produce a Unit Test suite (code) that tests the feature. Use mpl guidelines for writings tests. (For 8283) — 4 — IF

Investigate what code will need to be modified to implement for 6569. And Create a design for doing 6569. — 1 — L

Write out description for issue 6569 into the report. Write out description for issue 8283 into the report. — MC

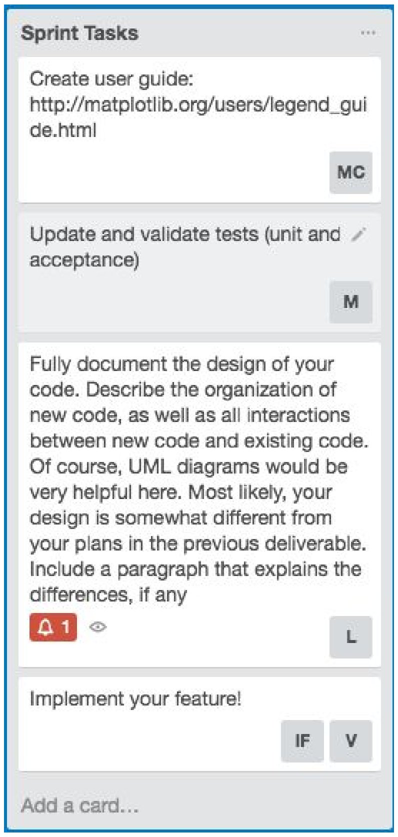
Write a solution for issue #8059 — Mar 4 — 1

Add a card…

Completed Backlog

Add a card…

Detailed diagram for Sprint Tasks:



Sprint Tasks

Create user guide: http://matplotlib.org/users/legend_guide.html — MC

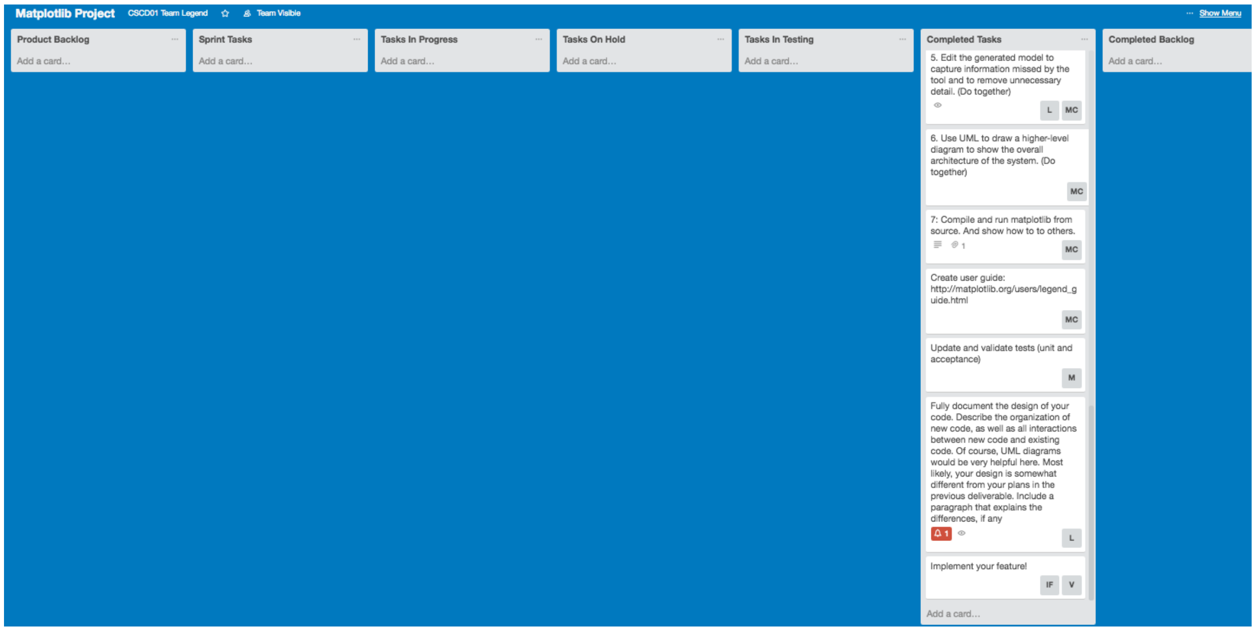
Update and validate tests (unit and acceptance) — M

Fully document the design of your code. Describe the organization of new code, as well as all interactions between new code and existing code. Of course, UML diagrams would be very helpful here. Most likely, your design is somewhat different from your plans in the previous deliverable. Include a paragraph that explains the differences, if any — 1 — L

Implement your feature! — IF V

Add a card…

The diagram below is the screenshot of our Trello Taskboard at the end of this sprint.



5. Edit the generated model to capture information missed by the tool and to remove unnecessary detail. (Do together) — L, MC

6. Use UML to draw a higher-level diagram to show the overall architecture of the system. (Do together) — MC

7: Compile and run matplotlib from source. And show how to to others. — 1 — MC

Create user guide: http://matplotlib.org/users/legend_guide.html — MC

Update and validate tests (unit and acceptance) — M

Fully document the design of your code. Describe the organization of new code, as well as all interactions between new code and existing code. Of course, UML diagrams would be very helpful here. Most likely, your design is somewhat different from your plans in the previous deliverable. Include a paragraph that explains the differences, if any — 1 — L

Implement your feature! — IF, V

Add a card…