Bài 5: MATLAB GUIDE

A. Thiết kế giao diện đồ họa người dùng

1. Mục tiêu

Mục tiêu của việc thiết kế giao diện đồ họa là nhằm giúp người dùng phổ thông có thể vận hành hệ thống một cách dễ dàng và hiệu quả. Một giao diện được thiết kế tốt không những cần có sự bố trí hợp lý giữa các thành phần trong chương trình mà còn phải đảm bảo các thao tác sai từ người dùng không gây ra các lỗi nghiêm trọng tới hệ thống và có các biện pháp cảnh báo một cách hợp lý.

2. Các bước thiết kế giao diện đơn giản

a. Làm rõ mục tiêu và chức năng của chương trình.

b. Phác thảo giao diện trên giấy:

- Toàn bộ giao diện có thể dùng chung trong một cửa sổ hay cần tách ra thành nhiều phần?
- Các nút ấn, hộp nhập dữ liệu,... cần được bố trí ở đâu để người dùng thao tác một cách hợp lý nhất.
- Làm sao để người dùng dễ dàng hiểu được chức năng của các thành phần trong giao diện? (Tên, nhãn, màu sắc,...)
- Làm sao để thông báo các trạng thái của hệ thống như: *đang xử lí, hoàn thành tác vụ, lỗi, chờ lệnh,...*

c. Xác định loại công cụ được dùng để thiết kế giao diện (Trong bài thực hành này ta sẽ dùng Matlab GUIDE.)

d. Xây dựng chương trình.

e. Kiểm tra và bảo đảm chương trình hoạt động đúng như yêu cầu đã đặt ra ở bước a.

B. Matlab GUIDE (Graphical User Interface Development Environment)

1. Giới thiệu

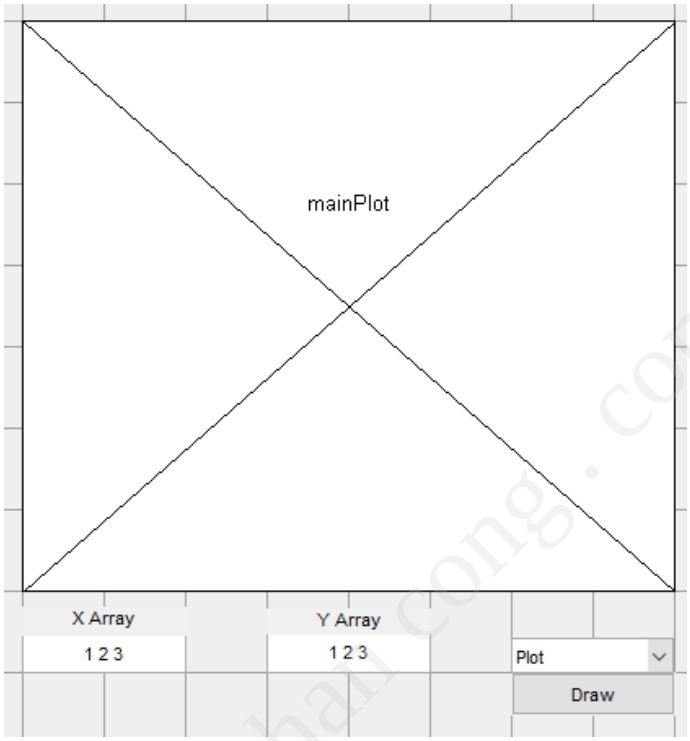
GUIDE là công cụ thường được dùng để thiết kế giao diện chương trình trong Matlab. So với các công cụ thiết kế giao diện của các ngôn ngữ khác như Java, C#, C++,... GUIDE tương đối đơn giản và khá giới hạn về chức năng. Thực chất, để có thể thiết kế các giao diện phức tạp trong GUIDE, ta thường phải sử dụng thêm các công cụ của Java.

Mỗi khi ta tạo một file giao diện mới từ GUIDE, Maltab sẽ tạo ra một file .fig (Hình 1) để lưu thông tin layout của các thành phần trong giao diện và một file .m để lưu code và cũng là logic thực thi của chương trình (Thuật toán 1 sẽ được lưu trong file .m).

Listing 1: Hàm Callback của push button **Draw**

```
% ---- Executes on button press in drawButton.
function OnDrawButtonCallback(hObject,eventdata,handles)
x = str2num(get(handles.XArrayEdit,'String'));
y = str2num(get(handles.YArrayEdit,'String'));
axes(handles.mainPlot);
selectedIndex = get(handles.plotStyle,'Value');
plotStyle = GetPlotStyle(selectedIndex);
switch plotStyle
  case 'plot'
    plot(x,y);
```

```
11    case 'stem'
12      stem(x,y);
13    case 'stairs'
14      stairs(x,y);
15  end
```



Hình 1: Giao diện đơn giản

Trong Matlab, khi người dùng thực hiện các thao tác như ấn nút hay nhập dữ liệu trên giao diện, các hàm được khai báo trong **Callback** ở Property Inspector sẽ được thực thi (Hình 2). Thuật toán 1 chính là hàm Callback khi người dùng nhấn nút **Draw** ở Hình 1. Hàm này sẽ lấy giá trị của mảng X và mảng Y, sau đó thực hiện vẽ biểu đồ như yêu cầu. Để đọc các giá trị của các phần tử trên giao diện trong code:

$$\text{var} = \text{get(handles.TagName,'ObjectValueName')}$$

Tương tự, để thay đổi giá trị từ code:

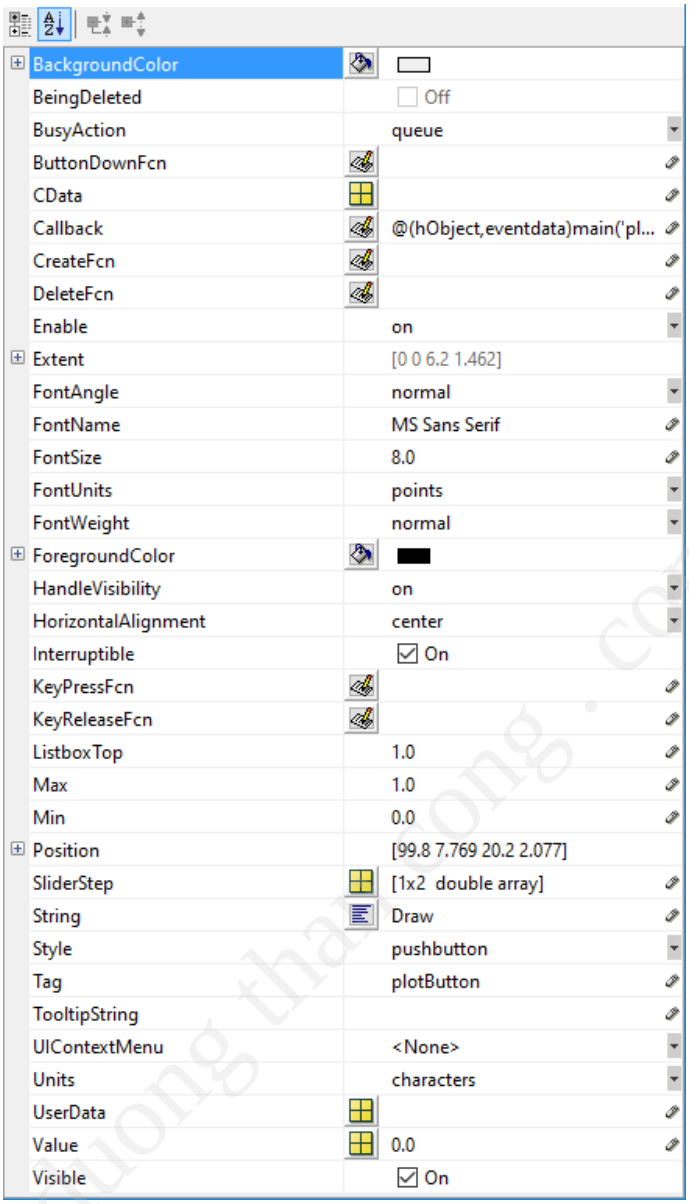
$$\text{set(handles.TagName,'ObjectValueName',newValue)}$$

2. Một số công cụ hỗ trợ trong GUIDE

a. Push Button

Push Button cho phép ta tạo các nút nhấn để người dùng điều khiển hoạt động của chương trình. Hình 2 trình bày các đặc tính của Push Button được Matlab hỗ trợ. Trong đó cần lưu ý:

- **BacgroundColor** quyết định màu nền của nút.
- **Callback** chỉ tới hàm được thực thi khi người dùng ấn nút.
- **FontName** và **FontSize** quyết định font được dùng để hiển thị tên của nút trên giao diện.
- **ForegroundColor** quyết định màu sắc của font.
- **Position** quyết định vị trí của nút trong giao diện.



Hình 2: Push Button property inspector.

- **String** quyết định tên của nút khi hiển thị trong giao diện.
- **Tag** quyết định tên của nút trong code (file .m).

b. Radio Button

Các đặc tính cũng tương tự như Push Button. Được dùng để tổ hợp các lựa chọn không được phép chọn cùng lúc. Trong đó **Value** cho biết một nút được chọn hay không. **Lưu ý**, ta phải đặt các Radio Button trong panel của **Button Group**.

c. Check Box

Tương tự như Radio Button nhưng thường được dùng khi người dùng được phép chọn một hoặc nhiều lựa chọn cùng lúc.

d. Edit Text

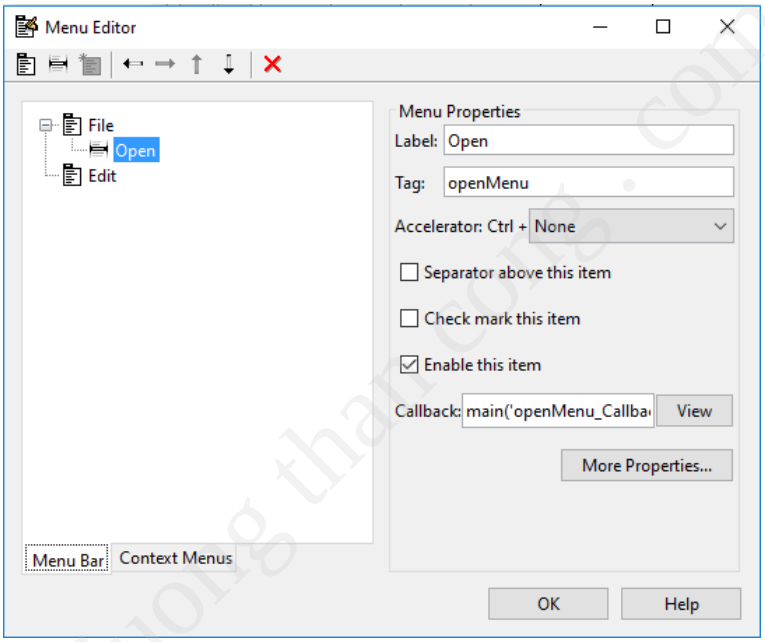
Cho phép người dùng nhập dữ liệu dạng string vào chương trình. Dữ liệu này được lưu trong biến **String** của Edit Text. Ta có thể đọc hay thay đổi giá trị người dùng nhập vào từ code như sau:

```
var = get(handles.editText1,'String')
set(handles.editText1,'String','some string')
```

e. Static Text
Được dùng để hiển thị read-only text trên giao diện. Ví dụ như **X Array, Y Array** ở Hình 1.

f. Pop-Up Menu
Ý nghĩa tương tự như Radio Button nhưng chiếm ít diện tích trên giao diện hơn. Các lựa chọn trong Pop-Up Menu được đánh số thứ tự từ 1 trở lên. **Value** sẽ lưu số thứ tự của lựa chọn đang được hiển thị trên giao diện. Ví dụ nếu giá trị **String** của Pop-up Menu lưu các giá trị sau: *Plot, Stem, Stairs*, thì khi người dùng chọn *Stem* trong Pop-up Menu, **Value** sẽ bằng 2.

g. Menu Editor
Matlab cho phép tạo các thành phần của menu bar (File, Edit, Help,...) và context menus (Copy, Paste,...) thông qua Menu Editor (Tools>Menu Editor) như ở Hình 3.



Hình 3: Menu Editor.

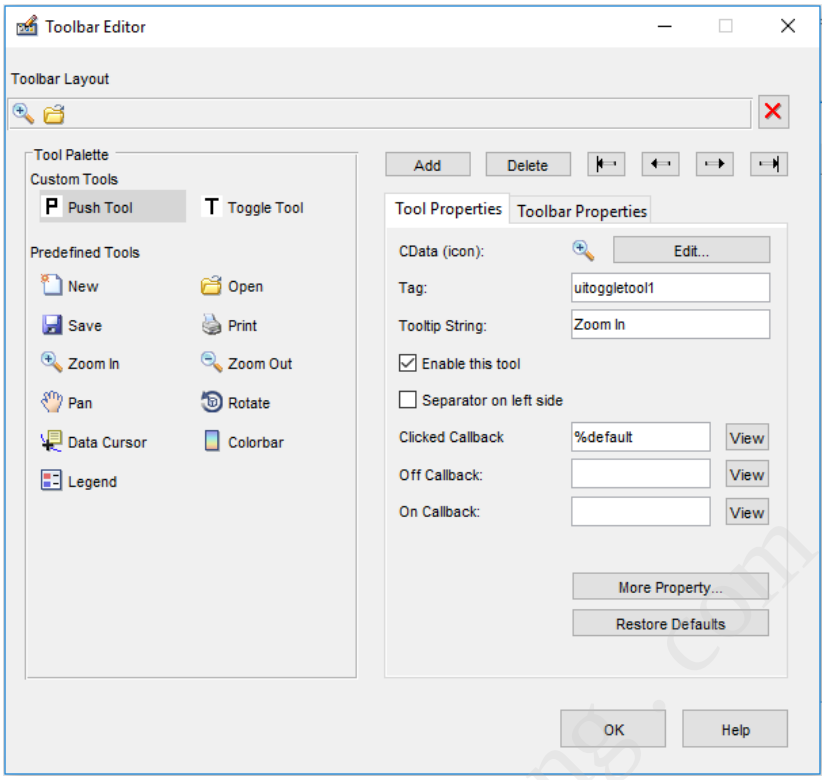
h. Toolbar Editor
Matlab cũng hỗ trợ tạo Toolbar từ các công cụ sẵn có (open file, print, save,...) trong Toolbar Editor (Tools>Toolbar Editor) như ở Hình 4.

i. Axes
Được dùng để hiển thị các biểu đồ được vẽ từ các hàm vẽ của Matlab như plot, stem, stairs,... Nếu chương trình có từ 2 đồ thị trở lên, ta có thể xác định đồ thị cần vẽ bằng lệnh:

```
axes(handles.AxesTagName)
```

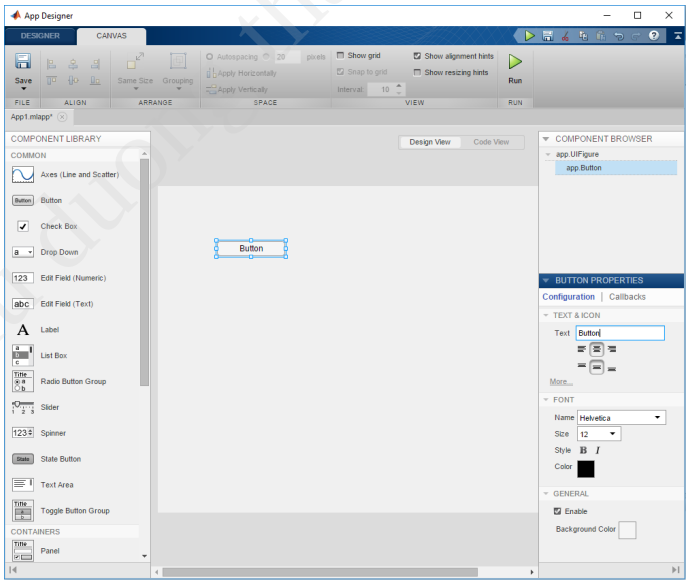
j. Panel
Được dùng để nhóm các phần tử phục vụ cho cùng một chức năng trong giao diện.

Trong các chương trình phức tạp, nếu ta hiển thị toàn bộ tất cả các thành phần (nút ấn, axes, dữ liệu...) cùng một lúc trên một cửa sổ sẽ khiến người dùng khó xác định được chức năng cụ thể của từng phần tử và logic của chương trình. Do đó ta có thể chia các phần tử này thành từng nhóm nhỏ ứng với các trạng thái khác nhau của chương trình và đưa vào các panel khác nhau. Sau đó ta có thể ẩn/hiện các panel tương ứng với quá trình thực thi của hệ thống nhằm đơn giản hóa giao diện.



Hình 4: Toolbar Editor.

C. App Designer



Hình 5: App Designer.

Trong các phiên bản mới, Matlab đã bắt đầu hỗ trợ một công cụ thiết kế giao diện mới tên là App Designer (Hình 5). Công cụ này tương đối thân thiện hơn với các lập trình viên đã quen thiết kế giao diện ở các ngôn ngữ khác. Tuy nhiên, do công cụ này được xây dựng dựa trên phương pháp thiết kế của ngôn ngữ lập trình hướng đối tượng, nên nó nằm ngoài nội dung môn học và sẽ không dược dùng trong bài thực hành.