Issue #16796

Add Checkboxes to Legend Controlling Subplot Visibility
https://github.com/matplotlib/matplotlib/issues/16796

Why we changed our implementation plan

We changed our implementation because after showing our idea to the matplotlib developers, they asked us to implement the feature as an example instead of as part of the API. Through further design choices, we have decided to implement it as a separate layer with its own function for creating an interactive legend. This module would take all of the specified arguments for the legend, and then generate the legend using a handler class, where the handler class is our custom designed one to account for the interactivity of the checkboxes.

How it is different for users

The only difference this makes to the user is that they will have to download our example and run the function in it to use the interactive legend instead of just an additional argument in the legend object's init function.

Feature Description

The feature is being able to initialize the legend with the ability to add interactive checkboxes that turn the visibility of the subplots on/off. Users can add the interactive checkboxes to the legend by directly calling the method interactiveLegend and passing in the required parameters for generating the legend of the graph. Then the method would utilize a custom handler class that will generate the legend as well as its interactive elements.

Implementation

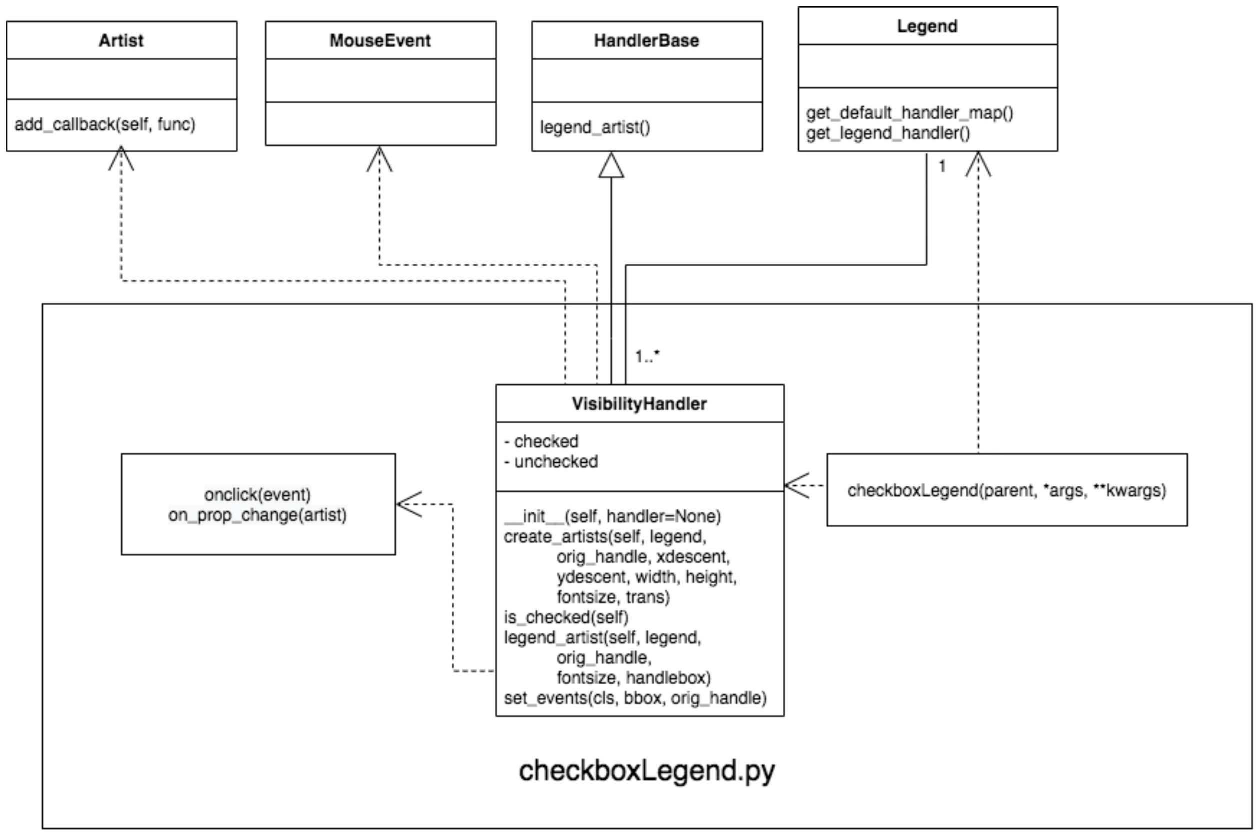
Code Organization

- There will be a new file that will be created. It represents a new module that will make the legend interactive by making the checkboxes clickable.
    - class VisibilityHandler
        - Two private attributes, which are the file paths of the checkbox and the unchecked box images
        - def __init__(self, handler=None)
            - The method for initializing the visibility handler
        - def create_artists(self, legend, orig_handle, xdescent, ydescent, width, height, fontsize, trans)
            - Create the necessary artists for the legend
        - def legend_artist(self, legend, orig_handle, fontsize, handlebox)
            - Returns the artist that the handler generates.
        - def is_checked(self)
            - Returns true if checked, false otherwise
        - def set_events(cls, bbox, orig_handle)
            - def onclick(event)
                - The event that toggles the visibility and the appearance of the checkbox (checked vs unchecked).
            - def on_prop_change(artist)
                - This method will match the checkbox status to visibility. For instance, if the checkbox is unchecked or checked, then the visibility will be updated accordingly.
    - def checkboxLegend(parent, *args, **kwargs)
        - This will be the global function that users can call and pass in parameters to create an interactive legend for their graph, where they are able to control the visibility of certain plots.

Requirements

- Learn how to add an image.
- Learn how to make a dynamic image so it responds on button press
- Learn how to use intricacies in using a handler to add 2 elements cleanly

Diagram


Artist
add_callback(self, func)
MouseEvent
HandlerBase
legend_artist()
Legend
get_default_handler_map()
get_legend_handler()
1
1..*
VisibilityHandler
- checked
- unchecked
__init__(self, handler=None)
create_artists(self, legend, orig_handle, xdescent, ydescent, width, height, fontsize, trans)
is_checked(self)
legend_artist(self, legend, orig_handle, fontsize, handlebox)
set_events(cls, bbox, orig_handle)
onclick(event)
on_prop_change(artist)
checkboxLegend(parent, *args, **kwargs)
checkboxLegend.py