

[資 料]

MATLAB Graphical User Interface 開発環境 GUIDE について

秋田県立大学 陳 国躍
東北大学情報部情報基盤課 共同利用支援係

MATLAB Graphical User Interface 開発環境 GUIDE は、グラフィカルユーザインターフェース (GUI) を作成するツールを提供する。これらのツールにより、MATLAB において対話的に作業することが可能であり、簡単に GUI を設計・作成できる。GUIDE ツールを使用すると次のことを行うことができる。

- GUI のレイアウト

GUIDE レイアウトエディタを使用すると、GUI コンポーネント (パネル、ボタン、テキストフィールド、スライダー、メニューなど) をクリック&ドラッグすることにより GUI をレイアウトエリアにレイアウトすることができる。

- GUI のプログラミング

GUIDE は、GUI の動作をコントロールする m-ファイルを自動的に作成する。m-ファイルは GUI を初期化し、また、GUI コンポーネントをクリックしたときに実行されるコマンドなど、すべての GUI コールバックに関する構造を含んでいる。ユーザーが最初に GUI を保存、または実行する場合、GUIDE は GUI を 2 つのファイルに保存される。

[1] 拡張子 .fig をもつ fig-ファイル、GUI レイアウトや、ポッシュボタン、メニュー、座標軸などの GUI コンポーネントの完全な描写を含んでいる。fig ファイルはバイナリファイルであり、GUIDE でレイアウトを変更する以外の方法では変更できない。

[2] 拡張子 .m をもつ m-ファイルは、コンポーネントに対するコールバックなど、GUI をコントロールするコードを含んでいる。一般に、GUI コンポーネントに対して記述するコールバックはこのファイルに追加する。

fig-ファイルと m ファイルは、同じ名前であればならない。これら 2 つのファイルは、通常同じフォルダーに存在し、GUI のレイアウトとプログラミングの作業に対応している。レイアウトエディタに GUI をレイアウトする場合、ユーザーの作業は fig-ファイル保存される。GUI をプログラミングする場合、作業は m-ファイルに保存される。

つぎの節では、GUIDE を用いた GUI アプリケーションの構築方法を説明し、GUI アプリケーションを作成する。Windows OS 上での日本語版 MATLAB が多いが、今回は、東北大学サイバーサイエンスセンターの英語版 MATLAB をベースにして説明する。

例 1 : 簡単な正弦(sin)関数を描く GUI の構築手順

- [1] Matlab を起動*する。
*起動方法についてはセンターWeb ページ
<http://www.ss.isc.tohoku.ac.jp/application/matlab.html> を参照
- [2] Command Window に `guide` と入力し Enter キーを押すと, 図 1.1 に示す「GUIDE Quick Start」というウィンドウが表示される。
- [3] 「Blank GUI (Default)」を選択し, 「OK」ボタンを押すことで図 1.2 のレイアウトエディタが表示される。
- [4] 正弦(sin)関数を表示するため, 図 1.3 のように axes 1 と Push Button を追加する。図 1.3 の左側にある Axes ボタンと Push Button をクリックし, マウスにより範囲を指定することで追加できる。
- [5] 名前を付けて保存する。ここでは SinWave.fig と名前をつけて保存すると, MATLAB エディター(Editor) に SinWave.m が表示される。
- [6] SinWave.m の
関数 `function pushbutton1_Callback(hObject, eventdata, handles)`
の下に下記のコードを追加する。

```
% -----
x = -6:0.1:8;
y = sin(x);
plot(x,y)
% x = -6 ~ 8 までのsin(x)関数を描画
% -----
```

- [7] SinWave.m のエディターの緑の▶実行ボタンをクリックすると, SinWave.m が自動的に保存され, SinWave の GUI が出力される。
- [8] GUI の画面上で, Push Button をクリックすると axes 上に sin(x)関数が表示される。

◎ 画面の編集をする場合は以下の方法がある。

1. 図 1.1 に示す GUI の初期ウィンドウに「既存の GUI を開く」から SinWave.fig を選ぶ
2. Command Window に `guide SinWave.fig` と入力
3. Command Window に `guide SinWave` と入力

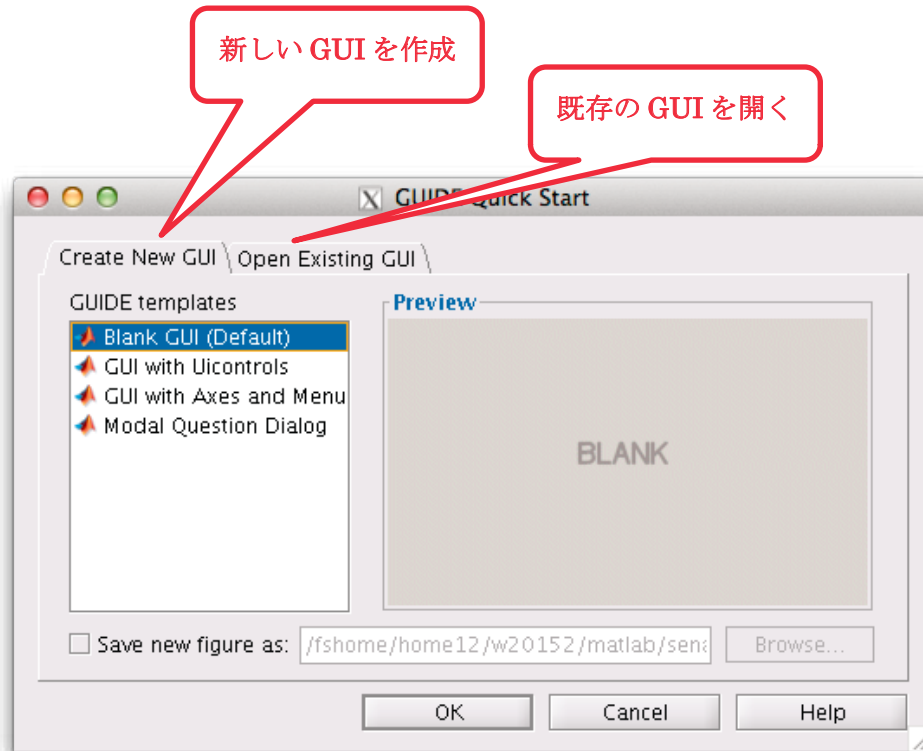


図 1.1. GUI の初期ウィンドウ (英語版)

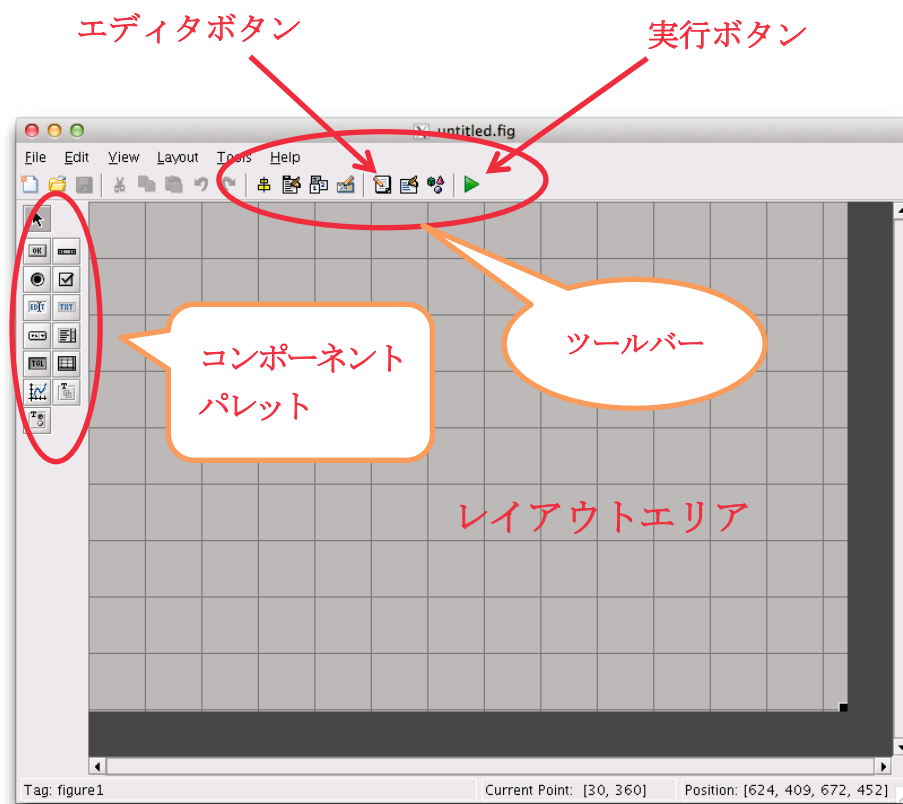


図 1.2. ブランク GUI テンプレートのレイアウトエディタ

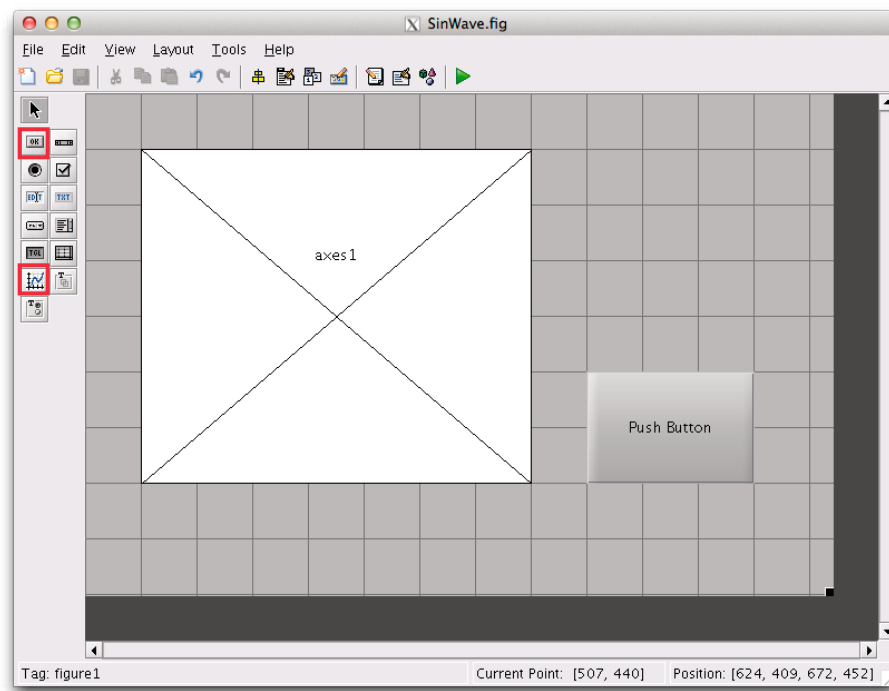


図 1.3. Push Button と axes 1 の追加

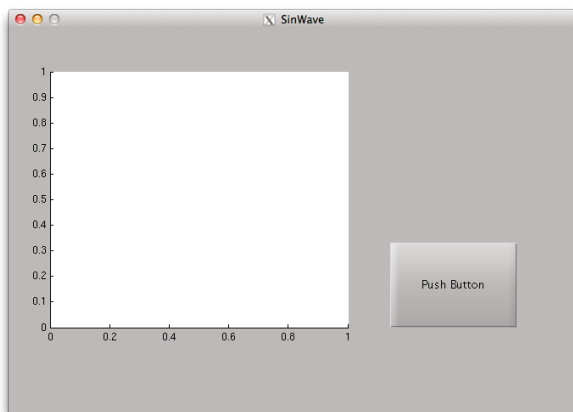


図 1.4. SinWave の初期状態

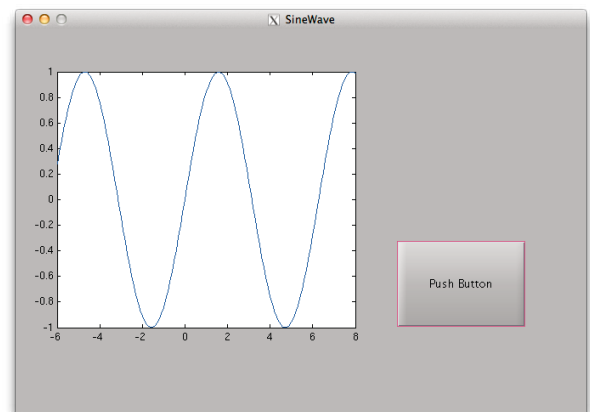


図 1.5. Push Button をクリックした後表示された結果

例 2 : GUI の応用: 正弦(sin)波線の太さの変更

例 1 で作成した GUI 上で, Slidar を用いて正弦 (sin) 波線の太さを変化させ, GUI のプログラムで関数間のデータを受け渡す方法を示す.

- [1] 例 1 で作成した SinWave.fig を編集するため Command Window に
`guide SinWave`
と入力し, GUI を編集する.
- [2] 例 1 で作成した GUI(SinWave.fig)に対し, Slidar を追加する.

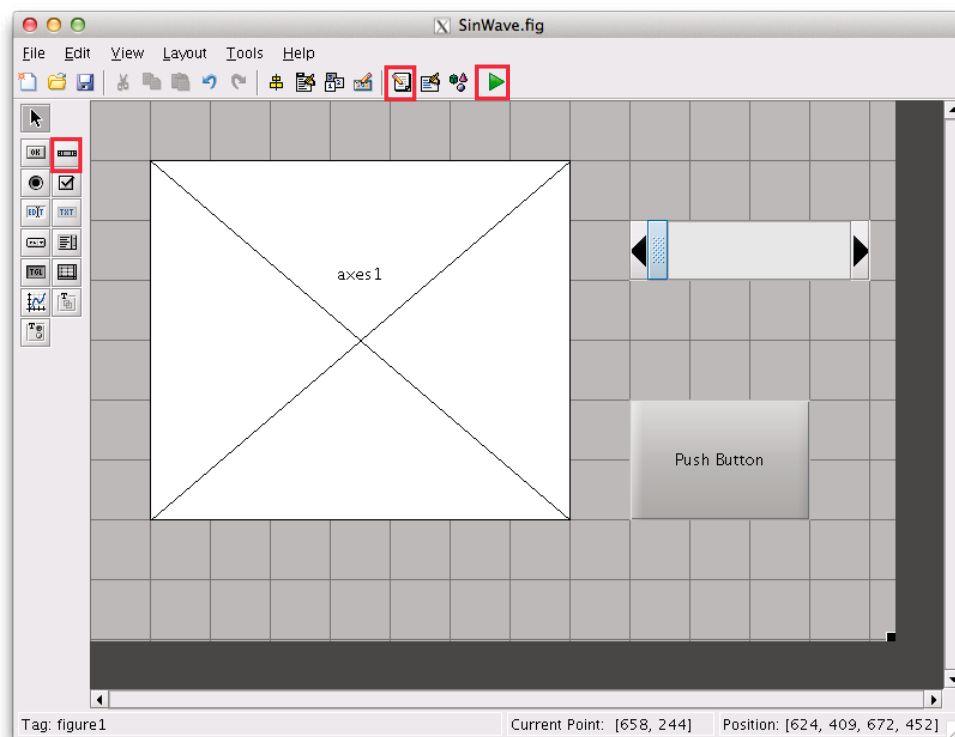



図 2.1. Slidar の追加

- [3] 図 2.1 のウィンドウ上でダブルクリックまたは, 右クリックし「Property Inspector」を選択すると, 図 2.2 に示すような「Inspector」ウィンドウが表示される. ここで, 「Tag」の値を “SinWave” と設定する.
- [4] 「Editor」 ボタン  をクリックすると, コードが MATLAB エディターに表示される.

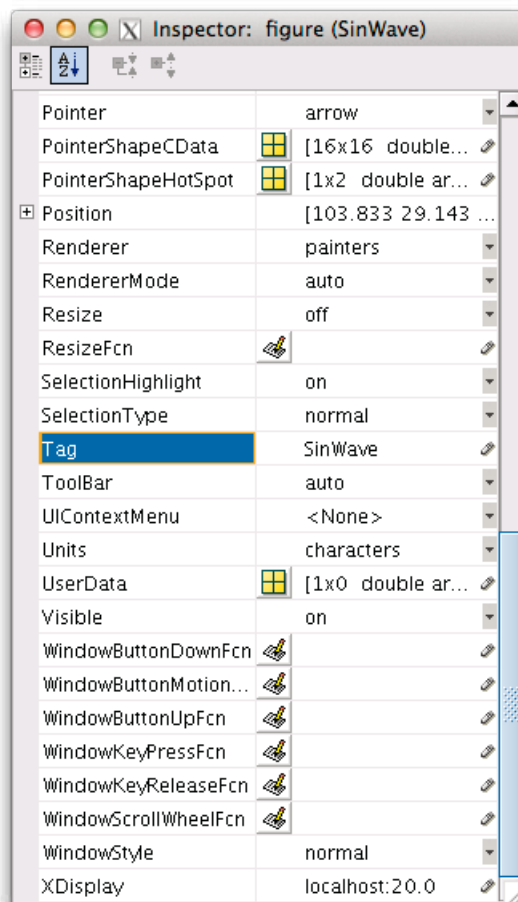


図 2.2. 「Inspector」の「Tag」に“SinWave”を設定

- [5] 関数 `function pushbutton1_Callback(hObject, eventdata, handles)` の下に下記のコードを追加する。

```


% -----
x = -6:0.1:8;
y = sin(x);
plot(x,y)

setappdata(handles.SinWave, 'x', x);
setappdata(handles.SinWave, 'y', y);
% データを格納
set(handles.slider1, 'max', 25, 'min', 1, 'value', 1);
% Sliderの最大値を25, 最小値を1, 初期値を1に設定
% -----

```

[6] 関数 `function slider1_Callback(hObject, eventdata, handles)` の下に下記のコードを追加する。

```
% -----  
x=getappdata(handles.SinWave, 'x');  
y=getappdata(handles.SinWave, 'y');  
% データを取得  
val=get(hObject, 'value');  
% Sliderの現在位置の値を取得  
plot(x,y, 'LineWidth', val);  
% Plotプロパティ(線幅LineWidth)の値を, Slidarの現在位置の値に設定  
% -----
```

[7] 実行ボタン  をクリックすると `SinWave.m` が自動的に保存される。同時に GUI が実行され、図 2.3 の初期状態が表示される。

[8] Push Button をクリックすると、図 2.4 の画面が表示される。更に、Slider を移動させると、図 2.5 のように正弦波線の太さを変更できる。

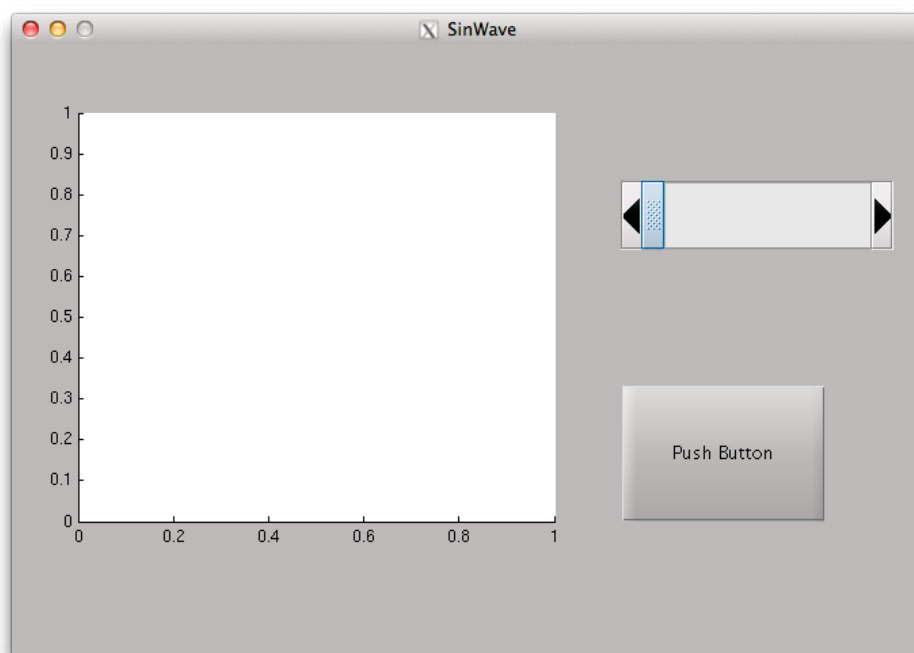


図 2.3. Slider を追加した後の SinWave の初期状態

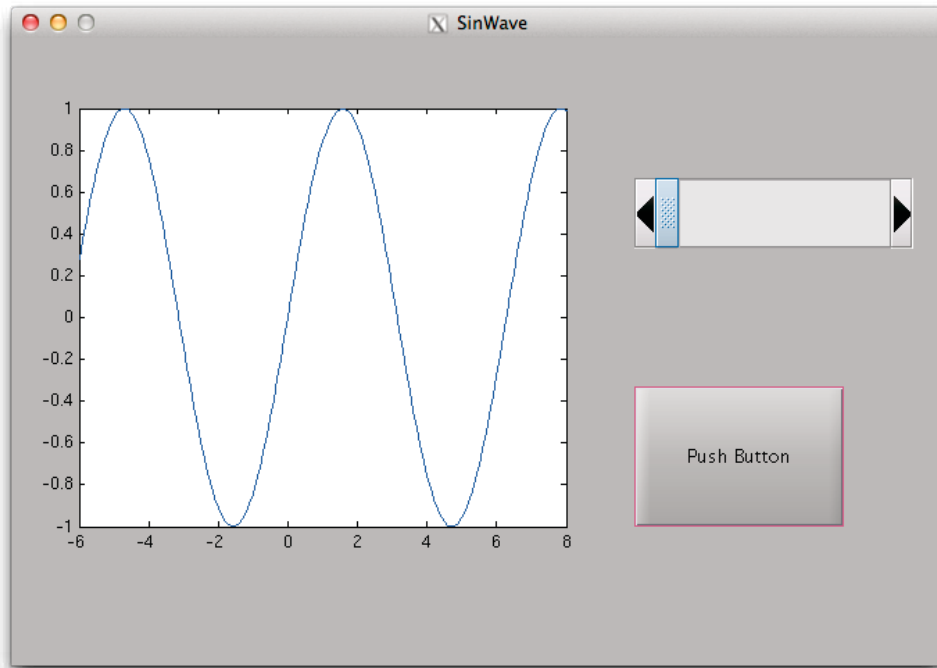


図 2.4. Push Button を押した後表示された結果

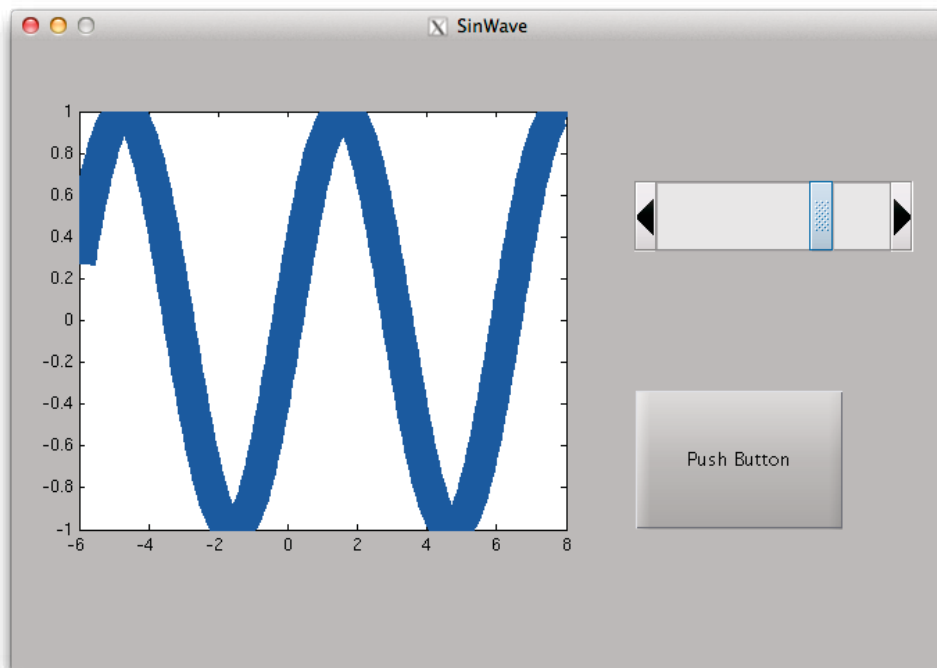


図 2.5. Slider を移動させ、正弦(sin)波線の太さを変化させた結果

例 3 : 画像処理 GUI の作成方法

図 3.1 に示すように、画像を読み込んでグレー化したものと、Slidar を用いて閾値の値を変化させ、その閾値で二値化した画像を表示する GUI の例を示す。

1. カラー画像の場合はグレー化する。グレー画像の場合はそのまま。
2. グレー化した画像からヒストグラムを表示させる。
3. Slidar を用いて閾値の値を変化させ、その閾値で二値化した画像を表示する。
4. 各種操作についてはメニューバーのメニュー項目から選択すると実行できる。ショートカット「Ctrl + ?」も実行できる。
5. キャプション等のフォント、サイズなどは簡単に変更できる。これらについての詳細な説明は省略する。

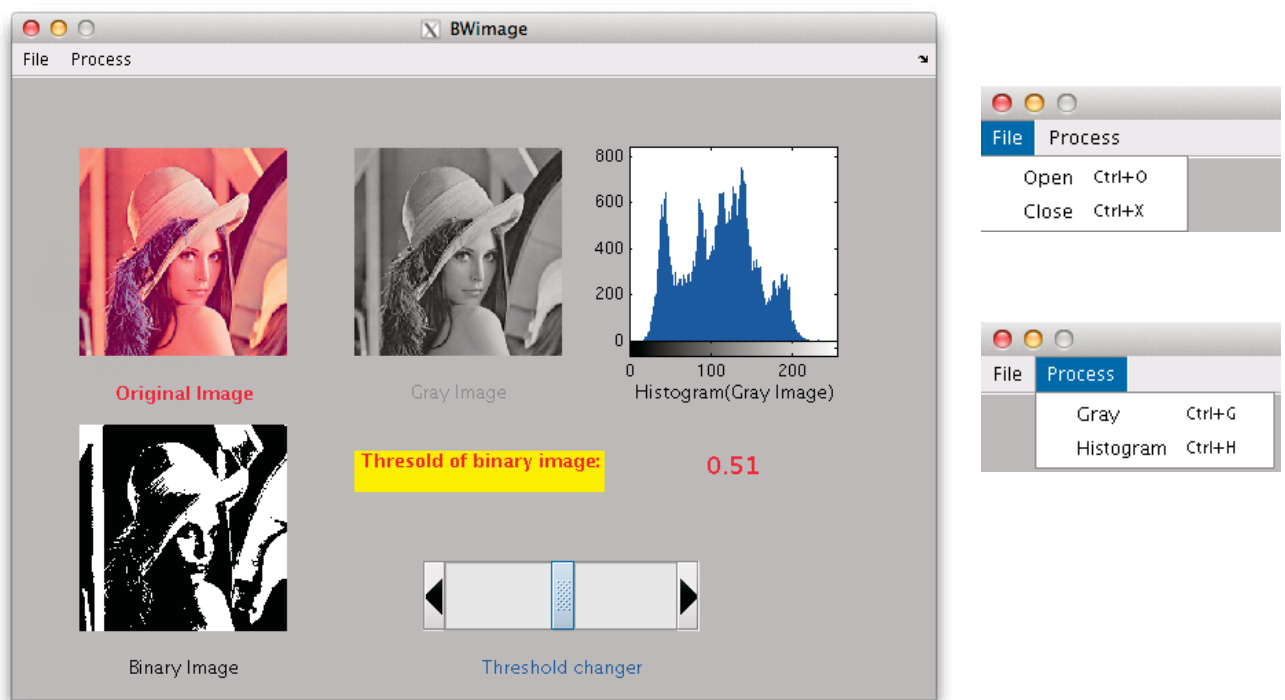


図 3.1. GUI を用いて画像処理した例（左側）、設定したメニュー項目（右側）

- [1] Matlab を立ち上げる。
- [2] Command Window に `guide` と入力し Enter キーを押すことで「Blank GUI (Default)」というウィンドウ（図 1.1）が表示される。
- [3] 「Blank GUI (Default)」を選択し、「OK」ボタンを押すことで図 3.2 の画面が表示される。
- [4] 図 3.2 のウィンドウ上でダブルクリックまたは右クリックし「Property Inspector」をクリックすると、図 3.3 に示すような「Inspector」ウィンドウが表示される。ここで、「Tag」の値を “BWImage” とする。

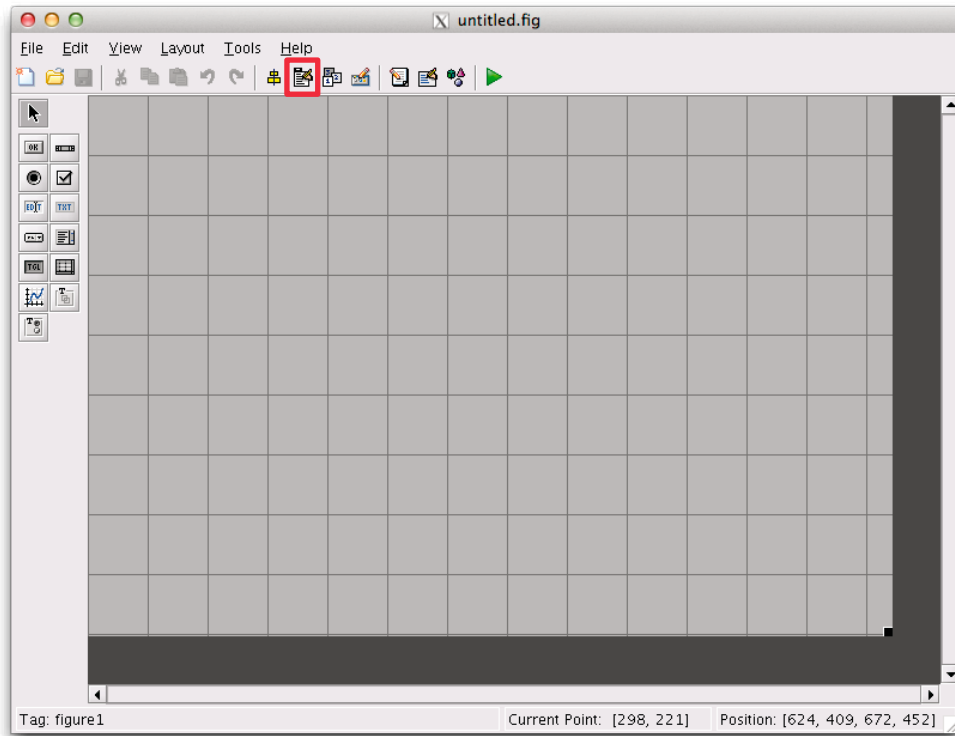


図 3.2. ブランク GUI テンプレートのレイアウトエディタ

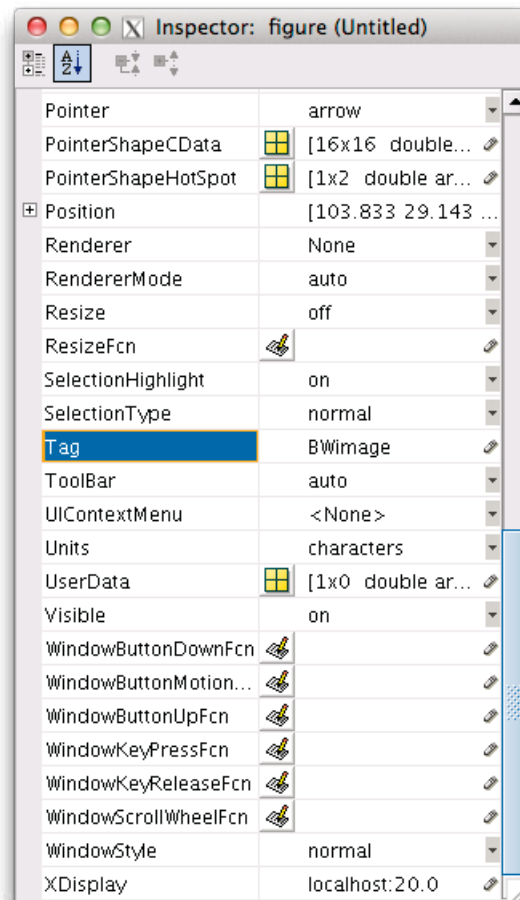


図 3.3. 「Inspector」の「Tag」に“BWimage”を設定

- [5] 図 3.3 のウィンドウを閉じ、図 3.2 の「Menu Editor」ボタンをクリックし、図 3.4 に示す「Menu Editor」ウィンドウを表示させる。

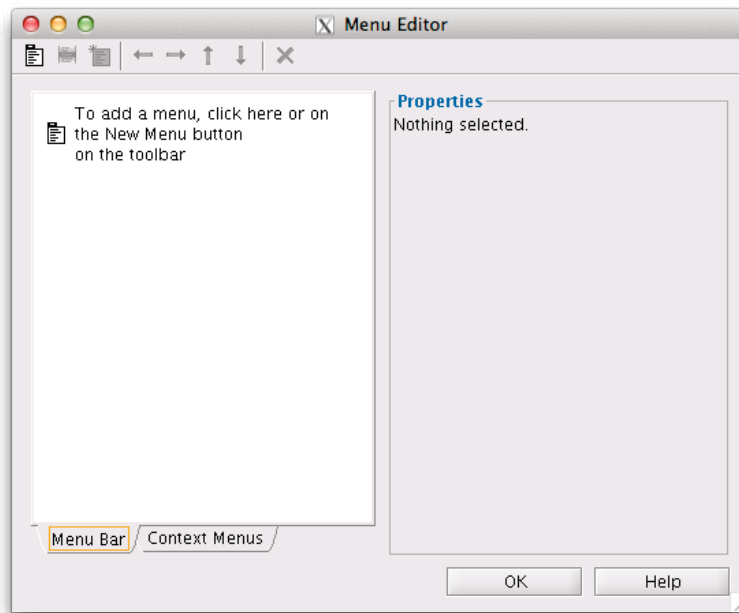


図 3.4. メニューエディタ

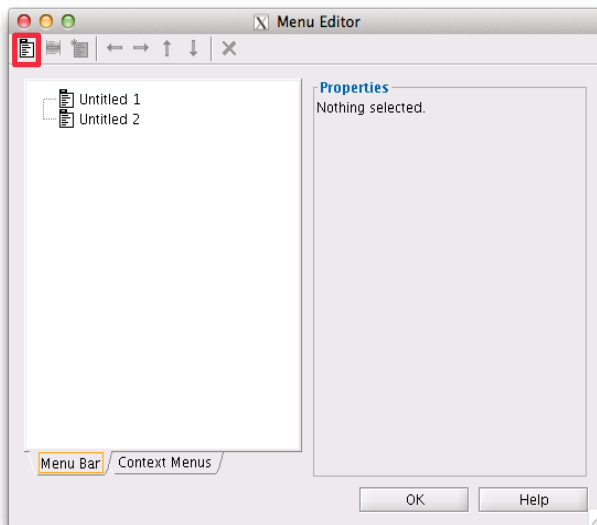


図 3.5. 新規メニューを追加

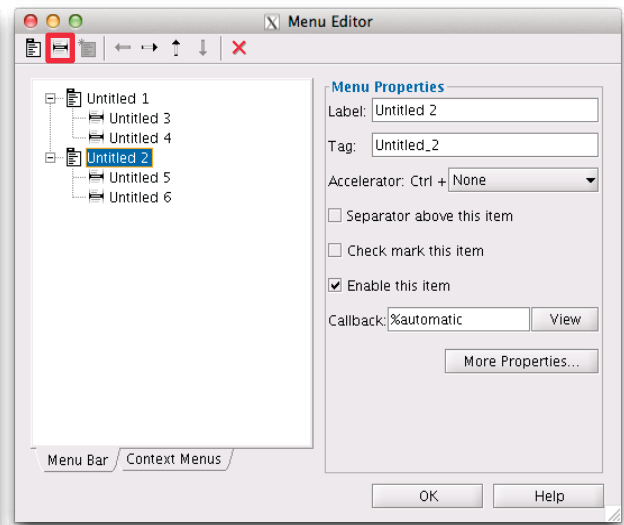


図 3.6. 新規メニュー項目を追加

- [6] 図 3.5 に示す「New Menu」ボタンをクリックすると「Untitled 1」というメニュータイトルが表示される。同様に「Untitled 2」というメニュータイトルが作成される。
- [7] 図 3.5 に示すように、「Untitled 1」をクリックすると「New Menu」ボタンの右側にある「New Menu Item」ボタンが使用可能となり、それをクリックすると図 3.6 のように「Untitled 3」というメニュー項目タイトルが表示される。同様に「Untitled 4～6」というメニュー項目タイトルが作成される。

- [8] 「Untitled 1」というメニューをクリックすると、右側に「Menu Properties」が表示される。そして図 3.7 のように「Label:」に“File”，「Tag:」に“m_file”を設定する。
- [9] 「Untitled 6」というメニューをクリックすると、右側に「Menu Properties」が表示される。次に図 3.8 のように「Label:」に“Histogram”，「Tag:」に“m_process_hist”，「Accelerator: Ctrl +」に“H”を設定する。

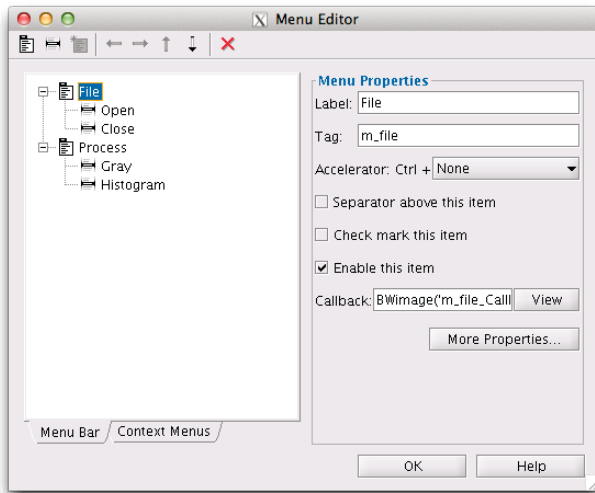


図 3.7. File メニューの設定

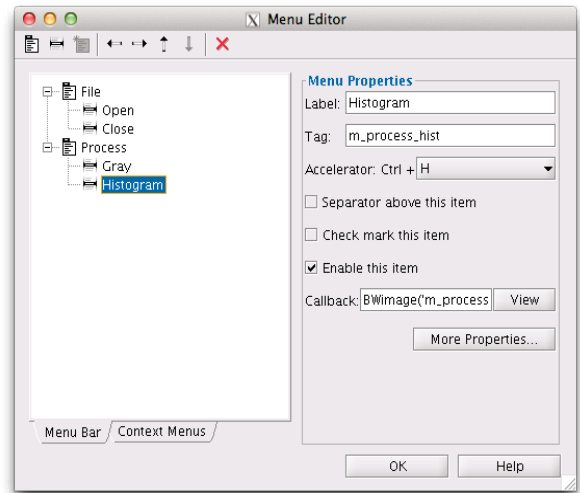


図 3.8. Histogram メニュー項目の設定

- [10] 同様に、各メニューとメニュー項目の「Menu Properties」を表 1 に示す。ただし、これらの「Menu Properties」の値は任意であり、変更しても構わない。「Tag:」の値がプログラムの関数名に自動的に設定されるため注意する必要がある。最後に OK ボタンをクリックして終了する。

表 1. 各メニューとメニュー項目のメニュープロパティ

	Label ラベル	Tag タグ	Accelerator アクセラレータ
ファイル	File	m_file	
開く	Open	m_file_open	O
終了	Close	m_file_close	X
処 理	Process	m_process	
グレー化	Gray	m_process_gray	G
ヒストグラム	Histogram	m_process_hist	H

- [11] 次に処理した画像を表示するための「axes」、そのキャプションを表示する「Static Text」、
「Slider」値を表示する「Static Text」を配置する. 図 3.9 の左側にあるコンポーネント
ボタンをクリックすると、マウスにより範囲を指定することができる. すなわち、クリッ
ク&ドラッグすることで、図 3.9 のように GUI をレイアウトエリアにレイアウトするこ
とができる.

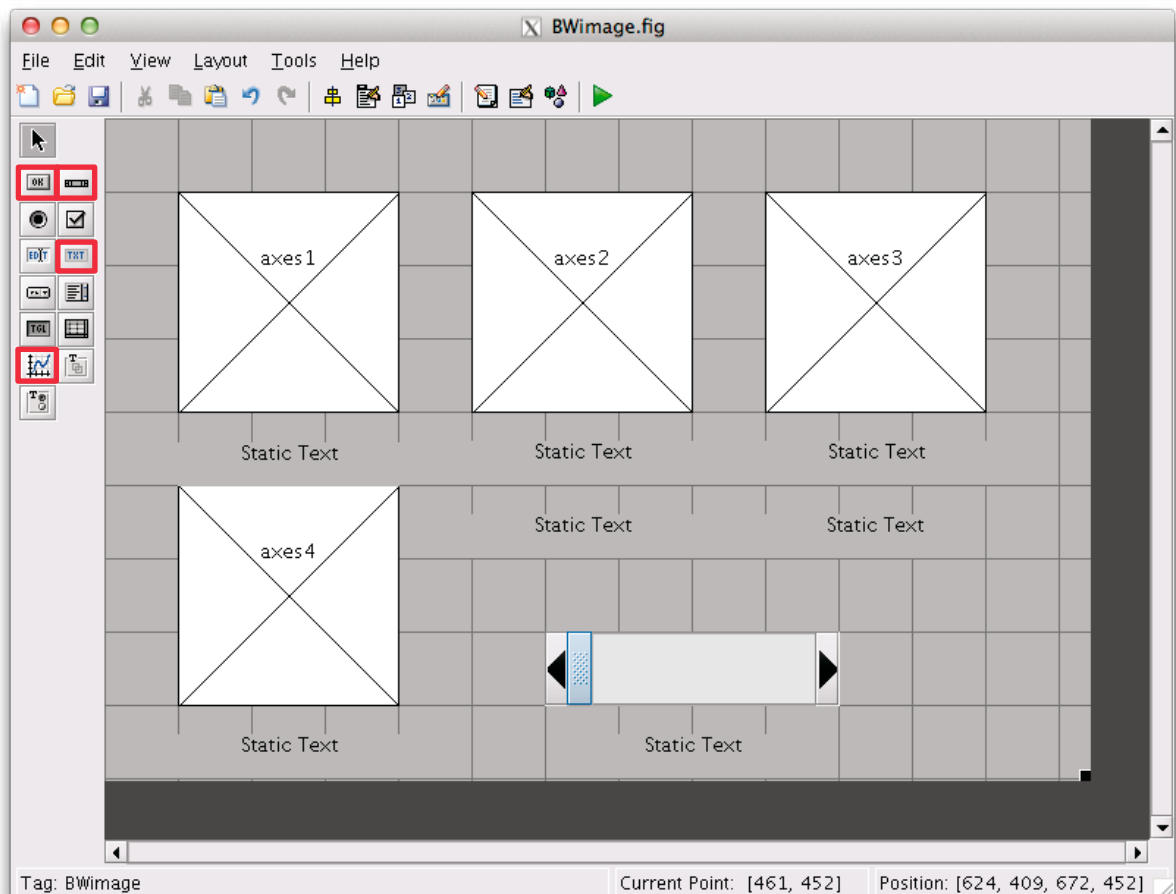


図 3.9. GUI をレイアウトエリアにレイアウトする

- [12] 図 3.9 における各コンポーネント上でダブルクリックまたは右クリックすることで各々の
「Inspector」を開くことができ、状況に応じて「Inspector」の内容を設定できる. いく
つかの例を図 3.10–3.12 に示す.
- [13] 図 3.10 に示すように、スタティックテキスト(Static Text)の「Inspector」の内容が表示
され、「Position」など多くの項目を修正、設定できる. 他のコンポーネントの「Inspector」
の項目はより多い場合もある.
- [14] 図 3.10 に示す通り、「BackgroundColor」に“黄色”，「FontWeight」に“bold”，
「ForegroundColor」に“赤色”，「String」に“Threshold of binary image”を設定する.

[15] 図 3.11 のように，axes 1 のキャプションとした Static Text の設定は，「String」に “Original Image”，その装飾として，「ForegroundColor」に “赤色”，「FontWeight」に “bold” を設定する。

[16] Slider の値を表示するため，その「Inspector」の初期値として「String」に “0”，「Tag」に “valcnt”，「ForegroundColor」に “赤色” を設定する。

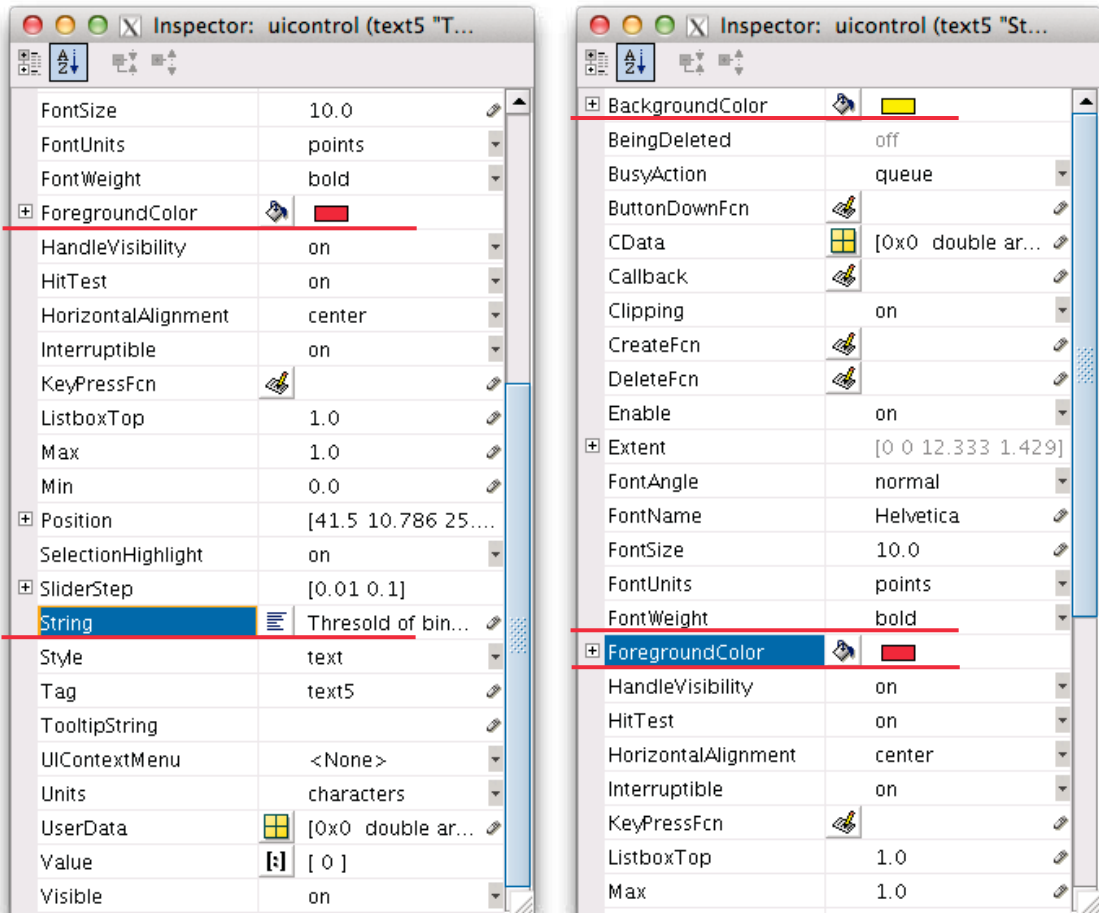


図 3.10. Static Text の「Inspector」の内容

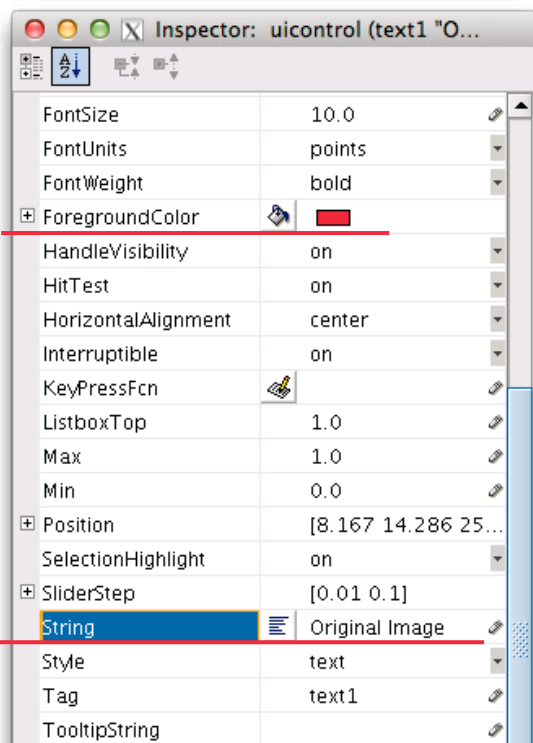


図 3.11. 「String」に“Original Image”，「ForegroundColor」に“赤色”，「FontWeight」に“bold”を設定

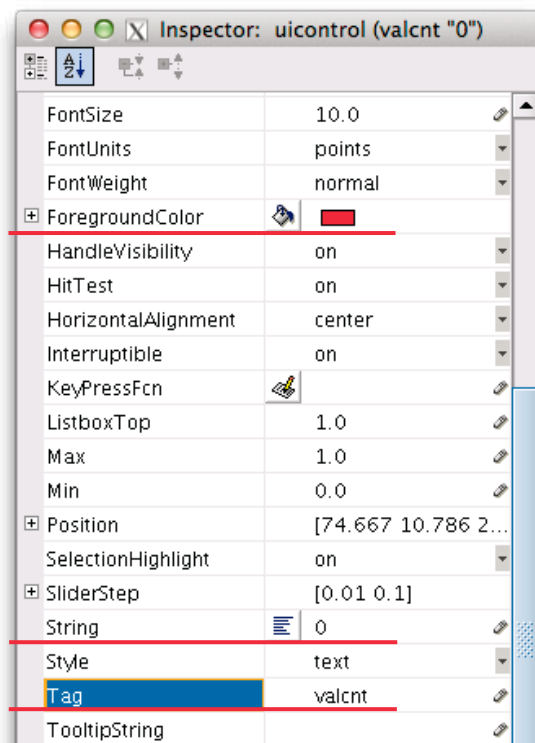


図 3.12. 「Slider」の値を表示するため、初期値「String」に“0”，「Tag」に“valcnt”，「ForegroundColor」に“赤色”を設定

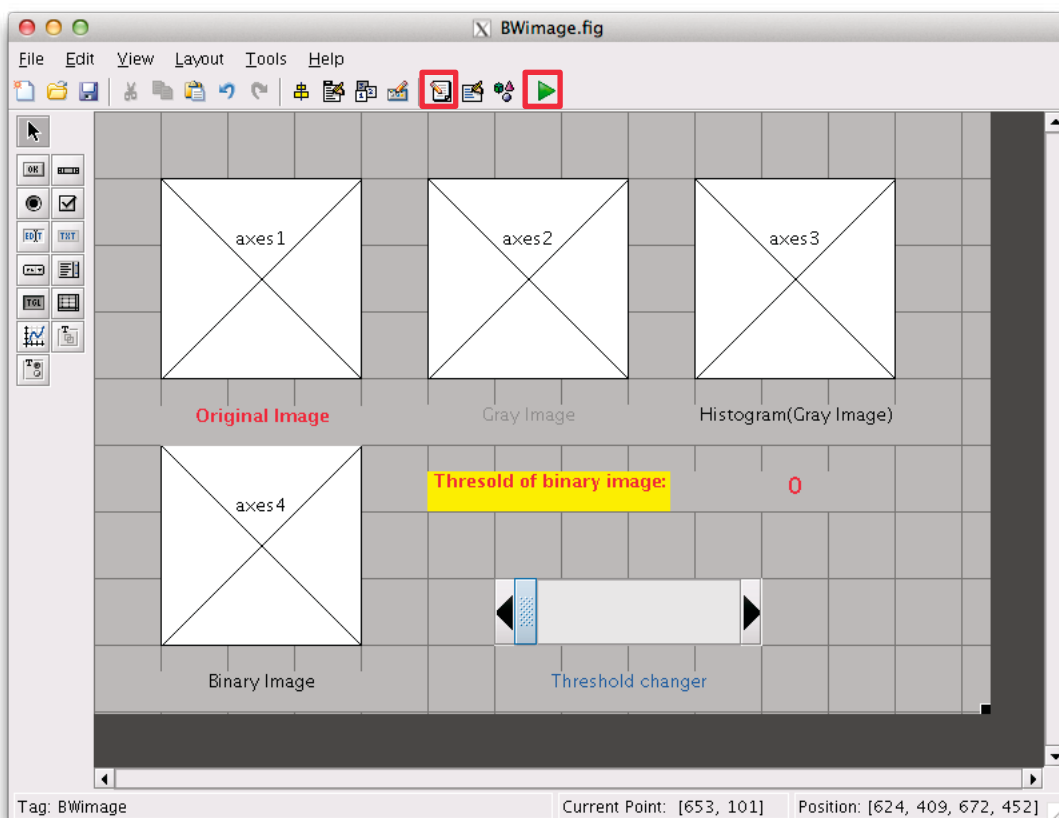



図 3.13. 各コンポーネントの「Inspector」を設定後の BWImage.fig 画面

- [17] 各「Inspector」を設定後の画面を図 3.13 に示す。設定したコンポーネント上で、ダブルクリックまたは右クリックし、その「Inspector」を開くことで、何度でも編集できる。
- [18] 「BWImage.fig」という名前で保存すると、図 3.14 に示すように「Editor」が起動する。
- [19] 図 3.9 の「Editor」  ボタンをクリックし、図 3.14 に示すように「Editor」を起動させることもできる。

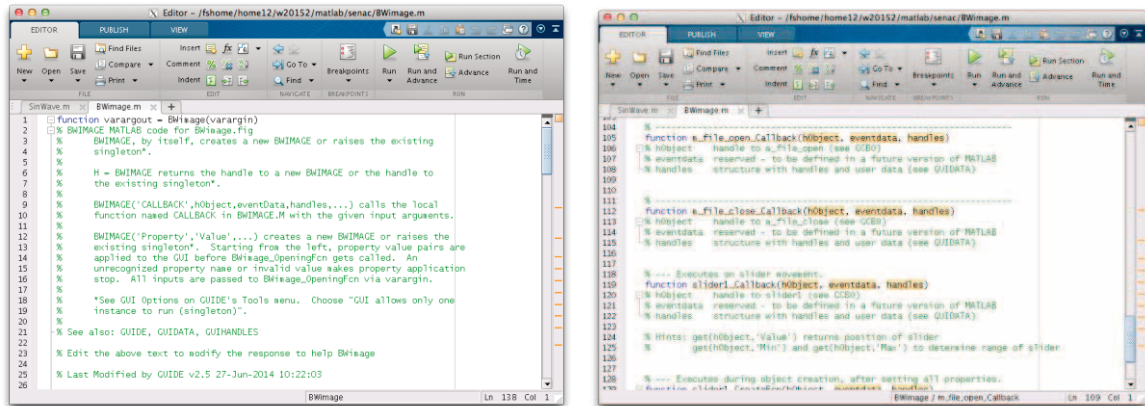


図 3.14. M-ファイルの Editor 画面

- [20] GUI をプログラミングするには、図 3.14 に示すような「Editor」の中に、[21]–[25]に示すようにコードを追加する。
- [21] 関数 `function m_file_open_Callback(hObject, eventdata, handles)` の下に以下のコードを追加する。

```

% -----
[filename,pathname]=...
    uigetfile({'*.bmp;*.jpg;*.png;*.jpeg;'.
    'Image Files (*.bmp,*.jpg,*.png,*.jpeg)';...
    '*.*', 'All Files (*.*)'}, 'Pick an image');
% オリジナル画像を読み込む
if isequal (filename,0) || isequal(pathname,0)
    return;
end
axes(handles.axes1);
fpath = [pathname filename];
[img_src,map] = imread(fpath);
imshow(img_src);
% オリジナル画像をaxes1に表示
setappdata(handles.BWImage, 'img_data', img_src);
% 画像データを格納
% -----

```


[22] 関数 `function m_file_close_Callback(hObject, eventdata, handles)` の下に以下のコードを追加する.

```
% -----
close all
% BWimageウィンドウを閉じ, 終了
% -----
```

[23] 関数 `function m_process_gray_Callback(hObject, eventdata, handles)` の下に以下のコードを追加する.

```
% -----
axes(handles.axes2);
img_src = getappdata(handles.BWimage, 'img_data');
% 画像データを取得
if (size(img_src,3)==3)
    img_gray=img_src(:,:,1) * 0.229 + img_src(:,:,2) * 0.587 ...
        + img_src(:,:,3) * 0.114 ;
else
    img_gray = img_src;
end
% 加重平均法によるカラー画像のグレイ化し, グレイ画像の場合はそのまま
imshow(img_gray);
% グレイ化した画像をaxes2に表示
setappdata(handles.BWimage, 'img_data', img_gray);
% グレイ化した画像を格納
% -----
```

[24] 関数 `function m_process_hist_Callback(hObject, eventdata, handles)` の下に以下のコードを追加する.

```
% -----
axes(handles.axes3);
img_gray = getappdata(handles.BWimage, 'img_data');
% グレイ化した画像を取得
imhist(img_gray);
% 画像のヒストグラムをaxes3に表示
% -----
```


[25] 関数 `function slider1_Callback(hObject, eventdata, handles)` の下に以下のコードを追加する.

```
% -----
val = get(hObject, 'Value');
% スライダー値を取得
set(handles.valcnt, 'String', num2str(val));
% スライダー値を表示
axes(handles.axes4);
```

```

img_gray = getappdata(handles.BWimage, 'img_data');
% グレー化した画像を取得
BW = im2bw(img_gray, val);
% 二値化画像を作成
imshow(BW);
% 二値化画像をaxes4に表示
% -----

```

- [26] 実行ボタン  をクリックすると `BWimage.m` が自動的に保存され、GUI が実行される。
 図 3.15 に示す `BWimage` の初期画面が表示される。
- [27] 図 3.16 に示すように、図の右側のメニュー、またはショートカット「`Ctrl + ?`」を実行すると左側の実行結果が表示される。
- [28] `Slider` を左右に移動すると、図 3.17 に示すように、二値化画像が表示され、その閾値も表示される。
- [29] 図 3.18 に示すように、グレー画像でも同様な結果が得られる。
- [30] 図 3.15–3.18 のメニューバーの「`File`」–「`Close`」、またはショートカット「`Ctrl + X`」を実行すると、`BWimage` ウィンドウを閉じ、終了することができる。

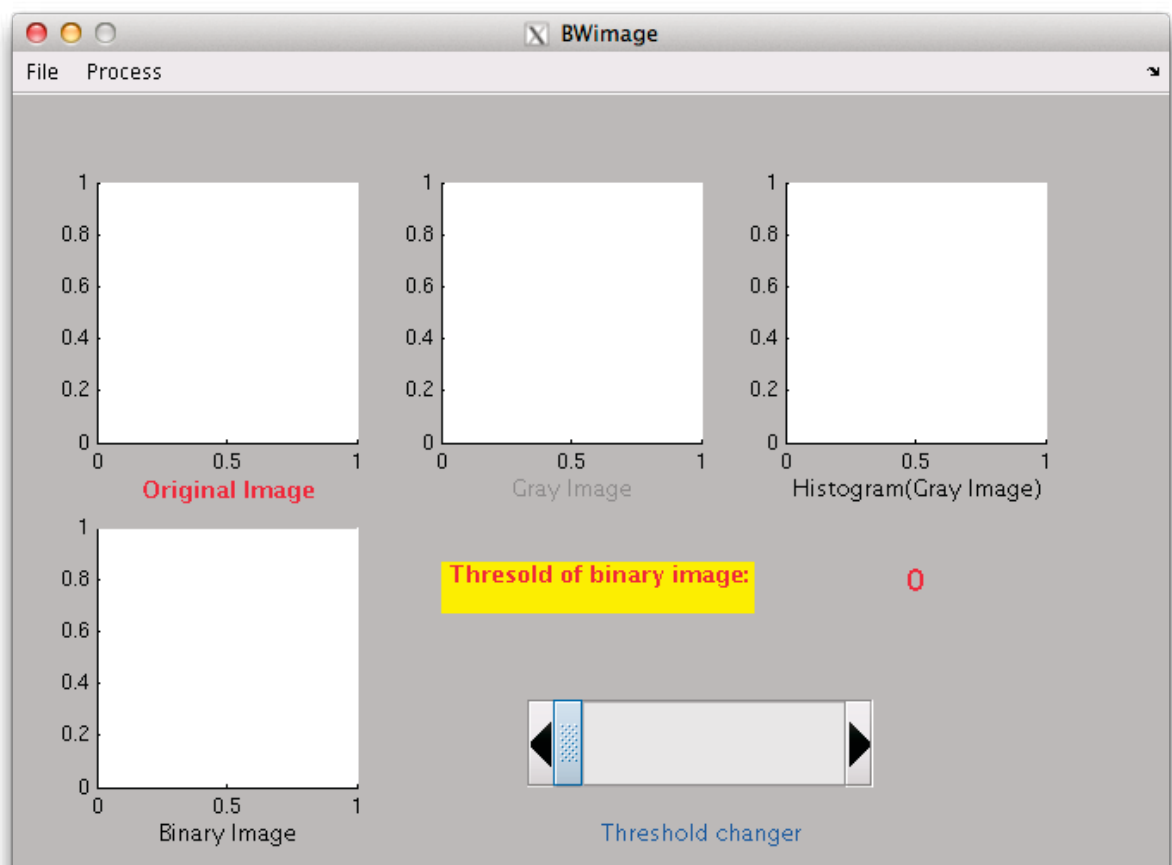


図 3.15. `BWimage` を起動した初期画面

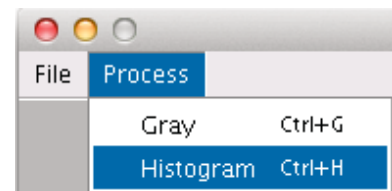
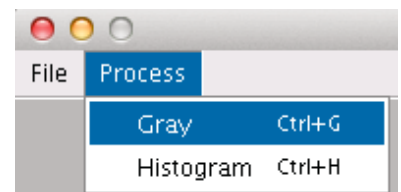
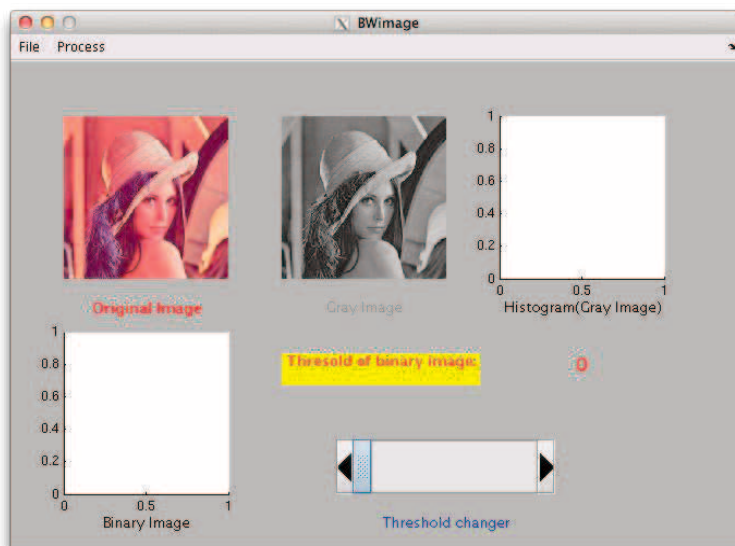
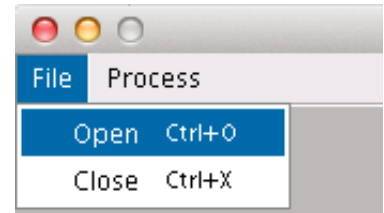


図 3.16. 右側のメニュー，或いはショートカット「Ctrl+?」を実行すると左側の実行結果が表示される

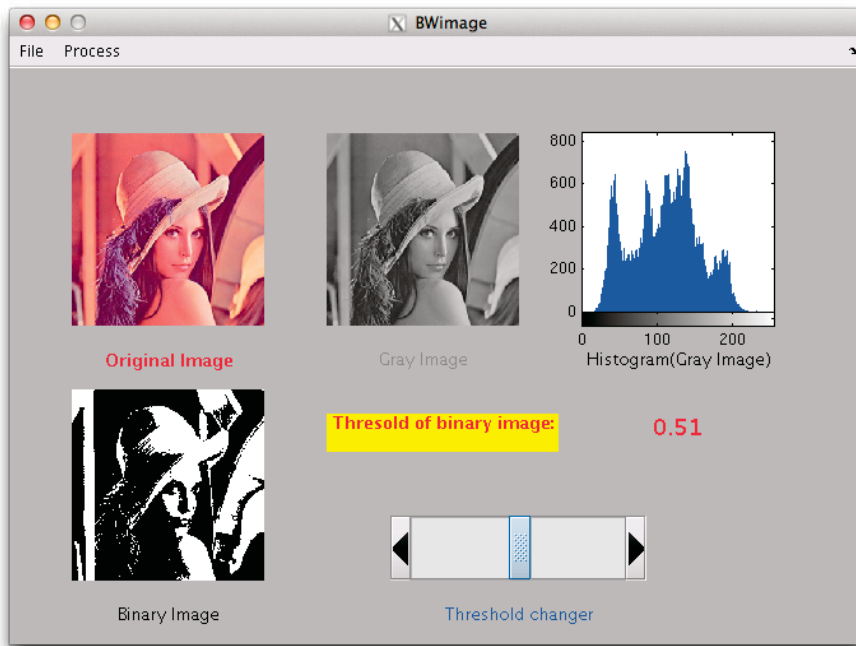


図 3.17. カラー画像を用いて処理した結果

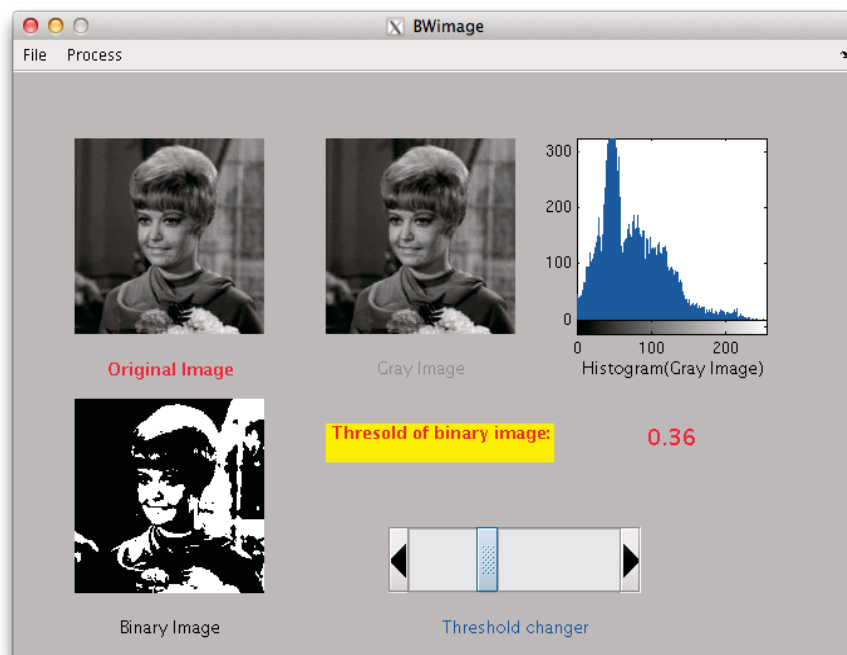


図 3.18. グレー画像を用いて処理した結果