

In this lab task, you will extend the image viewer app from the previous lab. Some functionalities will be added so that you can also modify the images. The following are descriptions of the specific tasks this time:

- Separate the original `select_file` into two functions, one for reading the image file and the other for displaying the image. Name the latter one `show_image`. Call `show_image` from `select_file`. Reason: We do not want to reload the image file every time we adjust the image. Keep the `im0` property as the original (unmodified) image.
- **Image color adjustments:**
 - ♦ We will let you play with color gamma adjustments: $c \leftarrow c^r \ (r > 0)$
Here c represents the value of a pixel in $[0,1]$, and r is a positive constant. We will use three different values of r , one for each color channel. Use `im2double` when reading the image file to scale the pixel values to the range of 0~1 first.
 - ♦ Add three `sliders` used for the selection of the three r values. Set the `Limits` property of the `sliders` to `[-2 2]`. You can query or change the current value of a `slider` using the `Value` property.
 - ♦ For each `slider`, add its `ValueChanged` callback function. Call `show_image` from the callback.
 - ♦ Within `show_image`, create a separate array `im` to represent the image to be displayed, so that `im0` remains unchanged. Modify the three color channels separately. For each color channel, use $r = 2^{-v}$, where v is the `Value` property of the corresponding `slider`.
 - ♦ Add a `pushbutton` for resetting the color. Within its callback, just set the `Value` of all the `sliders` to `1`, and then call `show_image`.
- **Scrollable container for the image:**
 - ♦ Add a `panel` container object in your app, and then insert an `axes` within the `panel`. Modify your `show_image` function to draw to this `axes`. This allows scrolled viewing of the image, so that we can see a large image in its original resolution.
 - ♦ Set the `Scrollable` property of the `panel` to `true`, and the `AutoResizeChildren` property to `false`, so that you can have exact control of the image size shown in the `panel`.
 - ♦ Within `show_image`, set the `Position` property of the `axes` to `[0 0 w h]`, where `w` and `h` are the width and height of `im`. If the `axes` (and the image itself) is larger than the `panel`, scroll bars will appear automatically.
- **Initialization:**
 - ♦ Add the callback for the `Startup` event of the main figure for initialization. This callback allows you to set some properties/parameters before the user can do anything.
 - ♦ We want to prevent the user from doing something that causes errors because the program is not ready. In this app, functionalities for image adjustments should be enabled only after an image is selected. So you should set the `Enable` property of those objects to `false` (or `0`, or `'off'`) in the initialization code. You can set the property to `true` (or `1`, or `'on'`) when an image is selected.
 - ♦ You should also set the `slider` values to their default (`1`) whenever a new image file is selected.
- **Optional experiments:**
 - ♦ The `ValueChanged` event of a `slider` is triggered only after you have finished dragging the `slider` handle (i.e., when you release your mouse button). If you want a more instant feedback on the adjusted image, try the `ValueChanging` event, which is triggered each time you move the `slider` handle. The `Value` property of the `slider` is not changed when you're still dragging the `slider` handle. Instead, you get the "temporary value" from the `event` argument passed to the callback. To use this temporary value for image adjustment, you need to modify `show_image`, possibly with some optional input arguments, to receive these temporary `slider` values. Try to decide how to do this yourself.