How To Optimize the User Experience of Your MATLAB Apps

This document provides some recommendations to improve the user experience of applications developed with MATLAB. Note: The content of this document should not be intended as an exhaustive list of actions to take to create an app with an optimal user experience.

1. Choose The Correct Position and Size

- ✓ `uimenu` – use as an API for loading and saving data, and to provide access to unfrequently-used app functionalities.
- ✓ `uitab`, `uitabgroup` – separate different parts of the workflow, convert multi-window apps to single-window apps, avoid concentrating all the UI components into a single panel.
- ✓ `uitoolbar` – alternative for menu, use as an API for functionalities that always need to be readily accessible.
- ✓ `uigridlayout` – effectively manage the position and resizing of all UI components inside the app. Tip: Insert grid layout objects before inserting the UI components. Note: Grid layouts can be used to implement dynamic layouts, where UI components appear and disappear within the lifetime of the app. To hide a UI component placed inside a grid layout, set the `ColumnWidth` or `RowHeight` property (depending on how the component is placed within the grid layout) of the grid layout object to `"0x"`. Then, to make it visible again, set the same property to some finite value (e.g., `"1x"`).
- ✓ `tiledlayout` – place a number of axes inside the same container (a panel, a grid layout object, a tab). Use it to dynamically insert axes inside a container with automatic fitting.

2. Set The Expectations

- ✓ Use icons, e.g., `obj.Icon = "myIcon.png";` icons provide an intuitive understanding of the functionality of the UI component.
- ✓ Use tooltips, e.g., `obj.Tooltip = "UI Component description.";` tooltips can describe the behaviour of the component without needing to act on it.
- ✓ `uialert` – warn the user about potential consequences or provide information through a dialog window.
- ✓ `uiprogressdlg` – progress bars can be determinate or indeterminate. Use to set the expectations of how long a calculation will take, if known.

3. Provide Feedback to The User

- ✓ `uiprogressdlg` – use when a calculation takes a long time to avoid a static screen.
- ✓ Logger – get feedback from users to improve the design of the app. Example: Advanced Logger for MATLAB.
- ✓ `uiconfirm` – provide positive/negative feedback through a dialog window, such as a confirmation message.

4. Anticipate User Errors

- ✓ Enable or disable interactivity with UI components when needed to avoid unintentional use.
- ✓ Hide UI components using the grid layout to avoid unintentional use.
- ✓ `try, catch` – handle unexpected behaviour in a robust manner.
- ✓ `uialert` – provide error messages through a dialog window.

5. Provide Documentation

- ✓ Live Scripts – document your code, app, or project; export to HTML to incorporate it inside an app.
- ✓ `uihtml` – container for the exported documentation.
- ✓ `uitree` – use as an index for the documentation inside the app.

6. Enhance The Appearance

- ✓ Customize colors to suit requirements and/or taste. Note: Colors also help to set expectations, e.g., green for "go" and red for "stop".
- ✓ `uihtml` – interface for implementing custom HTML-based components inside a MATLAB app. Create a custom component using the `matlab.ui.componentcontainer.ComponentContainer` class, and use **`uihtml`** to interface with an external UI component created using HTML, JavaScript® and CSS.
- ✓ Download "Customizable HTML Button" as an example of a custom UI component.

Additional Resources – Beyond User Experience

- Developing MATLAB Apps Using the Model-View-Controller Pattern
- Developing Robust MATLAB Code and Applications