Matplotlib coding

• EASY: On the pyplot subplots method, add a keyword argument duplicate_labels=False, and when sharex or sharey is True, turn off redundant labels

• MEDIUM: make a function "clickable" for the legend which builds the features of examples/event_handling/legend_picking.py into the legend itself

• MED HARD: write a basic 1D plotting routine. The trick here will be to generate the mesh over x on which to evaluate the function.

• HARD: Add support for a pan/zoom tool in the html5 backend

• VERY HARD: Edit the image functionality to work with an array of uint8 rather than float64, including color mapping and normalization. At the end of the plotting routine, the internal 64 bit floating point is converted to uint8 before rendering, so if we could keep this leaner representation throughout, it would vastly improve mpl's memory when working with images.

Matplotlib examples

- MEDIUM EASY: Follow the patterns in the glass dots demo but instead of using random dots, take an image and find the x,y locations where the image luminosity is above some threshol. Plot the random dots of these pixels, which will be a thresholded black and white approximation of the grayscale image. Apply small translations and stretchings to see if you can generate stereoscopic (3D) effects. Contribute the example to the matplotlib repository.

Matplotlib docs and testing

•MEDIUM EASY: Write a very explicit tutorial for a total beginner on how to add a new unit test, one with image comparisons and one without, and add it to the developer docs.

•EASY: Read through threads on matplotlib users, find a good question with answer, and convert it into a FAQ for the doc/faq/howto_faq.rst or whatever is most appropriate

•EASY: If you have a favorite trick or idiom that you use frequently, contribute a section to docs/users/recipes.rst

•MEDIUM EASY: Add a unit test to the matplotlib testing framework. This can either be a test on a non-image generating part of the code, or a baseline image comparison test.