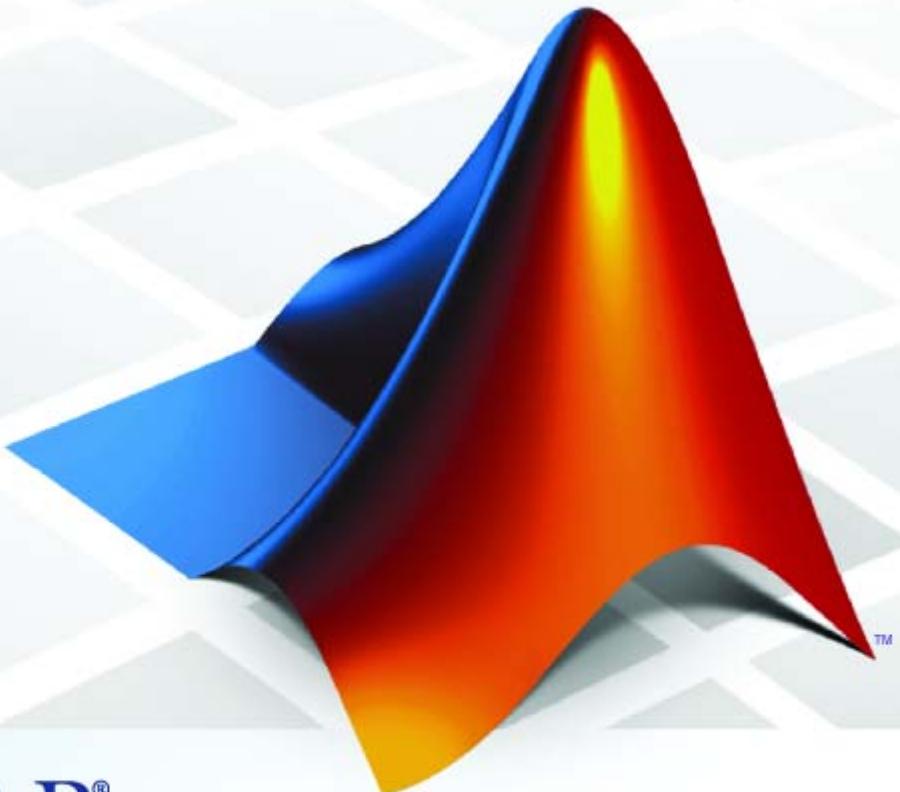


MATLAB® 7

グラフィカル ユーザー インターフェイスの 作成



MATLAB®

How to Contact The MathWorks



www.mathworks.com
comp.soft-sys.matlab
www.mathworks.com/contact_TS.html

Web
Newsgroup
Technical Support
Product enhancement suggestions
Bug reports
Documentation error reports
Order status, license renewals, passcodes
Sales, pricing, and general information



suggest@mathworks.com
bugs@mathworks.com
doc@mathworks.com
service@mathworks.com
info@mathworks.com



508-647-7000 (Phone)



508-647-7001 (Fax)



The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

MATLAB® グラフィカル ユーザー インターフェイスの作成

© COPYRIGHT 2000–2009 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

November 2000	Online Only	New for MATLAB 6.0 (Release 12)
June 2001	Online Only	Revised for MATLAB 6.1 (Release 12.1)
July 2002	Online Only	Revised for MATLAB 6.6 (Release 13)
June 2004	Online Only	Revised for MATLAB 7.0 (Release 14)
October 2004	Online Only	Revised for MATLAB 7.0.1 (Release 14SP1)
2005 年 3 月	Online Only	Revised for MATLAB 7.0.4 (Release 14SP2)
2005 年 9 月	Online Only	Revised for MATLAB 7.1 (Release 14SP3)
2006 年 3 月	Online Only	Revised for MATLAB 7.2 (Release 2006a)
2006 年 6 月	Online Only	Revised for MATLAB 7.2
2006 年 9 月	Online Only	Revised for MATLAB 7.3 (Release 2006b)
2007 年 3 月	Online Only	Revised for MATLAB 7.4 (Release 2007a)
2007 年 9 月	Online Only	Revised for MATLAB 7.5 (Release 2007b)
2008 年 3 月	Online Only	Revised for MATLAB 7.6 (Release 2008a)
2008 年 10 月	Online Only	Revised for MATLAB 7.7 (Release 2008b)
2009 年 9 月	Online Only	Revised for MATLAB 7.8 (Release 2009a)
2009 年 9 月	Online Only	Revised for MATLAB 7.9 (Release 2009b)

GUI 作成の紹介

MATLAB の GUI について

1

GUI とは	1-2
GUI の動作	1-4
はじめに	1-6
MATLAB GUI を作成する方法	1-8

GUIDE を使用して簡単な GUI を作成する

2

GUIDE:簡単な紹介	2-2
GUIDE とは	2-2
GUIDE を開く	2-2
GUI のレイアウト	2-7
GUI のプログラミング	2-8
例:簡単な GUI	2-9
簡単な GUI の概要	2-9
完成したレイアウトとその GUI M ファイルの表示	2-10
簡単な GUI のレイアウト	2-11
レイアウトエディター内に新しい GUI を開く	2-11
GUI の Figure サイズの設定	2-14
コンポーネントの追加	2-15
コンポーネントの整列	2-16
コンポーネントへのテキストの追加	2-18

完成したレイアウト	2-23
GUI レイアウトを保存	2-25
簡単な GUI のプログラミング	2-28
M ファイルへのコードの追加	2-28
プロットするデータの生成	2-28
ポップアップ メニューのプログラミング	2-31
プッシュ ボタンのプログラミング	2-33
GUI の起動	2-36

簡単な GUI をプログラミングで作成する

3

例:簡単な GUI	3-2
簡単な GUI の概要	3-2
完成した例の表示	3-3
関数のまとめ	3-4
GUI M ファイルの作成	3-6
簡単な GUI のレイアウト	3-8
Figure の作成	3-8
コンポーネントの追加	3-8
GUI の初期化	3-12
GUI のプログラミング	3-15
ポップアップ メニューのプログラミング	3-15
プッシュ ボタンのプログラミング	3-16
コールバックをコンポーネントと関連させる	3-16
GUI の起動	3-18
最終の M ファイル	3-18

GUIDE を使用して GUI を作成する

GUIDE とは

4

GUIDE:ご利用の前に	4-2
GUI のレイアウト	4-2
GUI のプログラミング	4-2
GUIDE ツールのまとめ	4-3

GUIDE の設定とオプション

5

GUIDE の設定	5-2
プリファレンス設定	5-2
設定の確認	5-2
下位互換性の設定	5-4
その他の設定	5-6
GUI オプション	5-9
[GUI オプション] ダイアログ ボックス	5-9
サイズ変更動作	5-10
コマンド ラインからのアクセス	5-10
FIG-ファイルと M ファイルの作成	5-11
FIG-ファイルのみ作成	5-14

GUI の設計	6-2
GUIDE の起動	6-4
GUI テンプレートの選択	6-5
テンプレートへのアクセス	6-5
テンプレートの説明	6-6
GUI サイズの設定	6-15
レイアウトエリアを最大化する	6-18
GUI にコンポーネントを追加	6-19
利用可能なコンポーネント	6-20
多くのコンポーネントをもつ GUI の取り扱い	6-24
GUIDE レイアウトエリアへのコンポーネントの追加	6-31
ユーザーインターフェイス コントロールの定義	6-38
パネルとボタン グループ定義	6-55
座標軸を定義する	6-60
テーブルを定義する	6-64
ActiveX コントロールの追加	6-75
レイアウトエリア内のコンポーネントの取り扱い	6-78
コンポーネントの位置決めと移動	6-81
コンポーネントのサイズ変更	6-84
コンポーネントの整列	6-87
配置ツール	6-87
プロパティインスペクター	6-90
グリッドとルーラ	6-94
ガイドライン	6-94
タブの順序の設定	6-96
メニューの作成	6-99
メニューバーに対してのメニュー	6-101
コンテキストメニュー	6-112

ツールバーの作成	6-120
GUIDE を使用してツールバーを作成する	6-120
ツール アイコンの編集	6-129
 オブジェクト階層の表示	6-134
 クロスプラットフォーム互換性のための設計	6-135
既定のシステム フォント	6-135
標準背景色	6-136
クロスプラットフォーム互換の Units	6-137

GUIDE GUI の保存と実行

7

GUI とそのファイルの名前付け	7-2
GUI ファイル	7-2
ファイルと GUI の名前	7-3
GUI と GUI ファイルの名前変更	7-3
 GUI の保存	7-4
GUI を保存する方法	7-4
新規の GUI の保存	7-5
既存の GUI の保存	7-8
 GUI の実行	7-10
M ファイルの実行	7-10
GUIDE レイアウトエディターから	7-10
コマンド ラインから	7-11
M ファイルから	7-11

GUIDE GUI のプログラミング

8

コールバック:概要	8-2
GUIDE を利用して作成する GUI のプログラミング	8-2

コールバックとは	8-2
コールバックの種類	8-2
GUI ファイルの概要	8-7
M ファイルと FIG-ファイル	8-7
GUI M ファイルの構造	8-8
既存の GUI M ファイルへのコールバック テンプレートの追加	8-9
GUIDE が生成するコールバックについて	8-9
コールバックとコンポーネントとの関連付け	8-11
GUI コンポーネント	8-11
コールバック プロパティを自動的に設定	8-11
GUI M ファイルからのコールバックの削除	8-14
コールバック構文と引数	8-15
コールバック テンプレート	8-15
コールバック関数の名前付け	8-16
GUIDE が割り当てるコールバックの変更	8-20
入力引数	8-21
コールバックの初期化	8-25
Opening 関数	8-25
出力関数	8-28
例:GUIDE GUI コンポーネントのプログラミング	8-30
プッシュ ボタン	8-30
トグル ボタン	8-32
ラジオ ボタン	8-32
チェック ボックス	8-33
エディット テキスト	8-34
テーブル	8-35
スライダー	8-36
リスト ボックス	8-36
ポップアップ メニュー	8-38
パネル	8-39
ボタン グループ	8-42
座標軸	8-44
ActiveX コントロール	8-48
メニュー項目	8-59

GUIDE でのアプリケーション データの管理と共有

9

データ管理の仕組み	9-2
概要	9-2
入れ子関数	9-4
UserData プロパティ	9-4
アプリケーション データ	9-5
GUI データ	9-7
GUI のコールバック間でデータを共有する例	9-10
複数の GUI を連携して動作させる方法	9-22
データを共有する方法	9-22
例 – ユーザーの入力に対するモーダル ダイアログ ボックスの 取り扱い	9-23
例 – アイコン操作ツールとして協同している個々の GUIDE GUI	9-31

GUIDE GUI の例

10

複数の座標軸をもつ GUI	10-2
複数の Axes の例について	10-2
複数の座標軸をもつ GUI の表示と実行	10-3
GUI の設計	10-5
Plot プッシュ ボタンのコールバック	10-8
ユーザー入力を数として有効にする	10-11
3-D 表示のアニメーションのための GUI	10-15
3-D アニメーションの例について	10-15
3-D 地球儀の GUI の表示と実行	10-16
GUI の設計	10-17
グラフィックスの手法	10-25
グラフィックスをさらに調べる	10-29
テーブルのデータを対話的に調べる GUI	10-32
tablestat の例について	10-32
tablestat の GUI の表示と実行	10-34

GUI の設計	10-36
Tablestat の拡張	10-52
リスト ボックス ディレクトリ リーダー	10-54
リスト ボックス ディレクトリの例について	10-54
リスト ボックス ディレクトリの GUI の表示と実行	10-55
リスト ボックス ディレクトリ GUI の実現	10-56
リスト ボックスからワークスペース変数にアクセス	10-61
ワークスペース変数の例について	10-61
ワークスペース変数の GUI の表示と実行	10-62
ワークスペース変数の読み込み	10-63
リスト ボックスからの選択項目の読み込み	10-64
Simulink モデル パラメーターを設定する GUI	10-66
Simulink モデル パラメーターの例について	10-66
Simulink パラメーター GUI の表示と実行	10-67
Simulink パラメーター GUI の利用法	10-69
GUI の起動	10-70
スライダーおよびエディット テキスト コンポーネントのプログラミング	10-71
GUI からのシミュレーションの実行	10-73
リスト ボックスから結果を削除する	10-75
結果データのプロット	10-76
GUI の [ヘルプ] ボタン	10-78
GUI のクローズ	10-78
リスト ボックス コールバックと作成関数	10-79
アドレス ブック リーダー	10-81
アドレス ブック リーダの例について	10-81
アドレス ブック リーダー GUI の表示と実行	10-82
GUI の起動	10-83
アドレス ブックをリーダーに読み込む方法	10-85
Contact Name コールバック	10-88
Contact Phone Number コールバック	10-90
アドレス ブックのページをめくる - Prev/Next	10-91
メニューからアドレス ブックに変更を保存する	10-93
新規作成メニュー	10-94
アドレス ブック サイズ変更関数	10-95
操作確認のためのモーダル ダイアログの使用	10-98
モーダル ダイアログの例について	10-98

モーダルダイアログ ボックス GUI の表示と実行	10-99
Close 確認ダイアログの設定	10-100
[Close] ボタンをもつ GUI の設定	10-101
クローズ確認の GUI の実行	10-102
閉じる確認の GUI の動作	10-103

プログラミングにより GUI を作成する

GUI のレイアウト

11 |

GUI の設計	11-2
GUI M ファイルの作成と実行	11-4
ファイル編成	11-4
ファイル テンプレート	11-4
GUI の起動	11-5
GUI の Figure の作成	11-6
GUI にコンポーネントを追加	11-9
利用可能なコンポーネント	11-9
ユーザー インターフェイス コントロールの追加	11-13
パネルとボタン グループを追加する	11-31
座標軸の追加	11-37
ActiveX コントロールの追加	11-40
対話型ツールを使用した GUI の作成およびコーディング	11-42
コンポーネントの位置の対話的な設定	11-43
コンポーネントの整列	11-52
色の対話的な設定	11-58
フォントの特性の対話的な設定	11-59
コンポーネントプロパティを設定するためのコードの生成	11-62
GUI 開発ツールのまとめ	11-67
タブの順序の設定	11-69
タブの動作仕様	11-69
既定のタブ順序	11-69

タブ順序の変更	11-72
メニューの作成	11-74
メニュー バーにメニューを追加する	11-74
コンテキスト メニューの追加	11-80
ツールバーの作成	11-87
関数 uitablebar の利用	11-87
一般に利用するプロパティ	11-87
ツールバー	11-88
標準ツールバーの表示と修正	11-91
クロスプラットフォーム互換性のための設計	11-93
既定のシステムフォント	11-93
標準背景色	11-94
クロスプラットフォーム互換の Units	11-95

GUI のプログラミング

12

はじめに	12-2
GUI の初期化	12-3
例	12-4
コールバック:概要	12-7
コールバックとは	12-7
コールバックの種類	12-8
コンポーネントに対するコールバックを与える	12-11
例:GUI コンポーネントのプログラミング	12-20
ユーザー インターフェイス コントロールのプログラミング	12-20
パネルとボタン グループのプログラミング	12-28
座標軸のプログラミング	12-30
ActiveX コントロールのプログラミング	12-34
メニュー 項目のプログラミング	12-34
ツールバー ツールのプログラミング	12-36

13

データ管理の仕組み	13-2
概要	13-2
入れ子関数	13-4
UserData プロパティ	13-5
アプリケーション データ	13-6
GUI データ	13-8
GUI のコールバック間でのデータの共有	13-11
入れ子関数を用いてデータを共有する	13-11
データを UserData と共有する	13-16
アプリケーション データとデータを共有する	13-18
GUI データとデータを共有する	13-21

コールバック実行の管理

14

コールバックの一時停止	14-2
コールバックの実行	14-2
Interruptible プロパティの動作仕様	14-2
Busy Action プロパティの動作仕様	14-4
例	14-4

プログラミングで作成する GUI の例

15

はじめに	15-2
座標軸、メニュー、ツールバーを含む GUI	15-3
座標軸、メニュー、ツールバーの例	15-3
AxesMenu ツールバー の GUI M ファイルの表示と実行	15-5
グラフ作成コマンドとデータの生成	15-6
GUI とそのコンポーネントの作成	15-7

GUI の初期化	15-12
コールバックの定義	15-12
補助関数:プロット タイプのプロット	15-16
テーブルのデータを表示してグラフを作成する GUI	15-17
tableplot の例について	15-17
tableplot の GUI M ファイルの表示と実行	15-21
uitable の設定と uitable とのやりとり	15-22
tableplot のサブ関数のまとめ	15-28
tableplot についてさらに検討する	15-28
リストのデータを管理する GUI	15-31
List Master の例について	15-31
List Master の GUI M ファイルの表示と実行	15-34
List Master の利用	15-35
List Master のプログラミング	15-40
List Master に “Import from File” オプションを追加する	15-48
List Master に “Rename List” オプションを追加する	15-48
カラー パレット	15-49
カラー パレットの例について	15-49
カラー パレットの例で用いる手法	15-53
カラー パレット GUI M ファイルの表示と実行	15-53
カラー パレットのためのサブ関数のまとめ	15-54
M ファイルの構造	15-55
GUI プログラムの手法	15-56
アイコン エディター	15-60
例について	15-60
アイコン エディターの GUI M ファイルの表示と実行	15-62
サブ関数のまとめ	15-65
M ファイルの構造	15-67
GUI プログラムの手法	15-67

例

A

Simple Examples (GUIDE)	A-2
-------------------------------	-----

Simple Examples (Programmatic)	A-2
Application Examples (GUIDE)	A-2
Programming GUI Components (GUIDE)	A-2
Application-Defined Data (GUIDE)	A-3
GUI Layout (Programmatic)	A-3
Programming GUI Components (Programmatic)	A-3
Application-Defined Data (Programmatic)	A-4
Application Examples (Programmatic)	A-4

Index

GUI 作成の紹介

章 1, MATLAB の GUI について
(p. 1-1)

GUI とは何か、GUI の動作、GUI の作成をはじめるには、などについて説明します。

章 2, GUIDE を使用して簡単な GUI を作成する (p. 2-1)

GUIDE を使用して簡単な GUI を作成するプロセスの手順を紹介します。

章 3, 簡単な GUI をプログラミングで作成する (p. 3-1)

簡単な GUI をプログラミングで作成するプロセスの手順を紹介します。

MATLAB の GUI について

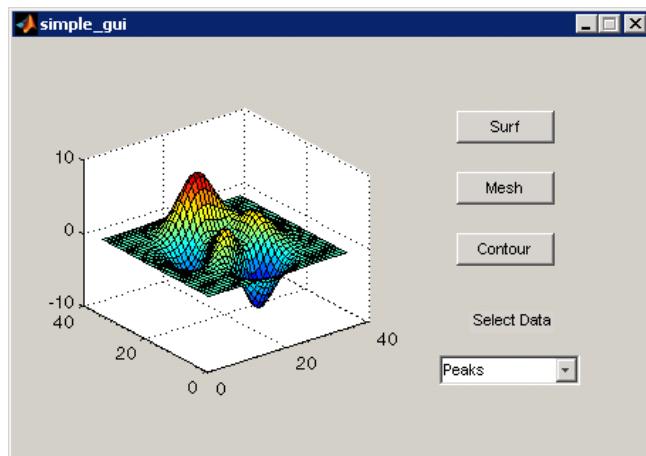
- ・ “GUI とは” (p.1-2)
- ・ “GUI の動作” (p.1-4)
- ・ “はじめに” (p.1-6)
- ・ “MATLAB GUI を作成する方法” (p.1-8)

GUI とは

グラフィカル ユーザー インターフェイス (GUI) は、コンポーネントと呼ばれるコントロールを含む 1 つ以上のウィンドウのグラフィカルな表示です。GUI を使用して、ユーザーは対話的に作業することができます。GUI のユーザーは、タスクを実行するためにスクリプトを作成したり、コマンド ラインでコマンドを入力する必要はありません。タスクを達成するプログラムのコーディングとは異なり、GUI のユーザーは、タスクがどのように実行されるかについての詳細を理解する必要がありません。

GUI コンポーネントの例として、メニュー、ツールバー、プッシュ ボタン、ラジオ ボタン、リスト ボックス、スライダーなどがあります。MATLAB のツールを使って作成される GUI は、任意の計算、データ ファイルの読み込みと書き込み、他の GUI とのやりとり、表またはプロットとしてデータを表示することなどを実行できます。

次の図は、ユーザーが容易に作成できる簡単な GUI を示します。



GUI は、以下を含みます。

- ・ 座標軸
- ・ MATLAB® 関数 peaks、membrane、sinc に対応する 3 つの異なるデータ セットをリストするポップアップ メニュー
- ・ ポップアップ メニューをラベリングするスタティック テキスト
- ・ 異なる種類のプロットを提供する 3 つのボタン:surface、mesh、contour

プッシュ ボタンをクリックすると、座標軸は指定した 3 次元プロットを用いて選択したデータを表示します。

GUI の動作

“GUI とは”(p.1-2) で述べる GUI では、ユーザーはポップアップ メニューからデータセットを選択してから、プロット タイプのボタンの 1 つをクリックします。マウスをクリックすると、選択したデータを座標軸にプロットする関数が呼び出されます。

大部分の GUI は、ユーザーが、コントロールを操作し、次に各アクションに対応するのを待ちます。各コントロールと GUI 自体は、ユーザーが記述する 1 つまたは複数のルーチン(実行可能な MATLAB コード)をもちます。このようなルーチンは、コールバックとして知られています。MATLAB が何かを実行するように、ルーチンが MATLAB に“コールバック”するので、このように呼ばれます。それぞれのコールバックの実行は、スクリーン ボタンを押す、マウス ボタンのクリック、メニュー項目の選択、文字列または数値の入力、またはコンポーネント上にカーソルを移動させるなど、特定のユーザー アクションで起こります。このとき、GUI はこれらのイベントに応答します。GUI の作成者であるユーザーは、コンポーネントがハンドル イベントに対して行うことを定義する、コールバックを提供します。

この種のプログラミングは、しばしばイベントドリブン プログラミングといわれることがあります。例では、ボタンクリックは、そのようなイベントの 1 つです。イベントドリブンのプログラミングにおいて、コールバックの実行は、ソフトウェアの外部のイベントによって起こり、非同期的です。MATLAB の GUI の場合、イベントの大部分は、ユーザーと GUI とのやりとりですが、たとえば、ファイルの作成や、デバイスをコンピューターに接続するなど他の種類のイベント対しても、GUI は同じように応答できます。

コールバックのコードは、次の 2 とおりの方法で記述できます。

- ・ M ファイルに記述され格納された MATLAB 関数
- ・ (' c = sqrt(a*a + b*b);' または ' print' など) MATLAB 表現、またはコマンドを含む文字列

M ファイルに格納された関数をコールバックとして使用すると、関数は引数にアクセスでき、よりパフォーマンスに優れフレキシブルなので、文字列を使用するよりも適切です。MATLAB スクリプト(関数を定義しない M ファイルに格納されたステートメントの列)はコールバックとして使用できません。

あるデータとともにコールバックを提供して目的の動作をさせることができます。ユーザー側でコールバックがいつ実行するかをコントロールすることはできません。つまり、GUI が使用されているとき、特定のコールバック、あるいは、その時点で実行している他のコールバックをトリガーするイベントの列をコントロールすることはできま

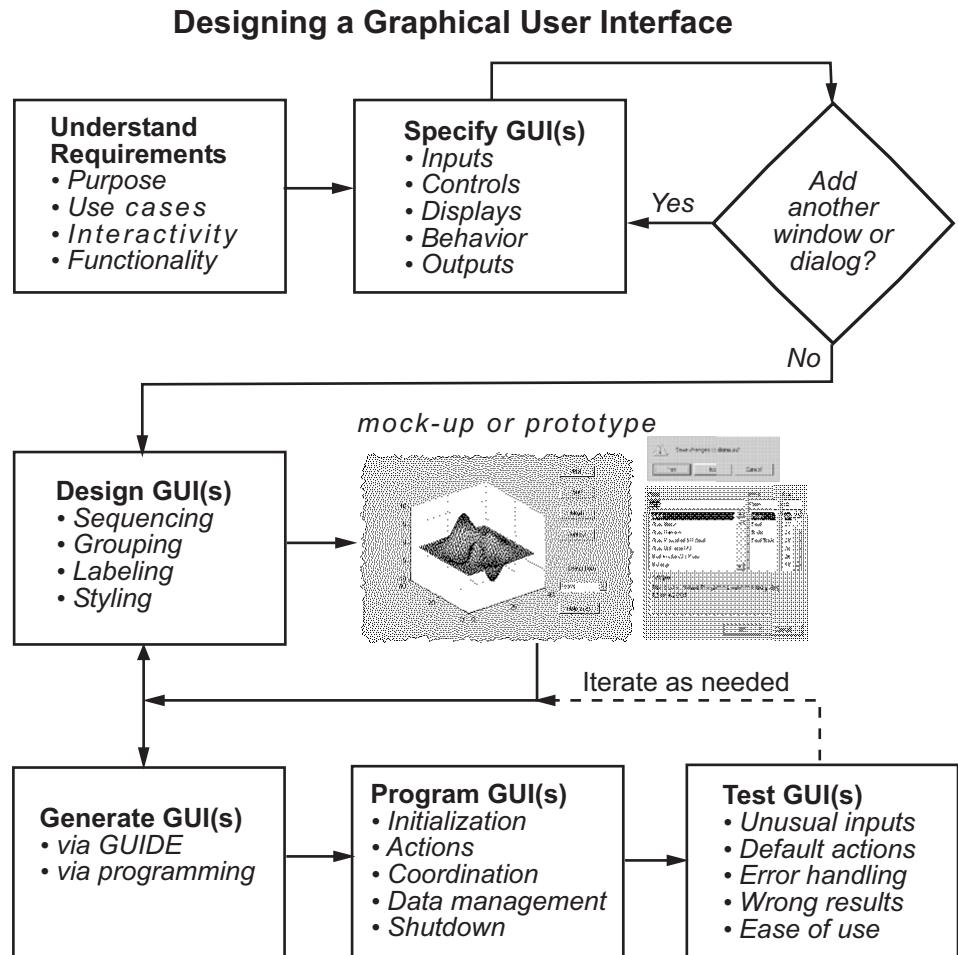
せん。この点は、イベントドリブンのプログラミングが、たとえば、一連のデータ ファイルを処理するなど、他のタイプのコントロール フローと異なる点です。

はじめに

GUI の作成を開始する前に、GUI を設計する必要があります。少なくとも、以下のことを決める必要があります。

- ・ GUI のユーザーは誰か
- ・ GUI で行うこと
- ・ ユーザーが GUI とどのようなやりとりをするか
- ・ GUI が機能するために必要となるコンポーネント

ソフトウェアを設計する場合、新規の GUI が満たすべき目的を理解する必要があります。ユーザー、あるいはその GUI を使う可能性のあるユーザーは、GUI の作成を開始する前に、ユーザー側の要求をできるだけ完全で正確に文書化する必要があります。これには、入力、出力、表示、GUI の動作、GUI がコントロールするアプリケーションの指定などが含まれます。GUI を設計後、正確で一貫した操作をするために、そのコントロールのそれぞれをプログラムする必要があります。最終的には、実際の条件下で仕様どおりに動作するかどうかを確かめるために、完成した GUI または試作した GUI をテストする必要があります。あるいは、誰か他の人がテストする方が望ましいでしょう。テストにより、設計またはプログラミングの不具合が明らかになった場合、GUI が仕様を満たす動作をするまで設計を繰り返します。次の図は、この過程の主な側面を示します。



“GUI の設計” (p.6-2) では、GUI の設計に役立ついくつかのリファレンスを一覧表示します。

さらに、GUI を作成するために使用する方法を決める必要があります。詳細は、次の節、“MATLAB GUI を作成する方法” (p.1-8) を参照してください。

MATLAB GUI を作成する方法

MATLAB の GUI は、ユーザーが操作するコントロールが追加される Figure ウィンドウです。ユーザーの好みに合わせて、コンポーネントを選択し、サイズや位置を決めることができます。コールバックを利用すると、ユーザーがコンポーネントをキーストロークでクリックまたは操作するときに、コンポーネントに目的の動作をさせることができます。

MATLAB の GUI は、以下の 2 つおりの方法で作成できます。

- ・ GUI 作成キットである GUIDE (GUI Development Environment) を対話形式で使用する
- ・ GUI を作成する M ファイルを、関数またはスクリプト (プログラミングによる GUI の作成) として作成する

最初の方法は、グラフィック レイアウト エディター内から、コンポーネントを配置する Figure から始めます。GUIDE は、GUI とそのコンポーネントに対するコールバックを含む、関連する M ファイルを作成します。GUIDE は、Figure (FIG-ファイルとして) と M ファイルを保存します。いずれか 1 つを開くと、GUI を実行するためにもう一方も開きます。

2 つめの方法として、プログラミングにより GUI を作成する方法では、すべてのコンポーネントのプロパティと動作を定義する M ファイルをコーディングします。M ファイルの実行、M ファイルによる Figure の作成、M ファイルにコンポーネントを置く、ユーザーの動作の取り扱いなどが含まれます。M ファイルが実行する度に新規の Figure が作成されるので、通常はセッションを超えて Figure は保存されません。

結果として、2 つの方法での M ファイルの外見は異なります。プログラミングによる M ファイルは、コールバックと同様、Figure とそのコントロールの各プロパティを明示的に定義するので、通常は、長くなる傾向があります。GUIDE による GUI では、Figure 自体の中にほとんどのプロパティを定義します。それらは、M ファイルではなく FIG-ファイルに定義を格納します。M ファイルは、コールバックと、M ファイルが開くときに GUI を初期化する他の関数を含みます。

MATLAB は、たとえば、ワーニングを表示したり、ファイルを開いたり保存するための、標準のダイアログ ボックスの作成を簡単にする機能も提供します。ユーザーが選択する GUI 作成方法は、ユーザーの経験、選択、GUI を操作するために必要なアプリケーションの種類に依存します。次の表は、いくつかの可能性を述べています。

GUI のタイプ	手法
ダイアログ ボックス	MATLAB では、1 度の関数呼び出しで作成できる標準的なダイアログ ボックスを選択できます。これらの関数へのリンクとして、MATLAB「関連リファレンス」ドキュメンテーションの GUI 開発の節“Predefined Dialog Boxes”を参照してください。
少数のコンポーネントのみを含む GUI	多くの場合、少数のコンポーネントのみを含む GUI をプログラミングによって作成することは、容易です。各コンポーネントは、1 度の関数呼び出しで完全に定義できます。
中程度に複雑な GUI	GUIDE により、中程度に複雑な GUI の作成が簡単になります。
多数のコンポーネントをもつ複雑な GUI、および、他の GUI との対話が必要な GUI	複雑な GUI をプログラミングにより作成すると、コンポーネントの配置を正確にコントロールし、再現性があります。

2 つの方法は、ある程度組合わせることができます。GUIDE で GUI を作成し、プログラミングで修正できます。ただし、GUI をプログラミングにより作成した後に GUIDE で修正することはできません。

どちらの手法を用いるか決めたら、以下の章の例に従い、MATLAB での GUI の作成を習得していくことができます。

- ・ 章 2, “GUIDE を使用して簡単な GUI を作成する”
- ・ 章 3, “簡単な GUI をプログラミングで作成する”

GUIDE を使用して簡単な GUI を作成する

- ・ “GUIDE:簡単な紹介” (p.2-2)
- ・ “例:簡単な GUI” (p.2-9)
- ・ “簡単な GUI のレイアウト” (p.2-11)
- ・ “GUI レイアウトを保存” (p.2-25)
- ・ “簡単な GUI のプログラミング” (p.2-28)
- ・ “GUI の起動” (p.2-36)

GUIDE:簡単な紹介

このセクションの内容…

“GUIDE とは” (p.2-2)

“GUIDE を開く” (p.2-2)

“GUI のレイアウト” (p.2-7)

“GUI のプログラミング” (p.2-8)

GUIDE とは

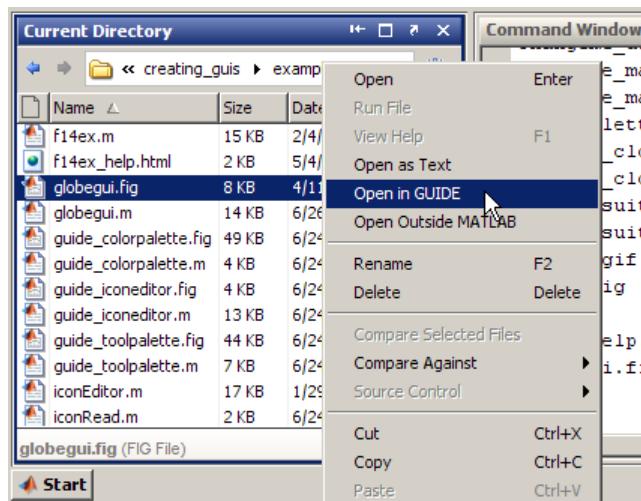
MATLAB のグラフィカル ユーザー インターフェイス開発環境である GUIDE は、グラフィカル ユーザー インターフェイス (GUI) を作成するためのツールセットを提供します。これらのツールを使うと、GUI のレイアウトやプログラミングの過程は非常に簡単にになります。

GUIDE を開く

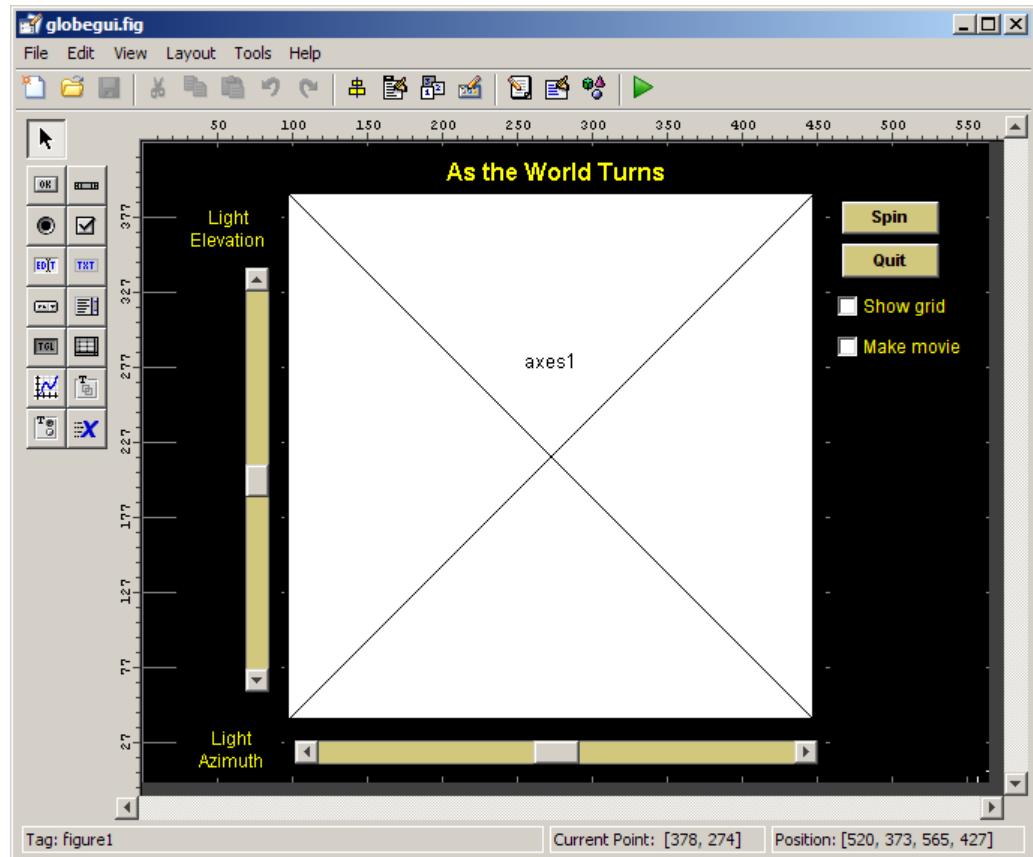
MATLAB のコマンド ラインから GUIDE を開く方法はいくつかあります。

コマンド	結果
guide	GUI のテンプレートを選択して GUIDE を開きます
guide FIG-file name	GUIDE に FIG-ファイルの name で開きます

現在のフォルダー ブラウザで FIG-ファイルを右クリックし、コンテキスト メニューから [GUIDE 内で開く] を選択することもできます。



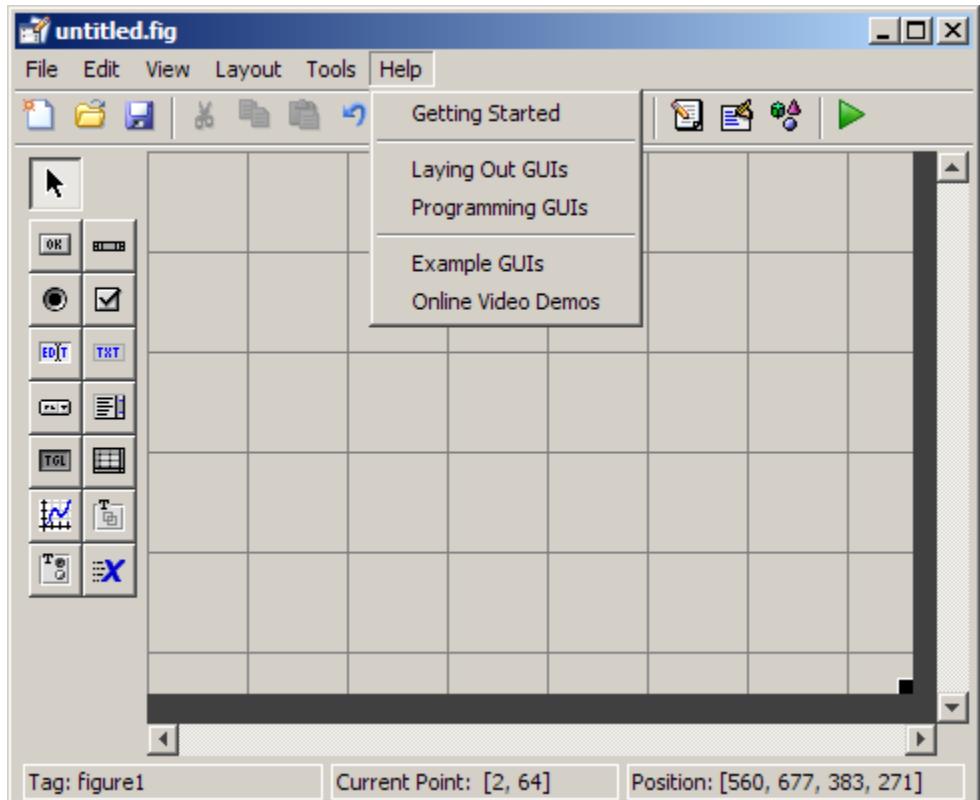
このような方法で FIG-ファイルを右クリックすると、Figure は GUIDE のレイアウトエディターを開きます。ユーザーは、このエディターで作業できます。



ツール パレットのすべてのツールには、ツールのヒントがあります。GUIDE の設定をすると、ツール名またはアイコンのみが表示され、GUIDE にパレットを表示します。詳細は、“GUIDE の設定”(p.5-2)を参照してください。

GUIDE のヘルプの取得

新規の GUI を作成するために GUIDE を開くと、グリッド付きのレイアウト エリアが表示されます。下記に示すように、レイアウト エリアの上部にはメニュー バーとツール バーがあり、左側にツール パレット、下部にステータス バーがあります。詳細は、“GUIDE ツールのまとめ”(p.4-3)を参照してください。どの点においても、次の図に示すように、GUIDE の [ヘルプ] メニューからヘルプ トピックスにアクセスできます。



最初の3つのオプションにより、GUIDEの利用をはじめる際に役立つGUIDEの説明のトピックスを参照できます。[GUIの例]オプションを用いると、GUIDEに開きブラウズや調査を行ったり実行できるGUIとして、GUIDEで作成された完成したGUIの一覧を開きます。

最も下にあるオプション[オンラインビデオデモ]からMATLAB Centralに進み、GUIDEや関連するGUI building video tutorialsの一覧を参照できます。以下の表のリンクをクリックして、MATLAB Centralと同様にMATLABのビデオデモにアクセスできます。

ビデオのタイプ	ビデオの内容
MATLAB の新機能のデモ	New Graphics and GUI Building Features in Version 7.6 (9 min, 31 s) New Graphics and GUI Building Features in Version 7.5 (2 min, 47 s) New Creating Graphical User Interfaces features in Version 7 (4 min, 24 s)
MATLAB Central ビデオ チュートリアル	2005 年から現在までの Archive for the “GUI or GUIDE” Category

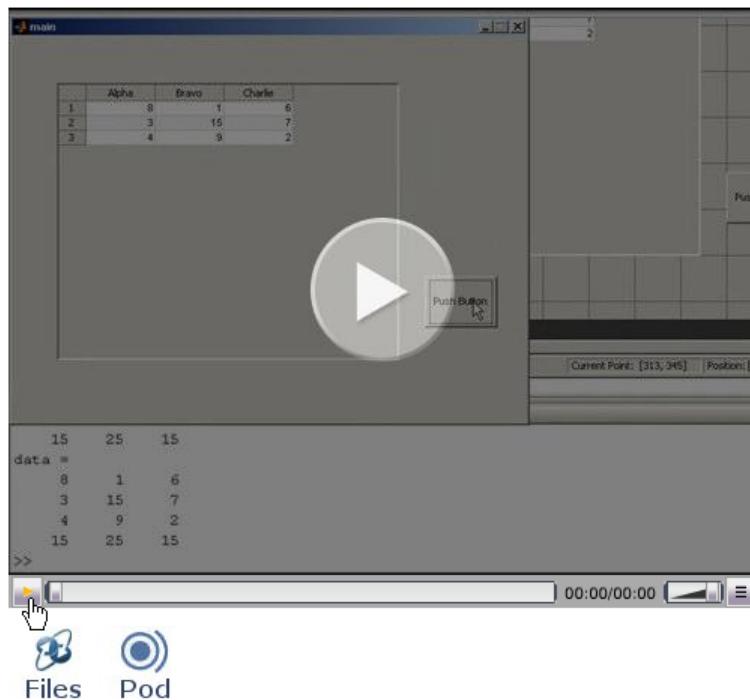
メモ システム ブラウザーで実行し、Adobe® Flash® Player プラグインが必要なビデオを実行するには、インターネットへの接続が必要です。

MATLAB Central ビデオ チュートリアルは、ほとんどが簡潔なものであり、5 分以内です。各ビデオ チュートリアルは、“Accessing data from one widget to another in GUIDE”など、特定のトピックについて述べています。このようなビデオの 1 つについて次のスクリーン ショットを参照してください(この図をクリックしても、ビデオは再生されません)。

May 13th, 2008

Accessing data from one widget to another in GUIDE

Last time, I showed how to add the newly documented UITABLE to a GUI. [click here] There were a few questions [click here] about how to access the data in the UITABLE from the callback of another widget. The answer to this question is applicable to all widgets, not just UITABLES. Basically, you are using the handles structure to access the handle of another control, then using the GET command to query it for its data.



作成者により、新しいチュートリアルが、徐々に追加されています。このページにブックマークして、時折参照するようにしてください。

GUI のレイアウト

GUIDE レイアウト エディターにより、GUI コンポーネントをレイアウトエリアにクリックしながらドラッグして GUI を配置できます。そこで、追加したボタン、テキストフィールド、スライダー、軸などのコンポーネントのサイズ変更やグループ化、整

列を行うことができます。レイアウト エディターからアクセス可能な他のツールによって、次のことができます。

- ・ メニューとコンテキスト メニューの作成
- ・ ツールバーの作成
- ・ コンポーネントの外観の修正
- ・ タブ順序の設定
- ・ コンポーネント オブジェクトの階層リストの表示
- ・ GUI オプションの設定

次のトピック “簡単な GUI のレイアウト” (p.2-11) は、これらのツールのいくつかを使用して、GUI のレイアウトの基本事項を示します。“GUIDE ツールのまとめ” (p.4-3) では、ツールについて説明します。

GUI のプログラミング

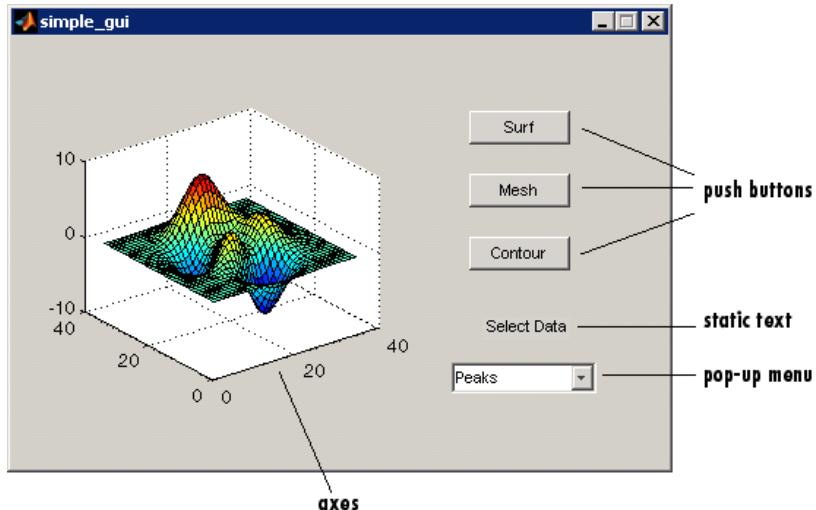
GUI レイアウトを保存すると、GUIDE は GUI の動作をコントロールするために利用可能な M ファイルを自動的に生成します。この M ファイルは、GUI を初期化するコードを提供し、GUI コールバックを構成します。コールバックは、マウス クリックなどユーザーが起こすイベントに応答して実行する関数です。M ファイル エディターを使用すると、コールバックにコードを追加してコールバックに行わせたい機能を実行させることができます。“簡単な GUI のプログラミング” (p.2-28) は、GUI が動作するために、例の M ファイルにどのようなコードを追加するかを示します。

例:簡単な GUI

このセクションの内容...
“簡単な GUI の概要” (p.2-9)
“完成したレイアウトとその GUI M ファイルの表示” (p.2-10)

簡単な GUI の概要

ここでは、次の図に示すグラフィカル ユーザー インターフェイス (GUI) を作成するための GUIDE の使用方法を示します。



GUI は、以下を含みます。

- ・ 座標軸
- ・ MATLAB 関数 peaks、membrane、sinc に対応する 3 つの異なるデータセットをリストするポップアップメニュー
- ・ ポップアップメニューをラベリングするスタティックテキスト
- ・ 異なるタイプのプロットを表示する 3 つのプッシュボタン: Surface、Mesh、Contour

GUI を使用するには、ポップアップ メニューからデータセットを選択し、プロット タイプ ボタンの 1 つをクリックします。ボタンをクリックすると選択したデータを座標軸にプロットするコールバックが実行されます。

“簡単な GUI のレイアウト”(p.2-11) ではじまる以下のトピックスでは、この GUI を作成する手順を紹介します。GUI を自分で作成することは、GUIDE を使用する方法を学習するのに最適な方法ですのでお勧めします。

完成したレイアウトとその GUI M ファイルの表示

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、この例を GUIDE レイアウトエディターと MATLAB エディターに表示できます。

メモ 以下のリンクは、MATLAB コマンドを実行し、MATLAB ヘルプ ブラウザー内で動作するように設計されています。オンラインあるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

- ・ ここをクリックすると、レイアウトエディターにこの GUI が表示されます。
- ・ MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。

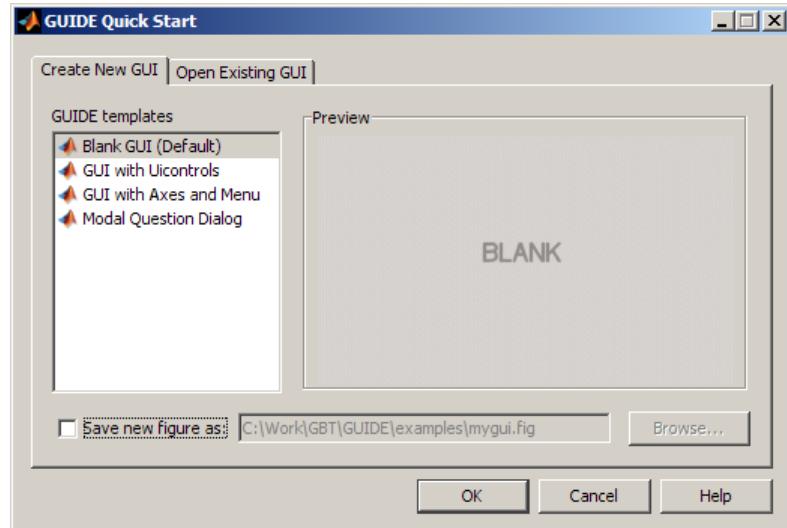
簡単な GUI のレイアウト

このセクションの内容…

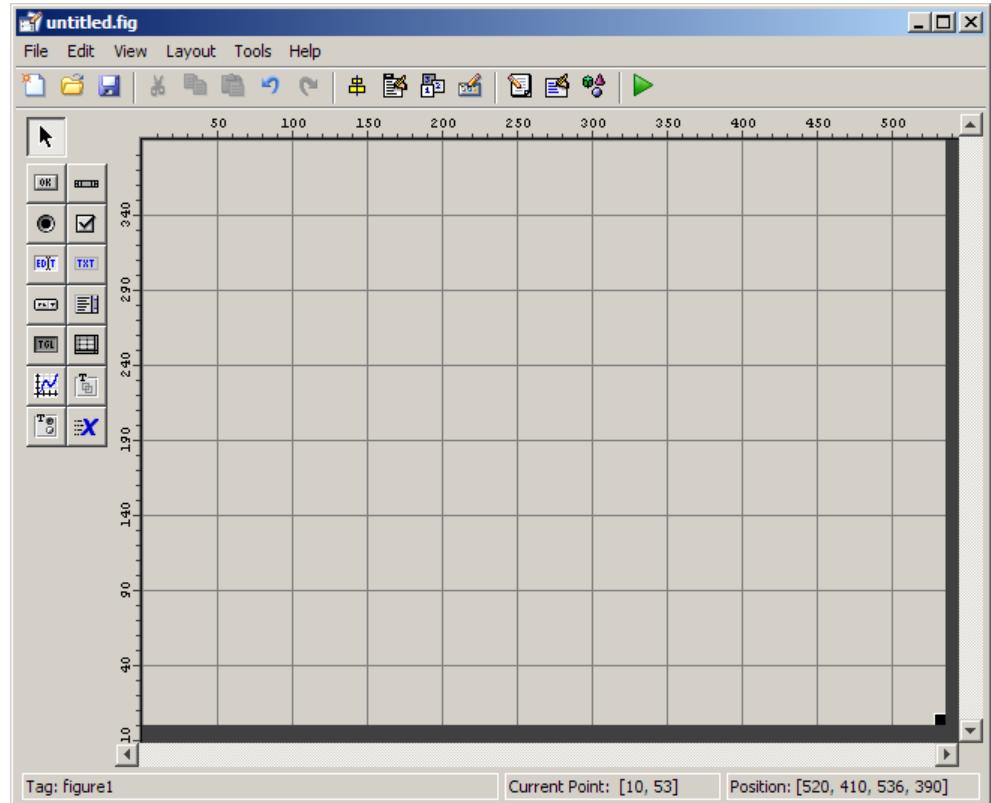
- “レイアウト エディター内に新しい GUI を開く” (p.2-11)
- “GUI の Figure サイズの設定” (p.2-14)
- “コンポーネントの追加” (p.2-15)
- “コンポーネントの整列” (p.2-16)
- “コンポーネントへのテキストの追加” (p.2-18)
- “完成したレイアウト” (p.2-23)

レイアウト エディター内に新しい GUI を開く

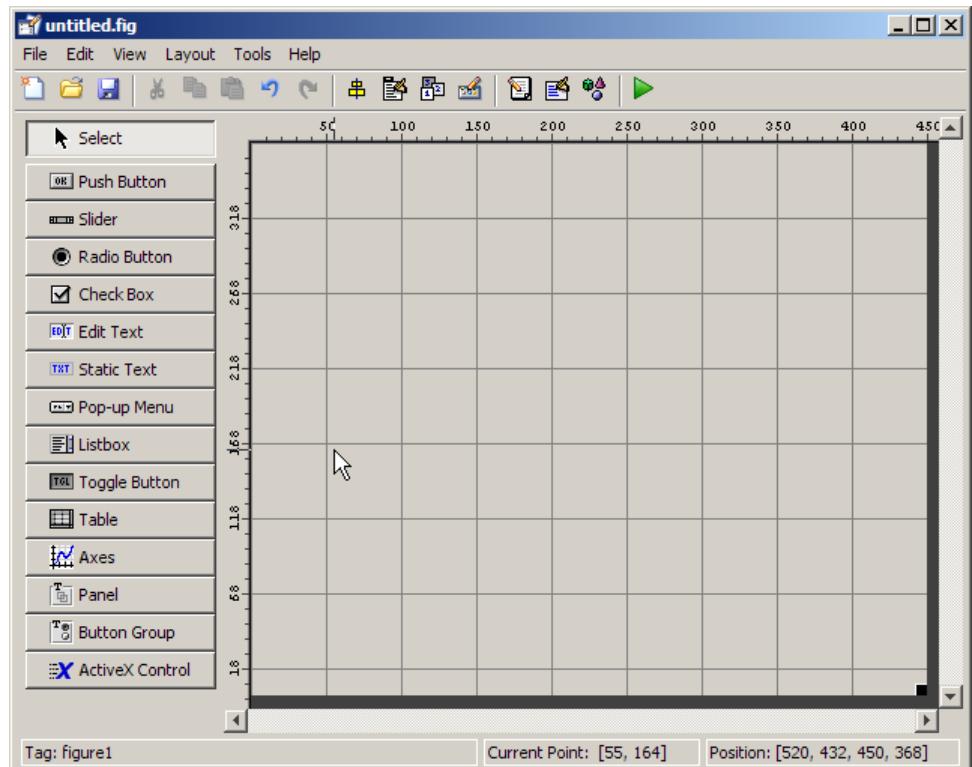
- 1 MATLAB プロンプトで `guide` と入力して、GUIDE を起動します。次の図のように [GUIDE のクイック スタート] ダイアログが表示されます。



2 [クリック スタート] ダイアログで、空白 GUI (既定値) テンプレートを選択します。[OK] をクリックすると、次の図に示すようにレイアウトエディターに空の GUI を表示します。

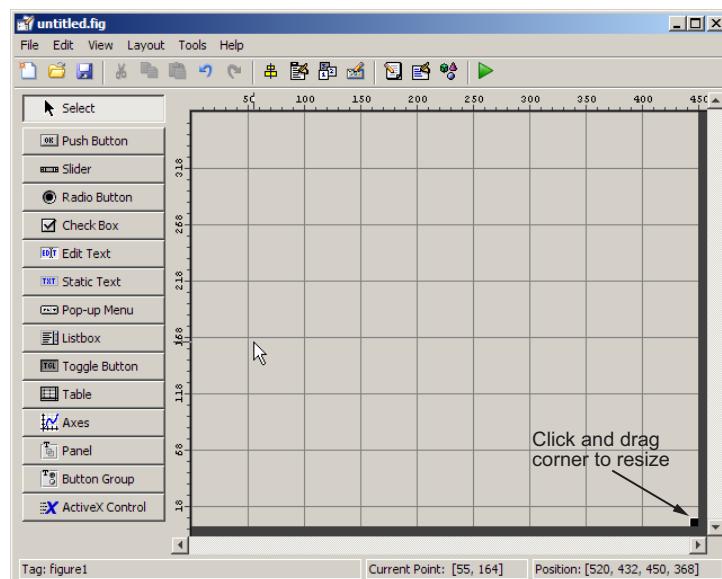


- 3 コンポーネント パレットに GUI コンポーネントの名前を表示します。MATLAB [ファイル] メニューから [設定] を選択します。[GUIDE] > [コンポーネント パレットに名前を表示] を選択し、[OK] をクリックします。次の図に示すように、レイアウト エディターが表示されます。



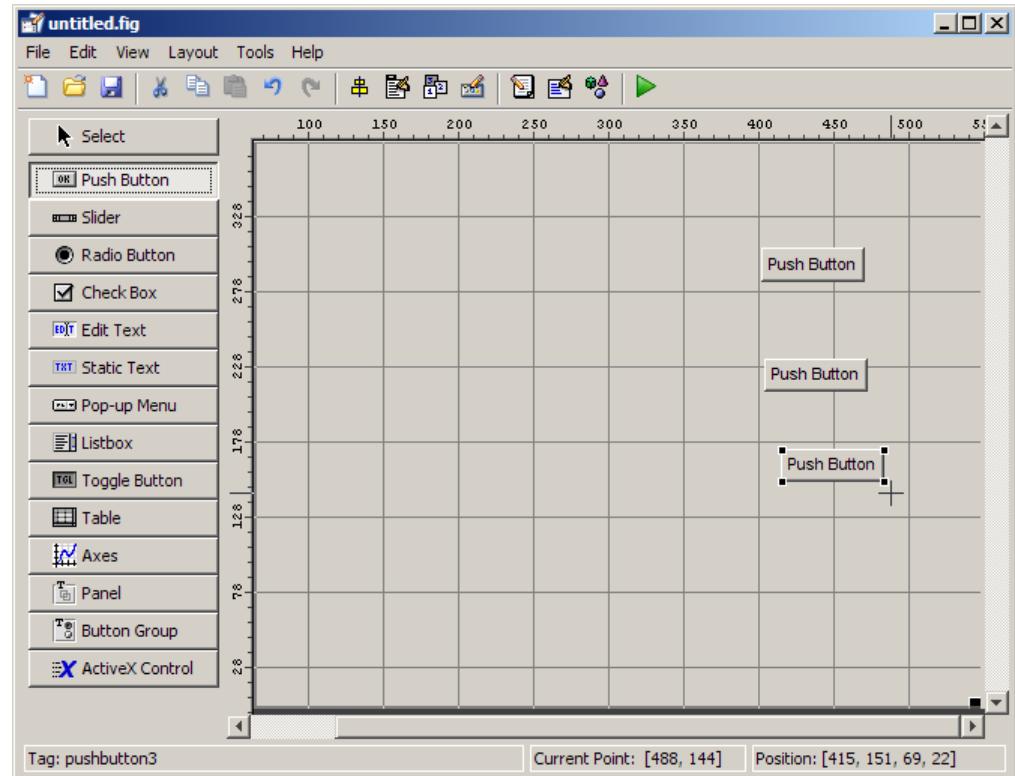
GUI の Figure サイズの設定

レイアウトエディター内のグリッドエリアをサイズ変更することで GUI のサイズを指定します。右下の角をクリックし、GUI の高さが 3 インチ幅が 4 インチくらいになるまでドラッグしてサイズ変更します。必要な場合は、ウィンドウをさらに大きくしてください。



コンポーネントの追加

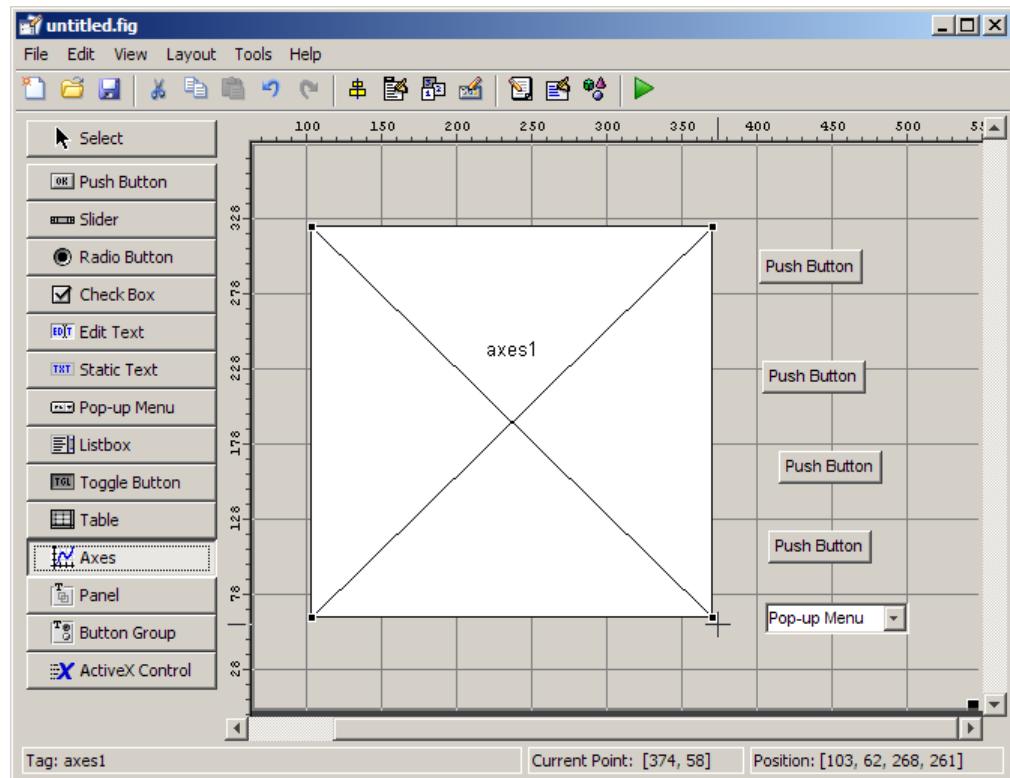
1 GUI に 3 つのプッシュ ボタンを追加します。レイアウト エディターの左でコンポーネント パレットからプッシュ ボタン ツールを選択して、レイアウト エリアにドラッグします。この方法で 3 つのボタンを作成し、次の図に示すように、およその位置に配置します。



2 残りのコンポーネントを GUI に追加します。

- ・ スタティック テキスト エリア
- ・ ポップアップ メニュー
- ・ 座標軸

次の図に示すように、コンポーネントを配置します。座標軸コンポーネントを約 2×2 インチにサイズ変更してください。



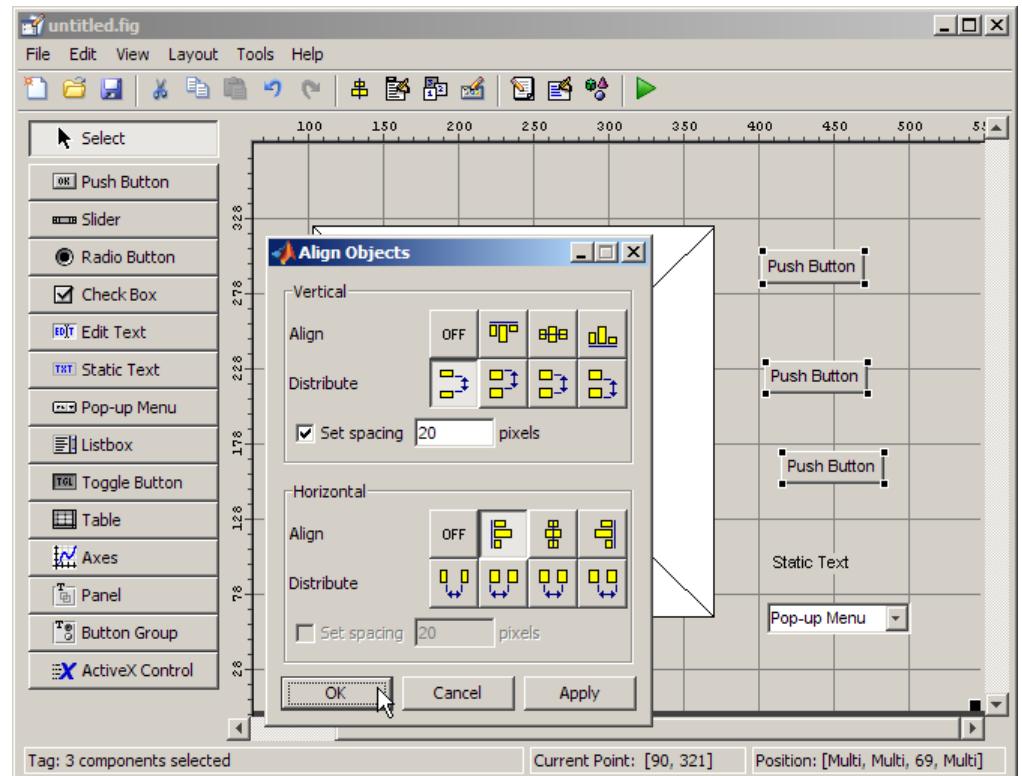
コンポーネントの整列

複数のコンポーネントが同じ親をもつ場合、整列ツールを使用して、それらを互いに整列することができます。3 つのプッシュ ボタンを整列するには、

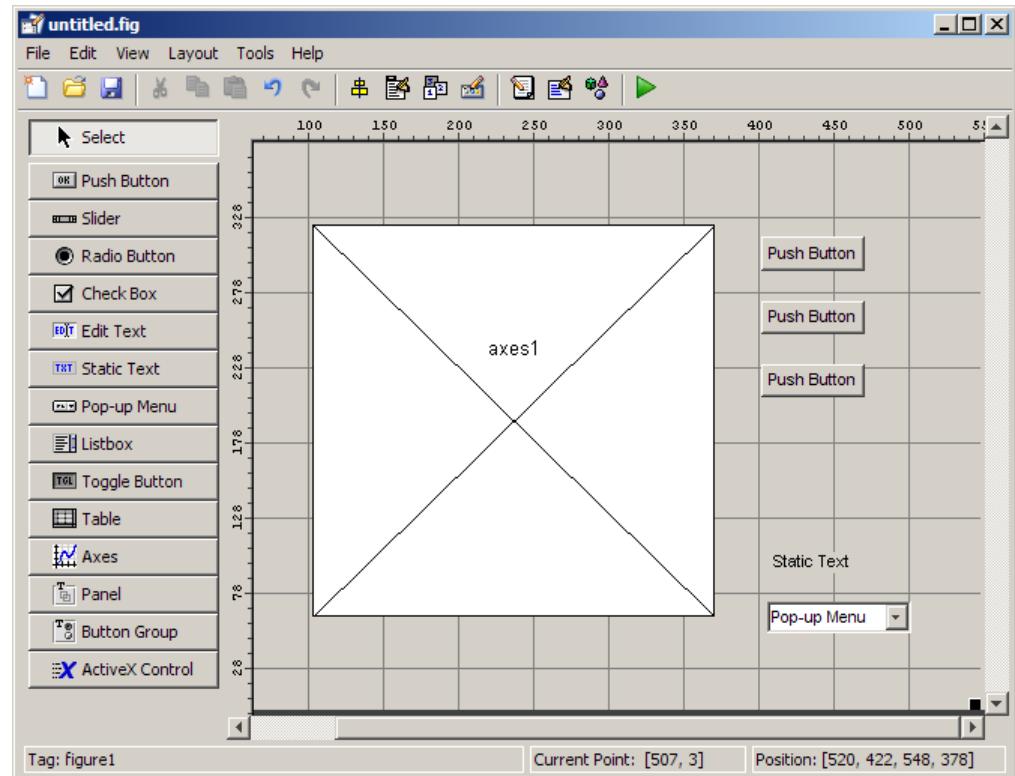
- 1 Ctrl を押し、プッシュ ボタンをクリックすることにより、3 つのプッシュ ボタンすべてを選択します。
- 2 [ツール] メニューから [オブジェクトの整列] を選択し、配置ツールを表示します。

3 次の図に示すように、配置ツールに次の設定をします。

- ・ プッシュ ボタン間の垂直方向の間隔を 20 ピクセルにします。
- ・ 水平方向には、左に整列されます。



4 [OK] をクリックします。レイアウトエディターでは GUI は次のように見えます。

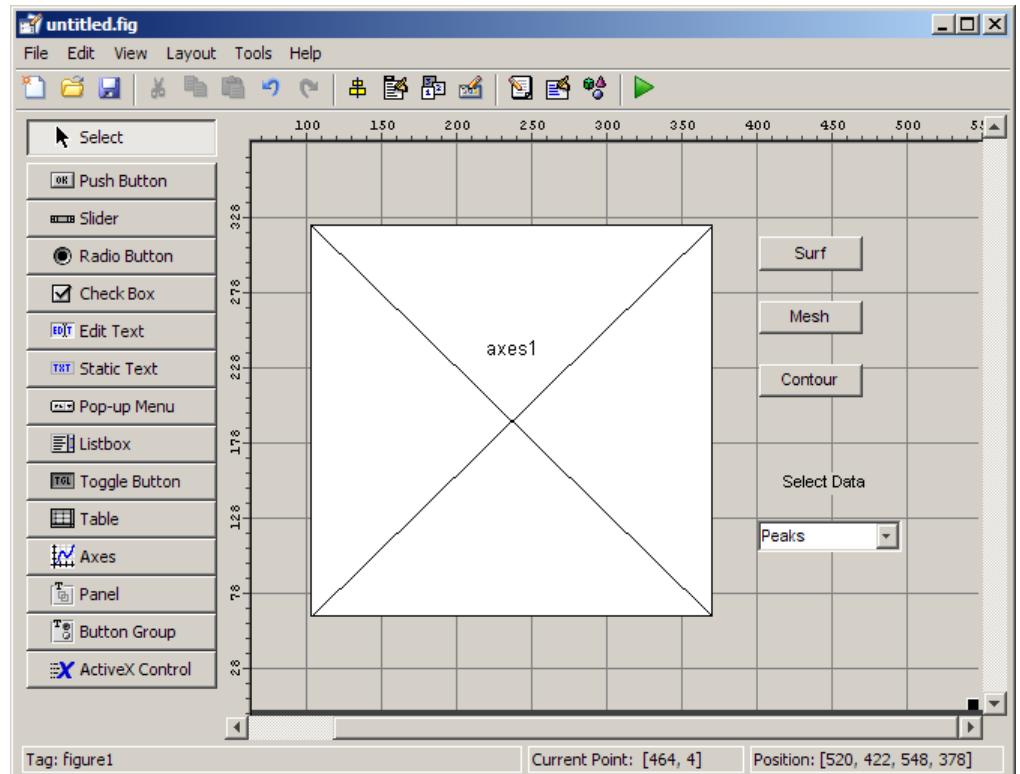


コンポーネントへのテキストの追加

プッシュボタン、ポップアップメニューおよびスタティックテキストには、作成時に既定のラベルがあります。それらのテキストは、[プッシュボタン1]など、一般的です。コンポーネントの目的を説明するように、テキストをGUIに固有になるように変更します。以下のトピックは、既定のテキストを修正する方法を示します。

- ・ “プッシュボタンのラベリング” (p.2-19)
- ・ “ポップアップメニュー項目の入力” (p.2-21)
- ・ “スタティックテキストの修正” (p.2-22)

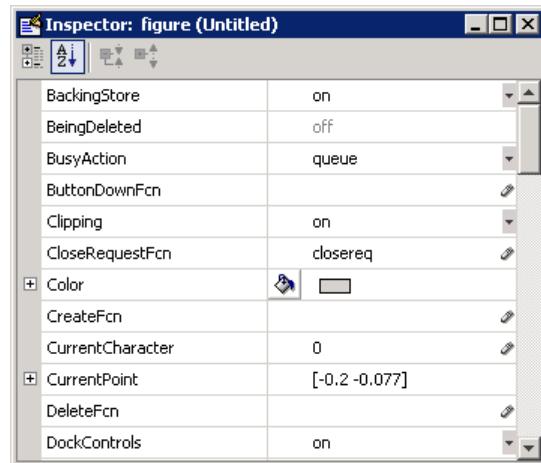
適切なテキストを追加した後、GUI はレイアウト エディターで次のようにになります。



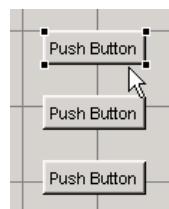
プッシュ ボタンのラベリング

3 つのプッシュ ボタンのそれぞれから、プロット タイプ surf、mesh、contour を選択できます。ここでは、これらの選択をもつボタンをラベリングする方法を示します。

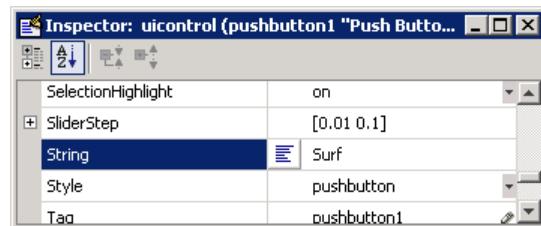
- 1 [表示] メニュー内の [プロパティ インスペクター] を選択します。



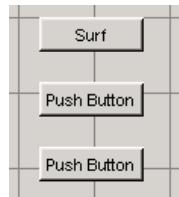
2 レイアウト エリアで、最も上にあるプッシュ ボタンをクリックして選択します。



3 プロパティ インスペクターで String プロパティを選択して、現在の値を Surf という語で置き換えます。



4 String フィールドの外側をクリックします。プッシュ ボタンのラベルが Surf に変わりります。

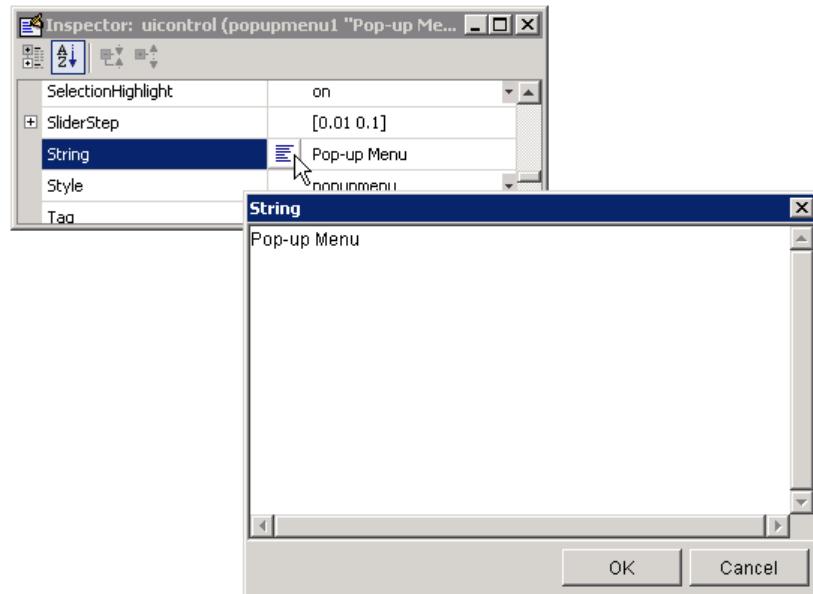


- 5 残りのプッシュボタンをそれぞれ順に選択して、手順 3 と 4 を繰り返します。中央のプッシュボタンを [Mesh]、下のプッシュボタンを [Contour] とラベルします。

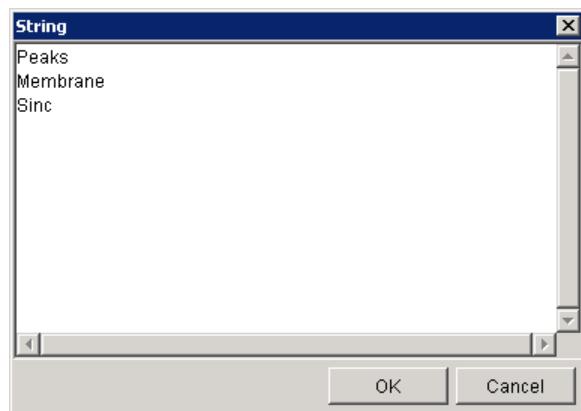
ポップアップメニュー項目の入力

ポップアップメニューには、3 つのデータセット peaks、membrane、sinc の選択肢があります。これらのデータセットは、同じ名前の MATLAB 関数に対応します。ここでは、ポップメニューの選択肢として、これらのデータセットをリストする方法を示します。

- 1 レイアウトエリアで、ポップアップメニューをクリックして選択します。
- 2 プロパティインスペクターで、String の隣のアイコンをクリックします。String ダイアログボックスが開きます。



3 3つのデータセット Peaks、Membrane、Sinc の名前で既存のテキストを置き換えます。Enter を押して次の行に移動します。



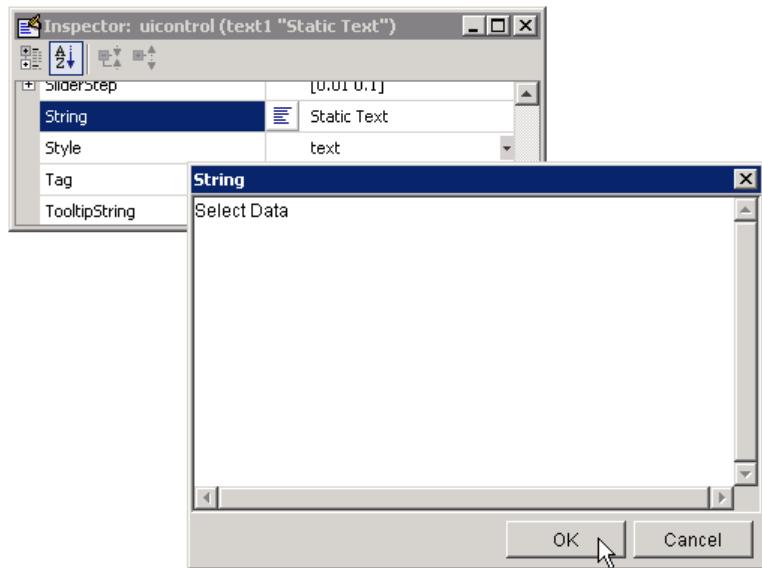
4 項目の編集が終了したら、[OK] をクリックします。リストの最初の項目 Peaks がレイアウトエリアのポップアップメニューに現れます。



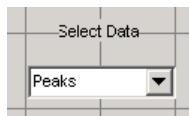
スタティック テキストの修正

この GUI では、スタティック テキストは、ポップアップメニューに対するラベルになります。ユーザーは、このテキストを変更できません。このトピックでは、スタティック テキストを変更して Select Data を読む方法を示します。

- 1 レイアウト エリアで、スタティック テキストをクリックして選択します。
- 2 プロパティ インスペクターで、String の隣のアイコンをクリックします。表示される [String] ダイアログ ボックスで、既存のテキストを Select Data という語で置き換えます。

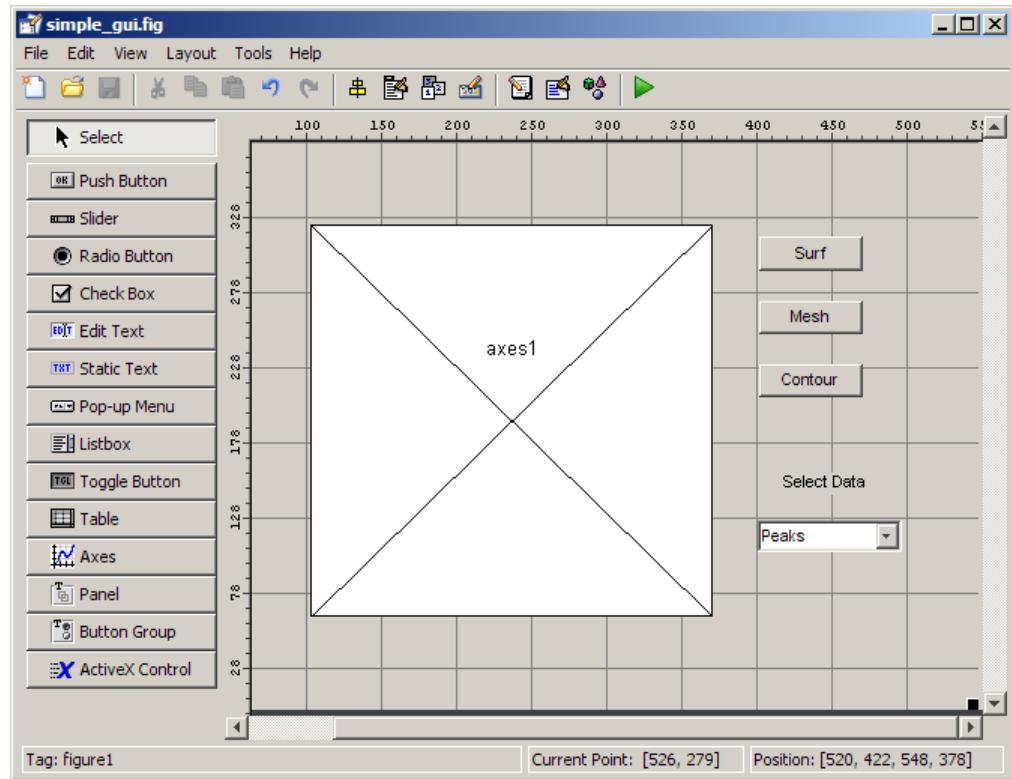


- 3 [OK] をクリックします。Select Data という語がポップアップ メニューの上のスタイルック テキスト コンポーネントに現れます。



完成したレイアウト

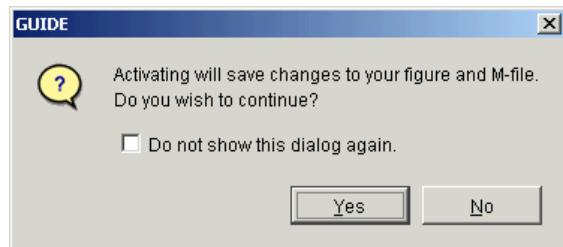
レイアウト エディターで、GUI は次のようになっています。次の手順はレイアウトを保存することです。次のトピック“GUI レイアウトを保存”(p.2-25)では、その保存方法を示します。



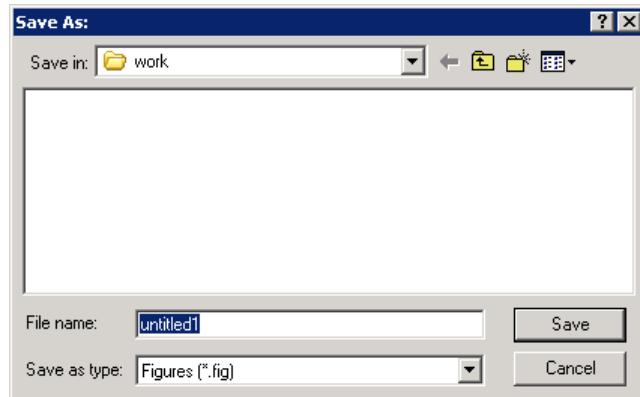
GUI レイアウトを保存

GUI を保存すると、GUIDE は 2 つのファイル FIG-ファイルと M ファイルを作成します。拡張子 .fig をもつ FIG-ファイルは、レイアウトの記述を含むバイナリファイルです。拡張子 .m をもつ M ファイルは、GUI をコントロールするコードを含みます。

- 1 [ツール] メニューから [実行] を選択することによって、GUI を保存し、アクティブにします。
- 2 GUIDE は、次のダイアログ ボックスを表示します。[Yes] をクリックして続行します。

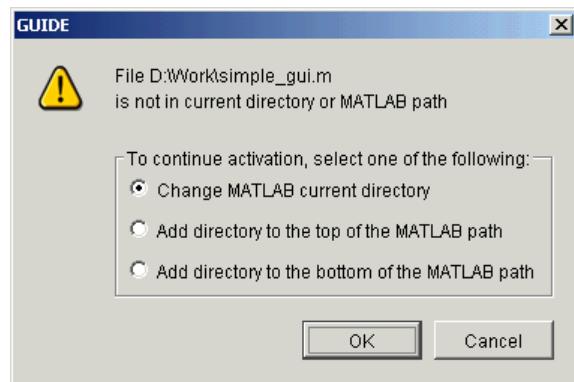


- 3 GUIDE は、現在のフォルダーで [別名で保存] ダイアログ ボックスを開き、FIG-ファイル名の入力が可能になります。



- 4 書き込み権限を持つフォルダーを参照して、FIG-ファイルに対するファイル名 simple_gui を入力します。GUIDE はこの名前を用いて FIG-ファイルと M ファイルの両方を保存します。

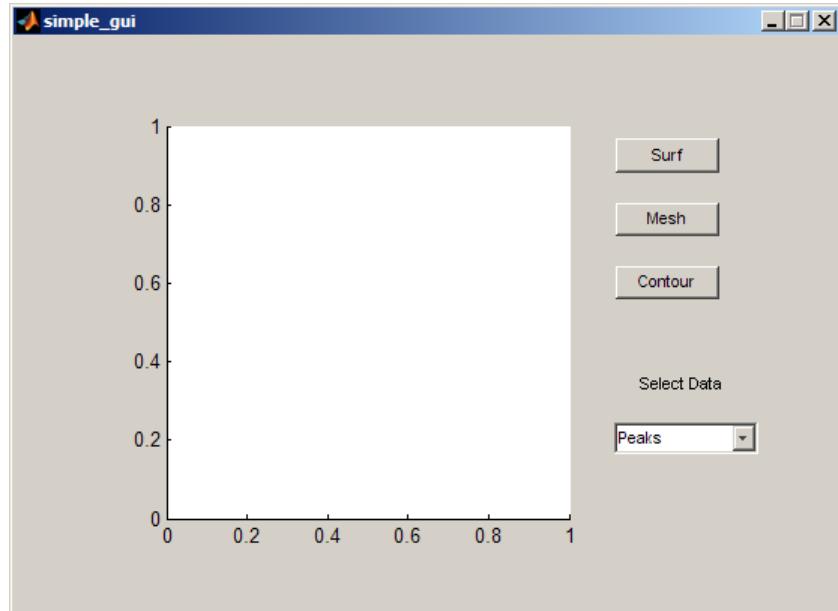
5 GUI を保存するフォルダーが MATLAB パス上にない場合、GUIDE は、現在のフォルダーを GUI ファイルを含むフォルダーに移動するか、そのフォルダーを MATLAB パスに追加するオプションを与える次のダイアログを開きます。



6 GUIDE はファイル simple_gui.fig と simple_gui.m を保存し、GUI をアクティブにします。さらに、MATLAB エディターに GUI M ファイルを開きます。

GUI が新規ウィンドウに開きます。この GUI には、MATLAB Figure ウィンドウが表示する標準のメニュー バーとツールバーがありません。GUIDE にユーザー独自のメニューとツールバー ボタンを追加できますが、既定では GUIDE GUI にはこれらのコンポーネントは何も含まれません。

simple_gui を操作する場合、ポップアップ メニューでデータ セットを選択してプッシュ ボタンをクリックできますが、何も起りません。なぜなら、ポップアップ メニューとボタンに対するコードが M ファイルに含まれていないためです。次のトピックス “簡単な GUI のプログラミング” (p.2-28) では、そのコントロールが機能するため GUI をどのようにプログラムするかについて説明します。



GUIDE を開かず GUIDE で作成された GUI を実行するには、M ファイル名を入力して M ファイルを実行します。

```
simple_gui
```

たとえば、M ファイルで run コマンドを使用することもできます。

```
run simple_gui
```

メモ GUIDE の外部で FIG-ファイルを開くことで、GUIDE GUI を実行しようとしないでください。そのようにすると、Figure が開き、使用できる準備ができているように見えますが、GUI は初期化せず、そのコールバックは機能しません。

簡単な GUI のプログラミング

このセクションの内容…

“M ファイルへのコードの追加” (p.2-28)

“プロットするデータの生成” (p.2-28)

“ポップアップ メニューのプログラミング” (p.2-31)

“プッシュ ボタンのプログラミング” (p.2-33)

M ファイルへのコードの追加

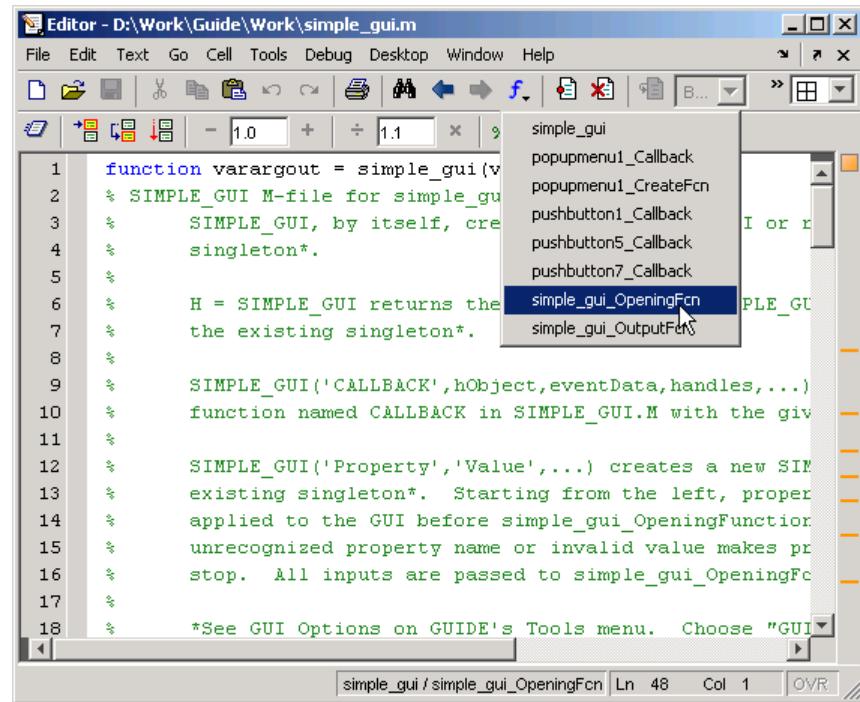
前のトピック “GUI レイアウトを保存” (p.2-25) で GUI を保存したときに、GUIDE は 2 つのファイルを作成しました。これらのファイルは、GUI レイアウトを含む FIG-ファイル simple_gui.fig と、GUI の動作を決めるコードを含む M ファイル simple_gui.m です。M ファイルは、MATLAB 関数 (つまり、スクリプトではない) のセットで構成されます。しかし、関数には GUI を動作させるコードが含まれていないので GUI は応答しませんでした。以下のトピックスでは、GUI が機能するように、M ファイルにどのようにコードを追加するかについて説明します。3 つの手順があります。

プロットするデータの生成

このトピックは、GUI ユーザーがボタンをクリックするときにプロットされるデータを作成する方法を示します。opening 関数は、MATLAB 関数を呼び出してこのデータを生成します。GUI が開くときにこれを初期化する opening 関数は、GUIDE が生成する各 GUI M ファイル内の最初のコールバックです。

この例では、opening 関数に 3 つのデータセットを作成するコードを追加します。このコードは、MATLAB 関数 peaks、membrane、sinc を使用します。

- 1 M ファイル エディターで opening 関数を表示します。GUIDE M ファイル simple_gui.m がまだエディターに開いていない場合、[表示] メニューから [M ファイル エディター] を選択して開きます。エディターでは、ツールバーの 関数アイコン  をクリックしてから、表示されるポップアップ メニューで [simple_gui_OpeningFcn] を選択します。



カーソルが、以下のコードを既に含む opening 関数に移動します。

```

% --- Executes just before simple_gui is made visible.
function simple_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to simple_gui (see VARARGIN)

% Choose default command line output for simple_gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes simple_gui wait for user response (see UIRESUME)

```

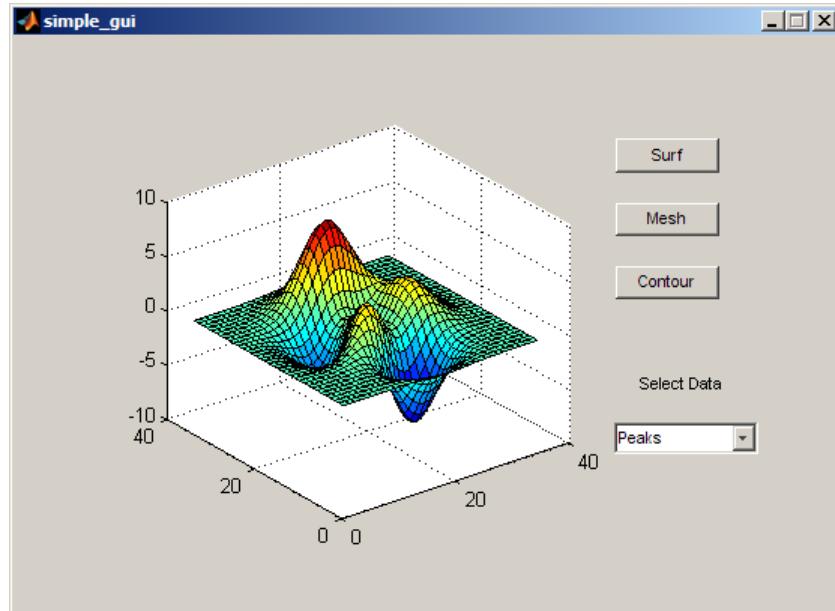
```
% uicontrol(handles.figure1);
```

2 opening 関数で % varargin... ではじまるコメントの直後に以下のコードを追加して GUI に対するデータを作成し、プロットします。

```
% Create the data to plot.  
handles.peaks=peaks(35);  
handles.membrane=membrane;  
[x,y] = meshgrid(-8:.5:8);  
r = sqrt(x.^2+y.^2) + eps;  
sinc = sin(r)./r;  
handles.sinc = sinc;  
% Set the current data value.  
handles.current_data = handles.peaks;  
surf(handles.current_data)
```

最初の実行可能な 6 行では、MATLAB 関数 peaks、membrane、sinc を使用してデータを作成します。これらが、データを、すべてのコールバックに対して与えられる引数である handles 構造体に保存します。プッシュ ボタンに対するコールバックは、handles 構造体からデータを取得できます。

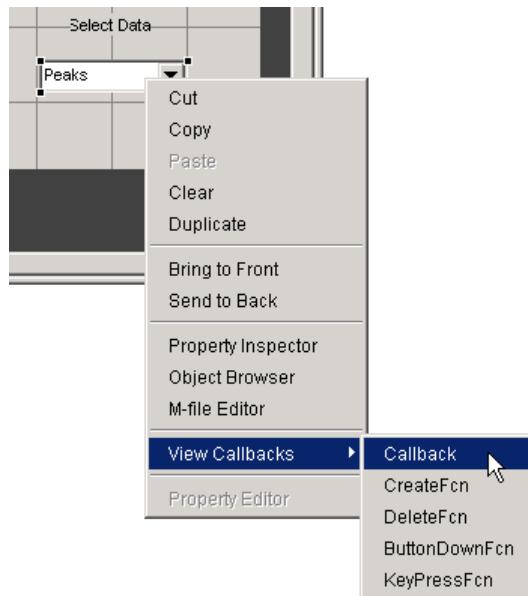
最後の 2 行は現在のデータ値を作成し、peaks に設定してから peaks に対する surf プロットを表示します。次の Figure は、GUI が最初に表示されるとき、どのような外見をもつか示します。



ポップアップ メニューのプログラミング

ポップアップ メニューにより、プロットするデータを選択することができます。3 つのプロットのいずれか 1 つを選択すると、MATLAB はポップアップ メニューの Value プロパティを選択された文字列のインデックスに設定します。ポップアップ メニュー コールバックは、ポップアップ メニューの Value プロパティを読み込み、メニューに現在表示されている項目を判別して、それに従い handles.current_data を設定します。

- 1 M ファイル編集エディターでポップアップ メニュー コールバックを表示します。
レイアウト エディターでポップアップ メニュー コンポーネントを右クリックして、
コンテキスト メニューを表示します。そのメニューから、[コールバックの表
示] > [Callback] を選択します。



エディターがまだ開いていない場合には、GUIDE がこれを開いて GUI M ファイルを表示し、すでに以下のコードを含むポップアップメニュー コールバックに、カーソルが移動します。

```
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

2 % handles... ではじまるコメントの後に、popupmenu1_Callback に対する以下のコードを追加します。

次のコードは、最初にポップアップメニューの 2 つのプロパティを取得します。

- String – メニューの内容を含むセル配列
- Value – 選択したデータセットのメニューのコンテンツへのインデックス

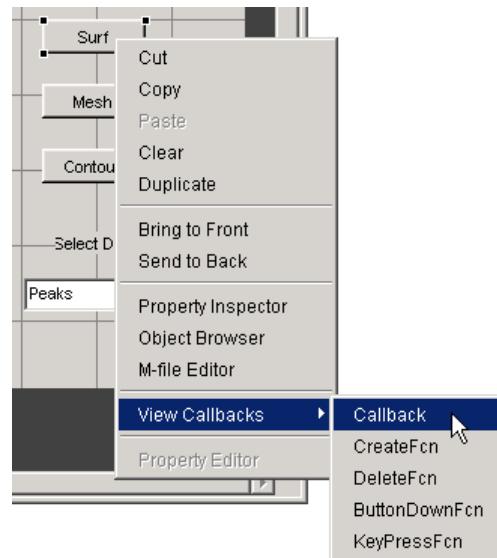
そして、switch ステートメントを使用して、選択したデータを現在のデータに設定します。最後のステートメントは、handles 構造体への変更を保存します。

```
% Determine the selected data set.  
str = get(hObject, 'String');  
val = get(hObject, 'Value');  
% Set current data to the selected data set.  
switch str{val};  
case 'Peaks' % User selects peaks.  
    handles.current_data = handles.peaks;  
case 'Membrane' % User selects membrane.  
    handles.current_data = handles.membrane;  
case 'Sinc' % User selects sinc.  
    handles.current_data = handles.sinc;  
end  
% Save the handles structure.  
guidata(hObject, handles)
```

プッシュ ボタンのプログラミング

それぞれのプッシュ ボタンは、ポップアップ メニューの現在の選択で指定されたデータを使用して、異なるタイプのプロットを作成します。プッシュ ボタン コールバックは、handles構造体からデータを取得し、プロットします。

- 1 [Surf] プッシュボタン コールバックを M ファイル エディターに表示します。レイアウトエディターで、[Surf] プッシュボタンを右クリックして、コンテキストメニューを表示します。そのメニューから、[コールバックの表示] > [Callback] を選択します。



エディターで、すでに以下のコードを含む GUI M ファイル内の Surf プッシュボタン コールバックにカーソルが移動します。

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

- 2 % handles... ではじまるコメントの直後のコールバックに次のコードを追加します。

```
% Display surf plot of the currently selected data.
surf(handles.current_data);
```

- 3 手順 1 と 2 を繰り返して、Mesh と Contour プッシュボタン コールバックに同様のコードを追加します。

- 次のコードを Mesh プッシュボタン コールバック pushbutton2_Callback に追加します。

```
% Display mesh plot of the currently selected data.  
mesh(handles.current_data);
```

- 次のコードを Contour プッシュ ボタン コールバック pushbutton3_Callback に追加します。

```
% Display contour plot of the currently selected data.  
contour(handles.current_data);
```

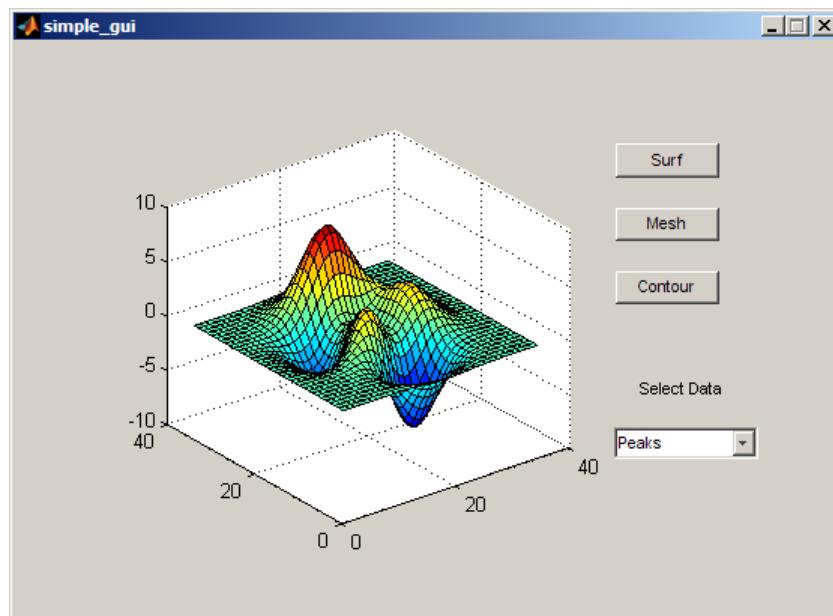
4 [ファイル] メニューから [保存] を選択して M ファイルを保存します。

GUI を実行する準備ができました。次のトピック “GUI の起動” (p.2-36) では、その方法を示します。

GUI の起動

“簡単な GUI のプログラミング”(p.2-28)では、ポップアップメニューとプッシュボタンについてプログラムを作成しました。さらに、これらのためのデータを作成し、表示を初期化しました。ここで、その GUI を実行し、動作を見ることができます。

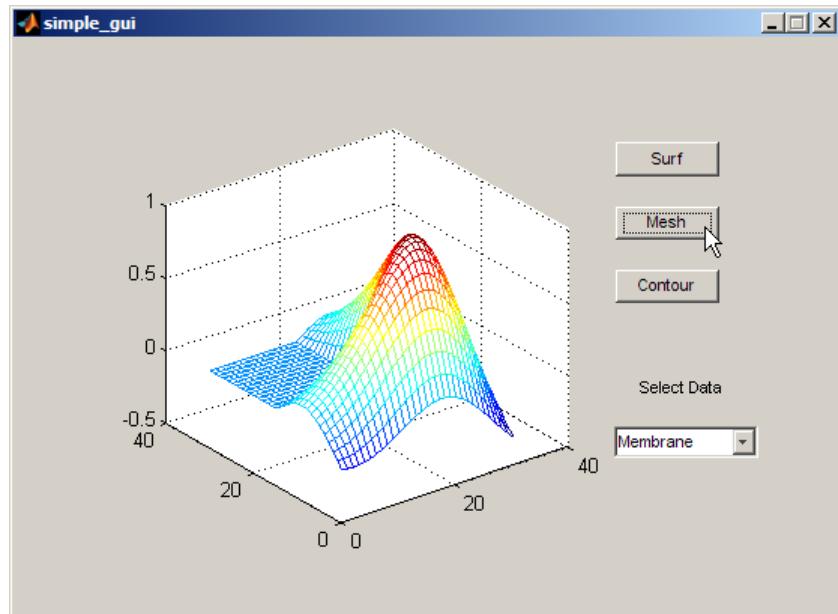
- 1 レイアウトエディターの [ツール] メニューから [実行] を選択して、GUI を実行します。GUI が MATLAB パス上または現在のフォルダー内にある場合、プロンプトで GUI の名前 `simple_gui` を入力することでも実行できます。GUI は最初に表示されたとき、次のように見えます。



通常の Figure と異なり、この GUI にはメニューバーもツールバーも表示されないことに注目してください。既定では、大部分の GUI で、標準のメニュー項目およびボタンが提供するすべての操作がサポートされているわけではないため、これらを省略します。ただし、標準の Figure ツールバーとメニューバーをオンにする、あるいは GUIDE メニュー エディターとツールバー エディターを用いてカスタムのものを作成することもできます。さらに、既定では、通常の Figure の場合とは異なり、GUIDE で作成された GUI には MATLAB デスクトップでドックする機能がありません。GUI のドックをコントロールできますが、GUI はそれらを有効にするメニュー

バーまたはツールバーを表示する必要があります。詳細は、“メニューが Figure のドックにどのように影響するか”(p.6-101)を参照してください。

- 2 ポップアップメニューで、[Membrane] を選択してから、[Mesh] ボタンをクリックします。GUI は The MathWorks™ L-shaped Membrane logo™ のメッシュプロットを表示します。



- 3 GUI を閉じる前に他の組合せのいくつかを試します。

追加のコントロールの機能がある同じような GUIDE GUI の例は、“多くのコンポーネントをもつ GUI の取り扱い”(p.6-24)を参照してください。

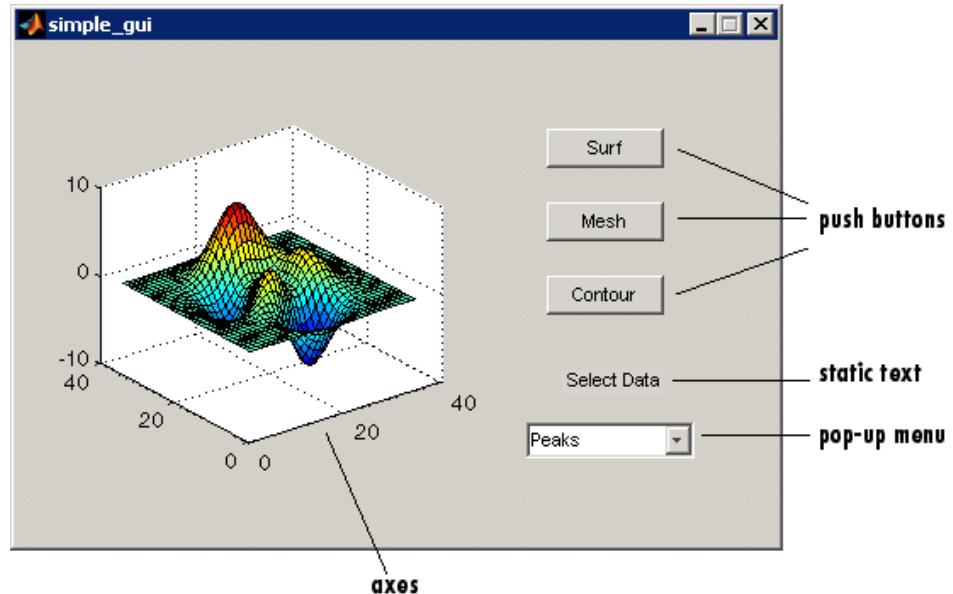
簡単な GUI をプログラミング で作成する

- ・ “例:簡単な GUI” (p.3-2)
- ・ “関数のまとめ” (p.3-4)
- ・ “GUI M ファイルの作成” (p.3-6)
- ・ “簡単な GUI のレイアウト” (p.3-8)
- ・ “GUI の初期化” (p.3-12)
- ・ “GUI のプログラミング” (p.3-15)
- ・ “GUI の起動” (p.3-18)

例:簡単な GUI

簡単な GUI の概要

ここでは、下図に示すグラフィカル ユーザー インターフェイス (GUI) の例を作成する M コードを記述する方法を示します。



GUI は、以下を含みます。

- ・ 座標軸
- ・ MATLAB 関数 peaks、membrane、sinc に対応する 3 つの異なるデータセットをリストするポップアップメニュー
- ・ ポップアップメニューをラベリングするスタティックテキスト
- ・ 異なる種類のプロット surface、mesh、contour を提供する 3 つのプッシュボタン

GUI を使用するには、ポップアップメニューからデータセットを選択してから、プロットタイプのプッシュボタンの 1 つをクリックします。ボタンをクリックすると選択したデータを座標軸にプロットするコールバックが実行されます。

次のトピックス、“関数のまとめ”(p.3-4)は、この GUI を作成する例に使用される関数をまとめます。

以下のトピックスは、GUI の作成プロセスを紹介します。このプロセスは、“GUI M ファイルの作成”(p.3-6)からはじめます。この GUI は実際に自分で作成することをお勧めします。

完成した例の表示

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の GUI とその M ファイルを表示できます。

メモ 以下のリンクは、MATLAB コマンドを実行し、MATLAB ヘルプ ブラウザー内で動作するように設計されています。オンラインあるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

- ・ ここをクリックすると、例の GUI が表示されます
- ・ MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。

関数のまとめ

MATLAB は、GUI を作成するための一組の関数を提供します。ここでは、以下の図に示す例の GUI を作成するために必要な関数を紹介します。関数の説明を参照する場合、関数名をクリックしてください。

簡単な GUI の作成に使用する関数

関数	説明
align	ユーザー インターフェイス コントロールや座標軸などの GUI コンポーネントの整列。
axes	Axes オブジェクトの作成。
figure	Figure オブジェクトの作成。GUI は Figure オブジェクトです。
movegui	GUI の Figure をスクリーン上の指定した位置まで移動します。
uicontrol	プッシュ ボタン、スタティック テキスト、ポップアップメニューなど、ユーザー インターフェイス コントロール オブジェクトを作成します。

GUI のプログラミングで用いられる他の MATLAB 関数

関数	説明
contour	行列の等高線図
eps	浮動小数点数間距離
get	オブジェクト プロパティの取得
membrane	MATLAB ロゴ (関数 demo) で使用するデータの作成
mesh	メッシュ プロット
meshgrid	3 次元プロットのための配列 X と Y の作成
peaks	2 変数関数の例
set	オブジェクト プロパティの設定

GUI のプログラミングで用いられる他の MATLAB 関数 (続き)

関数	説明
sin	正弦。結果の単位はラジアン
sqrt	平方根
surf	3 次元影付き表面プロット

GUI M ファイルの作成

例の GUI の M ファイルを作成することから始めます。このファイルは関数を含むので、スクリプト M ファイルではなく、関数 M ファイルです。スクリプト M ファイルは、MATLAB コマンドの列を含みますが、関数を定義しません。

- 1 MATLAB プロンプトで、`edit` と入力すると、MATLAB はエディターを開きます。
- 2 エディターに次のステートメントを入力またはコピーします。次の `function` ステートメントはファイルの 1 行目に記述します。

```
function simple_gui2
```

- 3 `function` ステートメントに続いて、M ファイルに以下のコメントを追加します。これらは、`help` コマンドに応答してコマンド ラインに表示されます。コメントの後には、空行を続けます。

```
% SIMPLE_GUI2 Select a data set from the pop-up menu, then  
% click one of the plot-type push buttons. Clicking the button  
% plots the selected data in the axes.  
(Leave a blank line here)
```

- 4 ファイルの最後に `end` ステートメントを追加します。この `end` ステートメントは、`function` ステートメントに対応します。ここで、ユーザーのファイルは次のようにになります。

```
function simple_gui2  
% SIMPLE_GUI2 Select a data set from the pop-up menu, then  
% click one of the plot-type push buttons. Clicking the button  
% plots the selected data in the axes.  
  
end
```

メモ 例は入れ子関数を用いて記述されているので、`end` ステートメントが必要です。入れ子関数の詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”について参照してください。

- 5 そのファイルをユーザーの現在のフォルダーあるいは MATLAB パス上に保存します。

次の節“簡単な GUI のレイアウト”(p.3-8)は、ユーザー GUI にコンポーネントを追加する方法を示します。

簡単な GUI のレイアウト

このセクションの内容…

“Figure の作成” (p.3-8)

“コンポーネントの追加” (p.3-8)

Figure の作成

MATLABにおいて、GUIはFigureです。この最初の手順ではFigureを作成し、スクリーン上に配置します。この手順は、GUIを非可視にもするため、GUIユーザーは追加、あるいは初期化されたコンポーネントを見ることができません。GUIがすべてのコンポーネントをもち初期化されると、表示されます。

```
% Initialize and hide the GUI as it is being constructed.  
f = figure('Visible','off','Position',[360,500,450,285]);
```

関数 `figure` を呼び出す場合、2つのプロパティ/値の組を指定します。`Position` プロパティは、スクリーン上の GUI の位置とサイズを指定する 4 要素ベクトル [左端からの距離、下端からの距離、幅、高さ] です。既定の単位はピクセルです。

次のトピックス、“コンポーネントの追加” (p.3-8) は、GUI にプッシュボタン、座標軸、他のコンポーネントを追加する方法を示します。

コンポーネントの追加

例の GUI は 6 つのコンポーネント (3 つのプッシュボタン、1 つのスタティックテキスト、1 つのポップアップメニュー、1 つの座標軸) をもちます。これらのコンポーネントを GUI に追加するステートメントを記述することから始めます。関数 `uicontrol` を用いて、プッシュボタン、スタティックテキスト、ポップアップメニューを作成します。座標軸を作成するには、関数 `axes` を使用します。

1 `figure` への呼び出しに続き、M ファイルに次のステートメントを追加して、GUI に 3 つのプッシュボタンを追加します。

```
% Construct the components.  
hsurf = uicontrol('Style','pushbutton',...  
    'String','Surf','Position',[315,220,70,25]);  
hmesh = uicontrol('Style','pushbutton',...  
    'String','Mesh','Position',[315,180,70,25]);
```

```
hcontour = uicontrol('Style','pushbutton',...
    'String','Countour','Position',[315,135,70,25]);
```

これらのステートメントは、関数 `uicontrol` を使用してプッシュ ボタンを作成しています。各ステートメントは、一連のプロパティ/値の組を使用してプッシュ ボタンを定義します。

プロパティ	説明
Style	例では、 <code>pushbutton</code> はプッシュ ボタンとしてコンポーネントを指定します。
String	各プッシュ ボタン上に表示されるラベルを指定します。ここでは、3 つのタイプのプロット <code>Surf</code> 、 <code>Mesh</code> 、 <code>Contour</code> です。
Position	4-要素ベクトル [左端からの距離、下端からの距離、幅、高さ] を使用して GUI 内の各プッシュ ボタンの位置とサイズを指定します。プッシュ ボタンに対する既定の単位は、ピクセルです。

各呼び出しほは、作成されるコンポーネントのハンドルを返します。

- 2 プッシュ ボタンの定義に続き、次のステートメントを M ファイルに追加してユーザー GUI にポップアップ メニューとラベルを追加します。

```
hpopup = uicontrol('Style','popupmenu',...
    'String',{'Peaks','Membrane','Sinc'},...
    'Position',[300,50,100,25]);
htext = uicontrol('Style','text','String','Select Data',...
    'Position',[325,90,60,15]);
```

ポップアップ メニューに対して、`String` プロパティはセル配列を使用して、3 つの項目 `Peaks`、`Membrane`、`Sinc` のポップアップ メニューを指定します。スタティック テキスト コンポーネントは、ポップアップ メニューに対するラベルとして機能します。その `String` プロパティは、GUI ユーザーに `Select Data` と示します。これらのコンポーネントに対する既定の単位は、ピクセルです。

- 3 次のステートメントを M ファイルに追加することによって、GUI に座標軸を追加します。`Units` プロパティをピクセルに設定して、他のコンポーネントと同じ単位をもつようにします。

```
ha = axes('Units','pixels','Position',[50,60,200,185]);
```

- 4 次のステートメントを用いて、座標軸を除くすべてのコンポーネントについて中心を揃えて整列させます。すべてのコンポーネントの定義に続けて、このステートメントを M ファイルに追加します。

```
align([hsurf,hmesh,hcontour,htext,hpopup], 'Center', 'None');
```

- 5 align コマンドに続けて、次のコマンドを追加して GUI を可視にします。

```
set(f, 'Visible', 'on')
```

- 6 ここで、M ファイルは以下のようになります。

```
function simple_gui2
% SIMPLE_GUI2 Select a data set from the pop-up menu, then
% click one of the plot-type push buttons. Clicking the button
% plots the selected data in the axes.

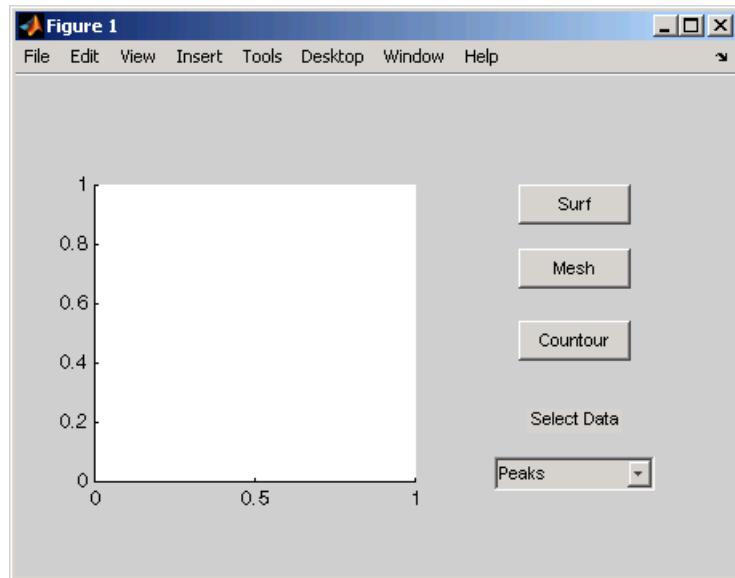
% Create and hide the GUI as it is being constructed.
f = figure('Visible', 'off', 'Position', [360, 500, 450, 285]);

% Construct the components.
hsurf = uicontrol('Style', 'pushbutton', 'String', 'Surf',...
    'Position', [315, 220, 70, 25]);
hmesh = uicontrol('Style', 'pushbutton', 'String', 'Mesh',...
    'Position', [315, 180, 70, 25]);
hcontour = uicontrol('Style', 'pushbutton',...
    'String', 'Contour',...
    'Position', [315, 135, 70, 25]);
htext = uicontrol('Style', 'text', 'String', 'Select Data',...
    'Position', [325, 90, 60, 15]);
hpopup = uicontrol('Style', 'popupmenu',...
    'String', {'Peaks', 'Membrane', 'Sinc'},...
    'Position', [300, 50, 100, 25]);
ha = axes('Units', 'Pixels', 'Position', [50, 60, 200, 185]);
align([hsurf,hmesh,hcontour,htext,hpopup], 'Center', 'None');

%Make the GUI visible.
set(f, 'Visible', 'on')

end
```

7 コマンド ラインで `simple_gui2` と入力して、M ファイルを実行します。ここで、ユーザー GUI は次のようになります。ポップアップ メニューでデータ セットを選択して、プッシュ ボタンをクリックできることに注意してください。しかし、何も起こりません。これは、ポップアップ メニューまたはボタンを使用するためのコードが M ファイルに存在しないためです。



8 コマンド ラインで `help simple_gui2` と入力します。MATLAB はこのヘルプ テキストを表示します。

```
help simple_gui2
SIMPLE_GUI2 Select a data set from the pop-up menu, then
click one of the plot-type push buttons. Clicking the button
plots the selected data in the axes.
```

次のトピック“GUI の初期化”(p.3-12)では、GUI を初期化する方法を示します。

GUI の初期化

GUI を可視化すると、ユーザーが準備を整えるために初期化を行う必要があります。このトピックは、以下の方法を示します。

- ・ GUI がコンポーネントと Figure の units を normalized に変更することによってサイズ変更されるときに、GUI が正しく動作するようにします。GUI がサイズ変更されるときコンポーネントもサイズ変更されます。規格化した単位は、Figure ウィンドウの左下隅を (0, 0) に上右隅を (1.0, 1.0) にマップします。
- ・ プロットするデータを作成するこの例には、3 つのデータセット peaks_data、membrane_data、sinc_data が必要です。各セットは、ポップアップメニューの項目の 1 つに対応します。
- ・ 座標軸に初期のプロットを作成する
- ・ GUI にウィンドウのタイトルに表示される名前を割り当てる
- ・ GUI をスクリーンの中央に移動する
- ・ GUI を可視化する

1 M ファイルの以下のコードを、

```
% Make the GUI visible.  
set(f, 'Visible', 'on')
```

以下のコードで置き換えます。

```
% Initialize the GUI.  
% Change units to normalized so components resize automatically.  
set([f, hsurf, hmesh, hcontour, htext, hpopup], 'Units', 'normalized');  
% Generate the data to plot.  
peaks_data = peaks(35);  
membrane_data = membrane;  
[x, y] = meshgrid(-8:.5:8);  
r = sqrt(x.^2+y.^2) + eps;  
sinc_data = sin(r)./r;  
% Create a plot in the axes.  
current_data = peaks_data;  
surf(current_data);  
% Assign the GUI a name to appear in the window title.  
set(f, 'Name', 'Simple GUI')
```

```
% Move the GUI to the center of the screen.
movegui(f, 'center')
% Make the GUI visible.
set(f, 'Visible', 'on');
```

2 ここで M ファイルは以下のようになります。

```
function simple_gui2
% SIMPLE_GUI2 Select a data set from the pop-up menu, then
% click one of the plot-type push buttons. Clicking the button
% plots the selected data in the axes.

% Create and hide the GUI figure as it is being constructed.
f = figure('Visible', 'off', 'Position', [360, 500, 450, 285]);

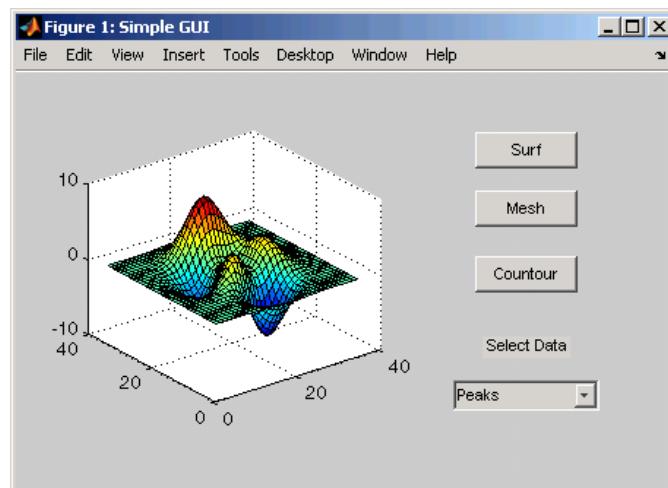
% Construct the components
hsurf = uicontrol('Style', 'pushbutton', 'String', 'Surf',...
    'Position', [315, 220, 70, 25]);
hmesh = uicontrol('Style', 'pushbutton', 'String', 'Mesh',...
    'Position', [315, 180, 70, 25]);
hcontour = uicontrol('Style', 'pushbutton',...
    'String', 'Countour',...
    'Position', [315, 135, 70, 25]);
htext = uicontrol('Style', 'text', 'String', 'Select Data',...
    'Position', [325, 90, 60, 15]);
hpopup = uicontrol('Style', 'popupmenu',...
    'String', {'Peaks', 'Membrane', 'Sinc'},...
    'Position', [300, 50, 100, 25]);
ha = axes('Units', 'Pixels', 'Position', [50, 60, 200, 185]);
align([hsurf, hmesh, hcontour, htext, hpopup], 'Center', 'None');

% Create the data to plot
peaks_data = peaks(35);
membrane_data = membrane;
[x, y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc_data = sin(r)./r;

% Initialize the GUI.
% Change units to normalized so components resize
```

```
% automatically.  
set([f, hsurf, hmesh, hcontour, htext, hpopup], ...  
    ' Units' , ' normalized' );  
%Create a plot in the axes.  
current_data = peaks_data;  
surf(current_data);  
% Assign the GUI a name to appear in the window title.  
set(f, ' Name' , ' Simple GUI' )  
% Move the GUI to the center of the screen.  
movegui(f, ' center' )  
% Make the GUI visible.  
set(f, ' Visible' , ' on' );  
  
end
```

- 3 コマンド ラインで `simple_gui2` と入力して、M ファイルを実行します。上記の初期化により、関数 `surf` を用いて、既定値の `peaks` データが表示されます。GUI は次のような外見になります。



次のトピック“GUI のプログラミング”(p.3-15)は、座標軸に異なるプロットを対話的に作成するような、プッシュ ボタンとポップアップ メニューのプログラミングの方法を示します。

GUI のプログラミング

このセクションの内容…

“ポップアップ メニューのプログラミング” (p.3-15)

“プッシュ ボタンのプログラミング” (p.3-16)

“コールバックをコンポーネントと関連させる” (p.3-16)

ポップアップ メニューのプログラミング

ポップアップ メニューにより、プロットするデータを選択することができます。GUI ユーザーが 3 つのデータ セットの 1 つを選択する場合、MATLAB は、ポップアップ メニューの Value プロパティに選択した文字列のインデックスを設定します。ポップアップ メニューの コールバックは、ポップアップ メニューの Value プロパティを読み、現在どの項目が表示されているかを判別し、それに従って current_data を設定します。

ファイルに、以下のコールバックを初期化コードの後に続け、最後の end ステートメントとの間に追加します。

```
% Pop-up menu callback. Read the pop-up menu Value property to
% determine which item is currently displayed and make it the
% current data. This callback automatically has access to
% current_data because this function is nested at a lower level.
function popup_menu_Callback(source, eventdata)
    % Determine the selected data set.
    str = get(source, 'String');
    val = get(source, 'Value');
    % Set current data to the selected data set.
    switch str{val};
        case 'Peaks' % User selects Peaks.
            current_data = peaks_data;
        case 'Membrane' % User selects Membrane.
            current_data = membrane_data;
        case 'Sinc' % User selects Sinc.
            current_data = sinc_data;
    end
end
```

次のトピック“プッシュ ボタンのプログラミング”(p.3-16)は、3 つのプッシュ ボタンに対するコールバックを記述する方法を示します。

プッシュ ボタンのプログラミング

それぞれのプッシュ ボタンは、ポップアップ メニューで現在選択されたデータを使用して、異なるタイプのプロットを作成します。プッシュ ボタンのコールバックは、`current_data` のデータをプロットします。それらは、`current_data` よりも下位レベルで入れ子になっているため `current_data` へのアクセスを自動的に行います。

ファイルに、以下のコールバックをポップアップ メニュー コールバックの後に続け、最後の `end` ステートメントとの間に追加します。

```
% Push button callbacks. Each callback plots current_data in the
% specified plot type.

function surfbutton_Callback(source, eventdata)
    % Display surf plot of the currently selected data.
    surf(current_data);
end

function meshbutton_Callback(source, eventdata)
    % Display mesh plot of the currently selected data.
    mesh(current_data);
end

function contourbutton_Callback(source, eventdata)
    % Display contour plot of the currently selected data.
    contour(current_data);
end
```

次のトピックは、各コールバックにその特定のコンポーネントを関連させる方法を示します。

コールバックをコンポーネントと関連させる

GUI ユーザーがポップアップ メニューからデータ セットを選択したり、あるいはプッシュ ボタンの 1 つをクリックすると、MATLAB は特定のイベントに関連するコールバックを実行します。しかし、MATLAB は、どのようにしてどのコールバックを実行す

るか知るのでしょうか。各コンポーネントのCallback プロパティを使用して、コンポーネントが関連するコールバックの名前を指定する必要があります。

- 1 [Surf] プッシュボタンを定義する uicontrol ステートメントに対して、プロパティ/値 の組を追加します。

```
' Callback' , {@surfbutton_Callback}
```

従って、ステートメントは次のようにになります。

```
hsurf = uicontrol('Style','pushbutton','String','Surf',...
    'Position',[315,220,70,25],...
    'Callback',{@surfbutton_Callback});
```

Callback は、プロパティの名前です。surfbutton_Callback は、[Surf] プッシュボタンに情報を提供するコールバックの名前です。

- 2 同様に、[Mesh] プッシュボタンを定義する uicontrol ステートメントに対して、プロパティ/値 の組を追加します。

```
' Callback' , {@meshbutton_Callback}
```

- 3 [Contour] プッシュボタンを定義する uicontrol ステートメントに対して、プロパティ/値 の組を追加します。

```
' Callback' , {@contourbutton_Callback}
```

- 4 ポップアップメニューを定義する uicontrol ステートメントに対して、プロパティ/値 の組を追加します。

```
' Callback' , {@popup_menu_Callback}
```

次のトピック“GUI の起動”(p.3-18)では、最終の M ファイルを示し、GUI を実行します。

GUI の起動

このセクションの内容…

“最終の M ファイル” (p.3-18)

“GUI の起動” (p.3-21)

最終の M ファイル

ここで、最終的に M ファイルは以下のようになります。

```
function simple_gui2
% SIMPLE_GUI2 Select a data set from the pop-up menu, then
% click one of the plot-type push buttons. Clicking the button
% plots the selected data in the axes.

% Create and then hide the GUI as it is being constructed.
f = figure('Visible','off','Position',[360,500,450,285]);

% Construct the components.
hsurf = uicontrol('Style','pushbutton','String','Surf',...
    'Position',[315,220,70,25],...
    'Callback',{@surfbutton_Callback});
hmesh = uicontrol('Style','pushbutton','String','Mesh',...
    'Position',[315,180,70,25],...
    'Callback',{@meshbutton_Callback});
hcontour = uicontrol('Style','pushbutton',...
    'String','Countour',...
    'Position',[315,135,70,25],...
    'Callback',{@contourbutton_Callback});
htext = uicontrol('Style','text','String','Select Data',...
    'Position',[325,90,60,15]);
hpopup = uicontrol('Style','popupmenu',...
    'String',{'Peaks','Membrane','Sinc'},...
    'Position',[300,50,100,25],...
    'Callback',{@popup_menu_Callback});
ha = axes('Units','Pixels','Position',[50,60,200,185]);
align([hsurf,hmesh,hcontour,htext,hpopup],'Center','None');
```

```
% Create the data to plot.  
peaks_data = peaks(35);  
membrane_data = membrane;  
[x, y] = meshgrid(-8:.5:8);  
r = sqrt(x.^2+y.^2) + eps;  
sinc_data = sin(r)./r;  
  
% Initialize the GUI.  
% Change units to normalized so components resize  
% automatically.  
set([f, ha, hsurf, hmesh, hcontour, htext, hpopup], ...  
    ' Units', ' normalized');  
% Create a plot in the axes.  
current_data = peaks_data;  
surf(current_data);  
% Assign the GUI a name to appear in the window title.  
set(f, ' Name', ' Simple GUI')  
% Move the GUI to the center of the screen.  
movegui(f, ' center')  
% Make the GUI visible.  
set(f, ' Visible', ' on');  
  
% Callbacks for simple_gui2. These callbacks automatically  
% have access to component handles and initialized data  
% because they are nested at a lower level.  
  
% Pop-up menu callback. Read the pop-up menu Value property  
% to determine which item is currently displayed and make it  
% the current data.  
function popup_menu_Callback(source, eventdata)  
    % Determine the selected data set.  
    str = get(source, ' String');  
    val = get(source, ' Value');  
    % Set current data to the selected data set.  
    switch str{val};  
        case ' Peaks' % User selects Peaks.  
            current_data = peaks_data;  
        case ' Membrane' % User selects Membrane.  
            current_data = membrane_data;  
        case ' Sinc' % User selects Sinc.
```

```
        current_data = sinc_data;
    end
end

% Push button callbacks. Each callback plots current_data in
% the specified plot type.

function surfbutton_Callback(source, eventdata)
% Display surf plot of the currently selected data.
    surf(current_data);
end

function meshbutton_Callback(source, eventdata)
% Display mesh plot of the currently selected data.
    mesh(current_data);
end

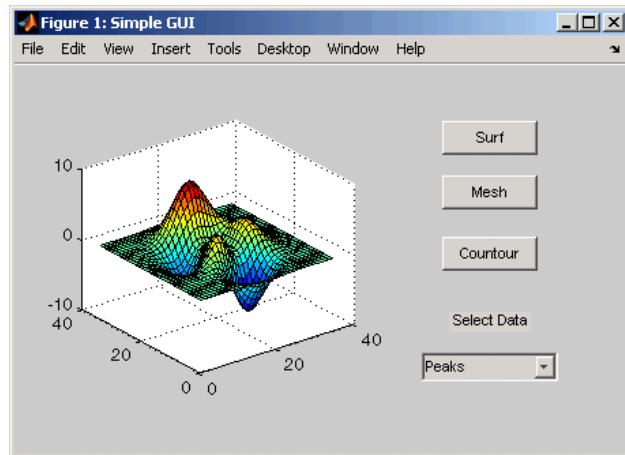
function contourbutton_Callback(source, eventdata)
% Display contour plot of the currently selected data.
    contour(current_data);
end
```

end

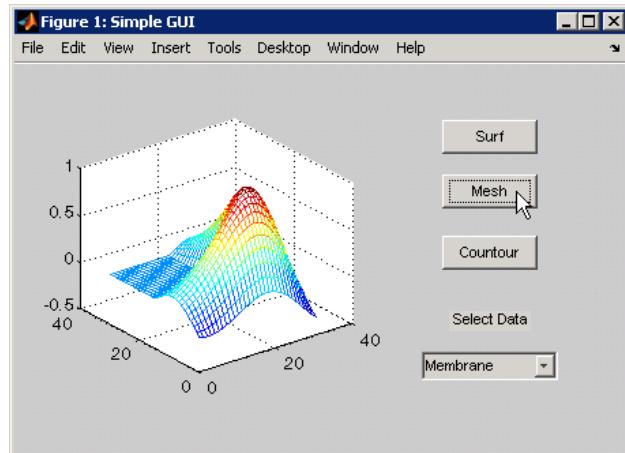
GUI の起動

1 コマンド ラインで M ファイルの名前を入力し、簡単な GUI を実行します。

```
simple_gui2
```



2 ポップアップメニューで、[Membrane] を選択してから、[Mesh] ボタンをクリックします。GUI は MATLAB ロゴのメッシュ プロットを表示します。



3 GUI を閉じる前に他の組合せのいくつかを試します。

GUIDE を使用して GUI を作成する

章 4, GUIDE とは (p. 4-1)

GUIDE を紹介します

章 5, GUIDE の設定とオプション
(p. 5-1)

利用可能な MATLAB の設定と GUI
オプションを簡単に説明します。

章 6, GUIDE GUI のレイアウト (p. 6-1)

GUIDE の起動方法、GUIDE から GUI
を表示する方法、メニューの作成方
法を説明します。クロスプラットフォー
ム互換のための GUI を設計するガ
イダンスを提供します。

章 7, GUIDE GUI の保存と実行
(p. 7-1)

GUI を保存するために使用するファ
イルについて説明します。GUI を保
存するプロセスの手順を紹介し、GUI
をアクティブにできる様々な方法を
リストします。

章 8, GUIDE GUI のプログラミング
(p. 8-1)

ユーザーが記述したコールバック
ルーチンが、どのように GUI の動作
をコントロールするかについて説明
します。コールバックを特定のコン
ポーネントと関連させる方法を示し、
コールバックの構文と引数を説明し
ます。各種コンポーネントをプログラ
ミングする簡単な例を説明します。

章 9, GUIDE でのアプリケーション
データの管理と共有 (p. 9-1)

アプリケーション定義のデータを管
理するための仕組みを説明し、GUI
のコールバック間でデータを共有す
る方法を説明します。

章 10, GUIDE GUI の例 (p. 10-1)

様々な動作をプログラミングする手
法を説明します。

GUIDE とは

- ・ “GUIDE:ご利用の前に” (p.4-2)
- ・ “GUIDE ツールのまとめ” (p.4-3)

GUIDE:ご利用の前に

このセクションの内容…

“GUI のレイアウト” (p.4-2)

“GUI のプログラミング” (p.4-2)

GUI のレイアウト

MATLAB グラフィカル ユーザー インターフェイス開発環境である GUIDE は、グラフィカル ユーザー インターフェイス (GUI) を作成する一連のツールを提供します。これらのツールを使用すると、GUI のレイアウトとプログラミングの過程を簡単に行えます。

GUIDE レイアウト エディターを使用すると、GUI コンポーネント (座標軸、パネル、ボタン、テキスト フィールド、スライダーなど) をクリックしながら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ます。GUI にメニューとコンテキスト メニューを作成することもできます。レイアウト エディターから、GUI のサイズ変更、コンポーネントの外見や特徴の修正、コンポーネントの整列、タブ順序の設定、コンポーネント オブジェクトの階層リストの表示、GUI オプションの設定ができます。

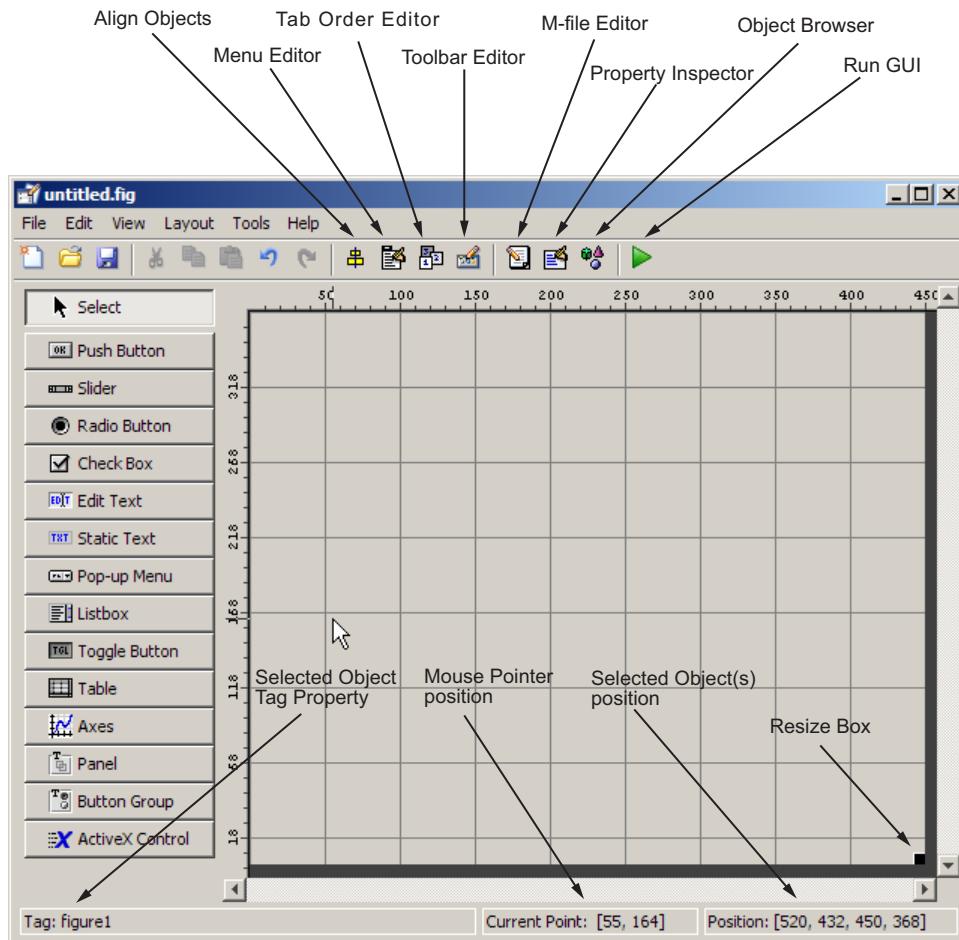
GUI のプログラミング

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します。この M ファイルは GUI を初期化するコードを提供し、また、GUI コールバック (ユーザーが GUI コンポーネントをクリックするときに実行するルーチン) に対するフレームワークを含みます。M ファイル エディターを使って、希望する関数を実行するためには、コールバックにコードを追加することができます。

メモ MATLAB では、1 度の関数呼び出しで作成できる標準的なダイアログ ボックスを選択できます。これらのダイアログ ボックスとその作成に使用する関数についての詳細は、MATLAB 関数リファレンス ドキュメンテーションの GUI Development セクションの “Predefined Dialog Boxes” を参照してください。

GUIDE ツールのまとめ

GUIDE ツールは、下図に示すレイアウトエディターから利用できます。次の図でツール名を示し、以下に簡単に説明します。以下の節では、その利用方法を示します。



ツール	ツールの用途
レイアウト エディター	レイアウト エディターの左側のコンポーネント パレットからコンポーネントを選択して、レイアウト アリアで配置します。詳細は、“GUI にコンポーネントを追加”(p.6-19)を参照してください。
Figure サイズ変更タブ	GUI を起動するとき、GUI が初期に表示されるサイズを設定します。詳細は、“GUI サイズの設定”(p.6-15)を参照してください。
メニュー エディター	メニューとコンテキストすなわちポップアップ メニューを作成します。詳細は、“メニューの作成”(p.6-99)を参照してください。
オブジェクトの整列	コンポーネントのグループを整列して配置します。グリッドとルーラによって、オプションのグリッドにスナップを使用してコンポーネントをグリッドに整列できます。詳細は、“コンポーネントの整列”(p.6-87)を参照してください。
タブ順序編集	レイアウトでタブとコンポーネントのタブとスタックの順序を設定します。詳細は、“タブの順序の設定”(p.6-96)を参照してください。
ツールバー エディター	あらかじめ定義されたカスタムのプッシュ ボタンとトグル ボタンを含むツールバーを作成します。詳細は、“ツールバーの作成”(p.6-120)を参照してください。
アイコン エディター	ツールバー内のツールのアイコンを作成したり修正します。詳細は、“ツールバーの作成”(p.6-120)を参照してください。
プロパティ インスペクター	レイアウトのコンポーネントのプロパティを設定します。すべての設定可能なプロパティのリストを提供し、それらの現在の値を表示します。
オブジェクト ブラウザー	Figure のオブジェクトの階層的なリストを表示します。詳細は、“オブジェクト階層の表示”(p.6-134)を参照してください。
実行	現在の GUI を保存、実行します。詳細は、章 7, “GUIDE GUI の保存と実行”を参照してください。
M ファイル エディター	MATLAB エディターに、GUI に関する M ファイルを表示します。詳細は、“GUI ファイルの概要”(p.8-7)を参照してください。
位置読み出し	マウス カーソルの位置と選択されたオブジェクトの位置を連続して表示します。

作成時にすべての GUI に適用する設定と GUI に固有のオプションも設定できます。詳細は、章 5, “GUIDE の設定とオプション”を参照してください。

GUIDE の設定とオプション

- ・ “GUIDE の設定” (p.5-2)
- ・ “GUI オプション” (p.5-9)

GUIDE の設定

このセクションの内容…

“**プリファレンス設定**” (p.5-2)

“**設定の確認**” (p.5-2)

“**下位互換性の設定**” (p.5-4)

“**その他の設定**” (p.5-6)

プリファレンス設定

[ファイル] メニューから[設定] を選択することにより、GUIDE に設定ができます。これらの設定は、GUIDE と作成するすべての GUI に適用されます。

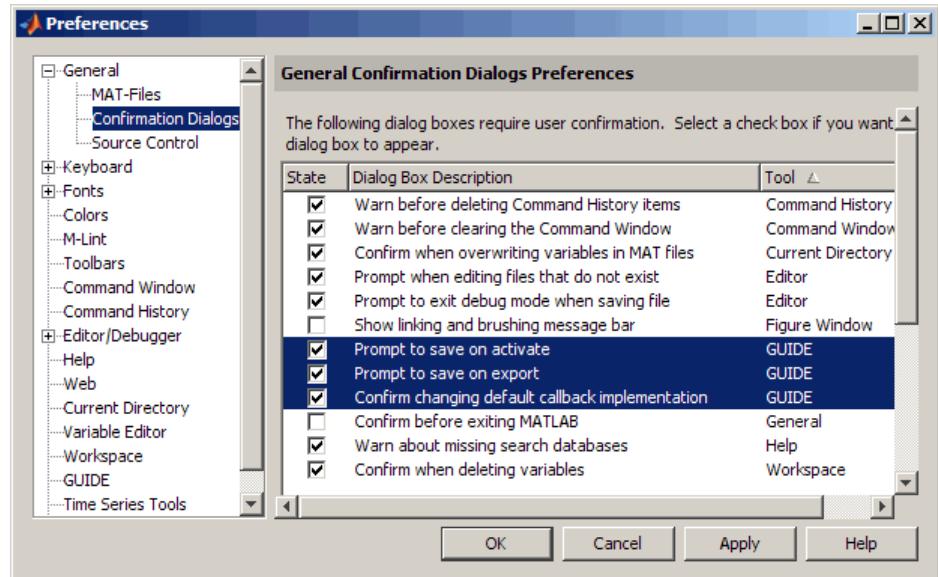
設定は、[設定] ダイアログ ボックス内で異なる場所にあります。

設定の確認

GUIDE では、2 つの設定の確認を行うことができます。以下のことを行う場合に、確認ダイアログ ボックスを表示するかどうかを選択できます。

- GUIDE から GUI をアクティブにする
- GUIDE から GUI をエクスポートする
- GUIDE により生成されるコールバックのシグネチャを変更する

MATLAB の [設定] ダイアログ ボックスで、[一般] > [確認ダイアログ] をクリックして、GUIDE の確認設定にアクセスします。[ツール] の列で GUIDE という語を探します。



実行時の保存

レイアウト エディターで [実行] ボタン をクリックすることにより、GUI をアクティブにする場合、ダイアログ ボックスは、保存するかどうかをたずね、作業を続けるかどうかを選択できます。



エクスポート時の保存

[ファイル] メニューから[エクスポート]を選択すると、ダイアログ ボックスは、保存するかどうかをたずね、作業を続けるかどうかを選択できます。

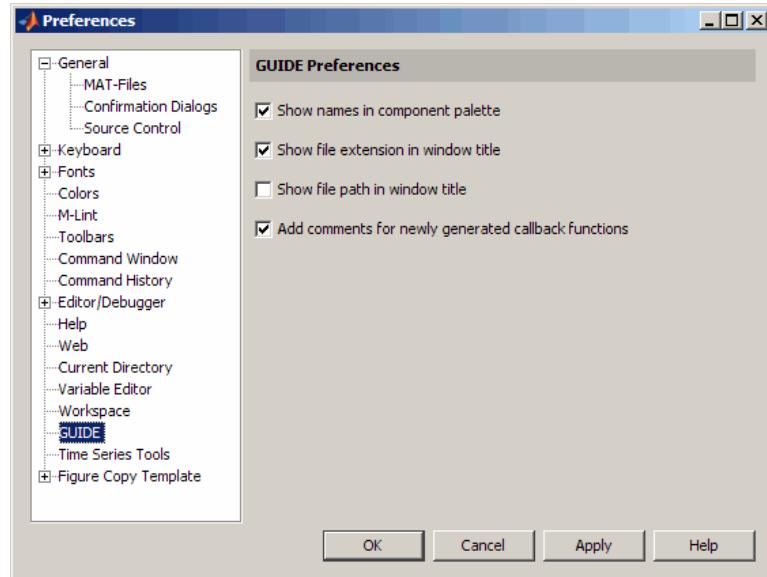


下位互換性の設定

MATLAB バージョン 5 以降の互換性

MATLAB 7.0 または それ以降のバージョンの MATLAB で作成されたり、修正された GUI FIG-ファイルは、バージョン 6.5 およびそれ以前のバージョンのものと自動的には互換性がありません。GUIDE は、MAT-ファイルの一種で、GUI のレイアウトの情報をもつ FIG-ファイルを自動的に作成します。

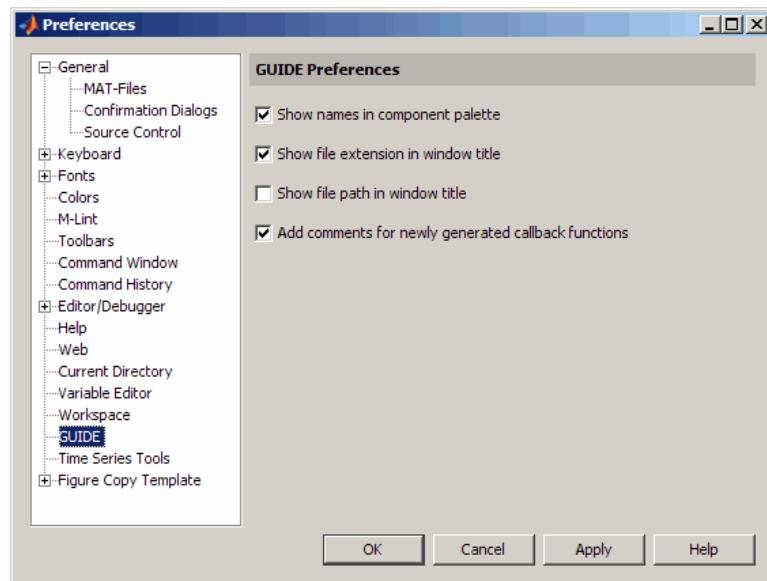
FIG-ファイルを下位互換にするには、MATLAB の [設定] ダイアログ ボックスで、下図に示すように [ファイル] > [設定] > [一般] > [MAT-ファイル] > [MATLAB Version 5 以降 (-v6 を保存)] を選択します。



メモ ここで述べるオプション [-v6] は廃止され、今後のバージョンではなくなる予定です。

その他の設定

GUIDE は、レイアウトエディターや M ファイルのコメントに対する他の設定をいくつか提供します。[設定] ダイアログ ボックスで、GUIDE をクリックして、これらの設定にアクセスできます。

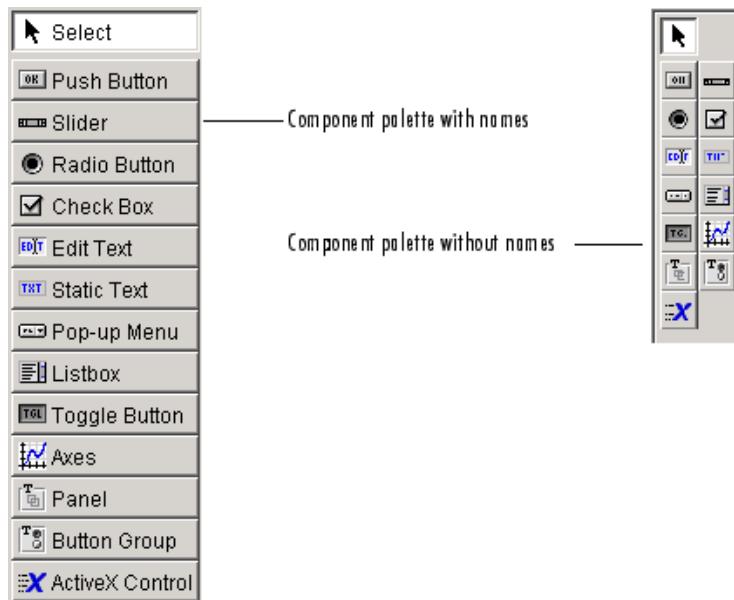


以下のトピックスは、このダイアログでの設定を説明します。

- ・ “コンポーネント パレットに名前を表示” (p.5-6)
- ・ “ウィンドウ タイトルにファイル拡張子を表示” (p.5-7)
- ・ “ウィンドウ タイトルにファイル パスを表示” (p.5-7)
- ・ “新規に作成されたコールバック関数にコメントを追加” (p.5-7)

コンポーネント パレットに名前を表示

コンポーネント パレットにアイコンと名前を表示します。チェックされないと、2 つの列にツールのヒントとともにアイコンが表示されます。



ウィンドウ タイトルにファイル拡張子を表示

レイアウトエディター ウィンドウ タイトルにファイル拡張子 .fig の付いた GUI FIG-ファイルのファイル名を表示します。チェックしない場合、ファイル名のみ表示されます。

ウィンドウ タイトルにファイルパスを表示

レイアウトエディター ウィンドウ タイトルにフル ファイルパスを表示します。チェックしない場合、ファイルパスは表示されません。

新規に作成されたコールバック関数にコメントを追加

コールバックは、GUI のユーザーによるボタンクリック、またはスライダーの操作などの動作に応答して実行する、コードまたは関数のブロックです。既定では、GUIDE は、関数宣言としてコールバックをコーディングするためのテンプレートを用意します。この設定がチェックされると、GUIDE は、M ファイルに追加するすべてのコールバック関数のはじめにコメント行を置きます。ほとんどのコメントは、以下のようになります。

```
% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB  
% handles structure with handles and user data (see GUIDATA)
```

関連するコンポーネントがオリジナルの GUIDE のテンプレートの一部であるコールバックは、自動的に追加されます。他の一般に使用されるコールバックは、コンポーネントを追加する際に、自動的に追加されます。[表示] メニューまたは、コンポーネントのコンテキストメニューの[コールバックの表示]から、コールバックを選択することにより、コールバックを明示的に追加することもできます。

この選択肢がチェックされないと、GUIDE は、オリジナルの GUIDE テンプレートをサポートするために自動的に含まれるコールバックに対してのみ、コメントを加えます。M ファイルに追加されるその他のコールバックに対しては、コメントは含まれません。

コールバックや上記のコメントで記述された議論についての詳細は、“コールバック構文と引数”(p.8-15)を参照してください。

GUI オプション

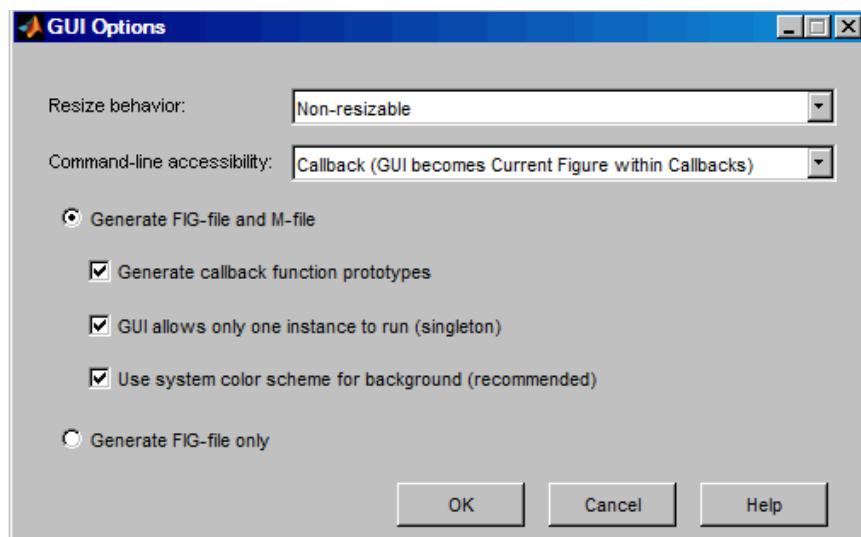
このセクションの内容…

- “[GUI オプション] ダイアログ ボックス” (p.5-9)
- “サイズ変更動作” (p.5-10)
- “コマンド ラインからのアクセス” (p.5-10)
- “FIG-ファイルと M ファイルの作成” (p.5-11)
- “FIG-ファイルのみ作成” (p.5-14)

[GUI オプション] ダイアログ ボックス

[GUI オプション] ダイアログ ボックスを使用すると、作成している GUI に固有の様々な動作を設定できます。これらのオプションは、この GUI を次回保存するときに有効です。

レイアウト エディターの[ツール] メニューから [GUI オプション] を選択してダイアログにアクセスします。



以下の節では、このダイアログ ボックスのオプションについて説明します。

サイズ変更動作

ユーザー GUI を含む Figure ウィンドウをサイズ変更可能にしたり、さらに、MATLAB によるサイズ変更の方法を設定できます。GUIDE は、3 つのオプションを提供します。

- ・ [サイズ変更不可]—ユーザーはウィンドウのサイズを変更できません（既定値）。
- ・ [比例 (プロポーショナル)]—MATLAB は、新規の Figure ウィンドウ サイズに比例して、GUI のコンポーネントを自動的に再スケールします。
- ・ [その他 (ResizeFcnを使用)]—Figure ウィンドウのサイズを変更する際の GUI の動作をプログラムします。

最初の 2 つのアプローチは、単純に、プロパティを適当に設定し、他のアクションを必要としません。[その他 (ResizeFcnを使用)] のサイズ変更では、新規の Figure サイズに基づくコンポーネントのサイズと位置を再計算するコールバック ルーチンをユーザーが記述する必要があります。ResizeFcn の利用とその例は、GUIDE の例 “パネル” (p.8-39) と “アドレス ブック リーダー” (p.10-81) を参照してください。さらに、MATLAB「グラフィックス」ドキュメンテーションの例 “Using Panel Containers in Figures — Uipanels”を参照してください。この例では、GUIDE は使用しません。

コマンド ラインからのアクセス

GUIDE の[コマンド ラインからのアクセス] オプションを使用してコマンド ラインまたは M ファイルから GUI の Figure のハンドルへのアクセスを制限することができます。

Figure ハンドルを明示的に指定しないと、plot など多くのコマンドは、現在の Figure すなわちルートの CurrentFigure プロパティで指定されたり、gcf コマンドで返される Figure を変更します。現在の Figure は、通常最も最近、作成、または描画されるかクリックされた Figure です。プログラミングにより、以下の 4 つの方法で現在の Figure として、Figure を指定できます。

- 1 `set(0, 'CurrentFigure', h)` —Figure h を現在の Figure にしますが、その Figure の可視性や、他の Figure との前後関係を変更しません。
- 2 `figure(h)` —Figure h を現在の Figure、可視にして、他の Figure の手前に表示します。
- 3 `axes(h)` —既存の Axes h を現在の座標軸にし、これを含む Figure を他の Figure よりも手前に表示します。

`4 plot(h,...)`、あるいは `Axes` を最初の引数とするプロット関数を用いても、既存の `Axes h` を現在の座標軸とし、これを含む `Figure` を他の `Figure` よりも手前に表示します。

関数 `gcf` は、現在の `Figure` のハンドルを返します。

```
h = gcf
```

GUIDE で作成される GUI に対して、[コマンド ラインからのアクセス] オプションを設定し、`plot` などのコマンドをコマンド ラインまたは M ファイルから実行することで GUI の外見や内容を誤って変更してしまうことがないようにします。次の表は、[コマンド ラインからのアクセス] に対する 4 つのオプションについて簡単に説明します。

オプション	説明
コールバック (GUI はコールバック中に現在の Figure になります)	GUI は、GUI コールバック内からのみアクセスできます。GUI はコマンド ラインや M スクリプトからアクセスできません。これは既定値です。
オフ (GUI は現在の Figure なりません)	GUI はハンドルがない場合、コールバック、コマンド ライン、M スクリプトからアクセスできません。
オン (GUI はコマンド ラインから現在の Figure として操作できます)	GUI は、コールバック、コマンド ライン、M スクリプトからアクセスできます。
その他 (プロパティ インスペクターからの設定を利用します)	プロパティ インスペクターから <code>HandleVisibility</code> と <code>IntegerHandle</code> プロパティを設定することによってアクセスをコントロールできます。

FIG-ファイルと M ファイルの作成

GUIDE により FIG-ファイルと GUI M ファイルの両方を生成したい（既定値）場合、[GUI Options] ダイアログの [FIG-ファイルと M ファイルの作成] を選択してください。一度、このオプションを選択すると、M ファイルを設定するためにフレーム内の以下の項目のいずれかを選択することができます。

- ・ “コールバック関数のプロトタイプを作成” (p.5-12)
- ・ “GUI は 1 つのインスタンスのみの実行を許可（シングルトン）” (p.5-12)

- ・ “背景にシステム配色を使用” (p.5-13)

これらのファイルについての詳細は、“GUI ファイルの概要” (p.8-7)を参照してください。

コールバック関数のプロトタイプを作成

[GUI Options] ダイアログで、[コールバック関数のプロトタイプを作成]を選択する場合、GUIDE は、GUI に追加するほとんどのコンポーネントに対する GUI M ファイルに、最も一般に使用されるコールバックに対するテンプレートを追加します。その後、これらのコールバックに対するコードを記述する必要があります。

GUIDE はレイアウト エディターの右クリック コンテキスト メニューからコールバックルーチンを編集したり、メニュー エディターを用いて GUI にメニューを追加する場合には、コールバックの追加も行います。

コールバックについての一般情報は、“コールバック構文と引数” (p.8-15)を参照してください。

メモ このオプションは最初に [FIG-ファイルと M ファイルの作成] オプションを選択した場合に限り、利用できます。

GUI は 1 つのインスタンスのみの実行を許可 (シングルトン)

このオプションにより、ユーザーは、GUI の Figure に対する 2 つの動作間で、選択可能になります。

- ・ MATLAB が、同時に、GUI のインスタンスを 1 つだけ表示するようにします。
- ・ MATLAB が、同時に、GUI の複数のインスタンスを表示できるようにします。

1 つのインスタンスのみが可能である場合、MATLAB は、GUI を起動するコマンドを実行すると、既存の GUI の Figure を再使用します。GUI がすでに存在する場合、MATLAB は、新規の Figure を生成せずに、これを手前にもってきます。

ユーザーがこのオプションをチェックしない場合、MATLAB は、GUI を起動するコマンドを実行するときは必ず、新規の GUI の Figure を作成します。

一度に GUI の 1 つのインスタンスのみの実行を可能にしていても、コマンド ラインからこのインスタンスを呼び出す度に初期化可能です。たとえば、OpeningFcn のコードは、初期化を防ぐ手順をとらない限り、GUIDE GUI が実行する度に実行します。handles 構造体にフラグを追加することは、そのような動作をコントロールする 1 つの方法です。OpeningFcn でこれを行うことができます。フラグがまだ存在しない場合は初期化コードを実行でき、フラグが存在する場合は初期化コードをスキップできます。

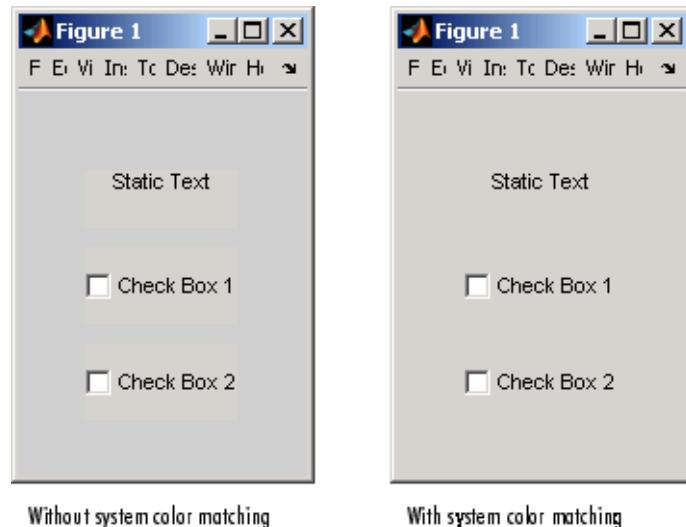
メモ このオプションは最初に [FIG-ファイルと M ファイルの作成] オプションを選択した場合に限り、利用できます。

背景にシステム配色を使用

GUI コンポーネントに対して使用される既定の色は、システムに依存して変わります。このオプションにより、Figure の背景色を既定のコンポーネント背景色と同じにできます。

[背景にシステム配色を使用] (既定値) を選択すると、GUIDE は Figure の背景色を変更して、GUI コンポーネントの色と同じにします。

次の図は、システム カラー マッチングがある場合とない場合の結果を示します。



メモ このオプションは最初に [FIG-ファイルと M ファイルの作成] オプションを選択した場合に限り、利用できます。

FIG-ファイルのみ作成

[FIG-ファイルのみ作成] オプションによって、編集が制限された Figure や GUI を開くことができます。これらは Figure ですが、GUI である必要はありません。GUI は GUIDE を使用して作成される必要はありません。このモードでは、編集機能が制限されていますが、バージョン 5.3 以前の MATLAB で作成した GUI には有効です。詳細は、関数 `guide` を参照してください。

GUIDE は、以下のいずれかを行う場合、既定値として [FIG-ファイルのみ作成] を選択します。

- コマンド ラインから GUIDE を起動し、引数として 1 つまたは複数の Figure ハンドルを与えます。

```
guide(fh)
```

この場合、GUIDE は同じ位置に対応する M ファイルがある場合でも、[FIG-ファイルのみ作成] を選択します。

- コマンド ラインから GUIDE を起動して、同じ位置に同じ名前をもつ M ファイルが存在しないような FIG-ファイルの名前を与えます。

```
guide('myfig.fig')
```

- 同じ位置に同じ名前をもつ M ファイルが存在しないような FIG-ファイルを開くには、GUIDE の [既存の GUI を開く] タブを使用します。

[FIG-ファイルのみ作成] を選択して Figure や GUI を保存すると、GUIDE は FIG-ファイルのみを保存します。必要に応じて、対応する M ファイルを更新する必要があります。

GUIDE で GUI M ファイルを管理したい場合は、GUI を保存する前に [FIG-ファイルと M ファイルの作成] の選択を変更してください。同じ位置に対応する M ファイルがない場合は、GUIDE が生成します。同じ名前をもつ M ファイルがオリジナルの Figure や GUI として同じ位置に存在する場合には、GUIDE が上書きします。既存の M ファイルの上書きを防ぐには、[ファイル] メニューから [別名で保存] を使用し

て GUI を保存し、他のファイル名を選択します。必要に応じて、新しい M ファイルを更新する必要があります。

M ファイルなしの GUI コールバック

GUI FIG-ファイルに関する M ファイルがない場合でも、GUI コンポーネントにコールバックを与えて、使用されるときに動作を実行するようにできます。プロパティ インスペクターで、文字列、組み込み関数、M ファイル名の形でコールバックを入力できます。GUI が実行するとき、可能であれば、GUI はそれらを実行します。コールバックが M ファイルの名前であるとき、引数を含むことができます。たとえば、プッシュ ボタンの Callback プロパティを `sqrt(2)` に設定すると、式の結果がコマンド ウィンドウに表示されます。

```
ans =  
1.4142
```

コールバックが実行する M ファイルは、現在のフォルダー、または MATLAB パス上になければなりません。コールバックの動作に関する詳細は、“コールバック:概要”(p.8-2)を参照してください。

GUIDE GUI のレイアウト

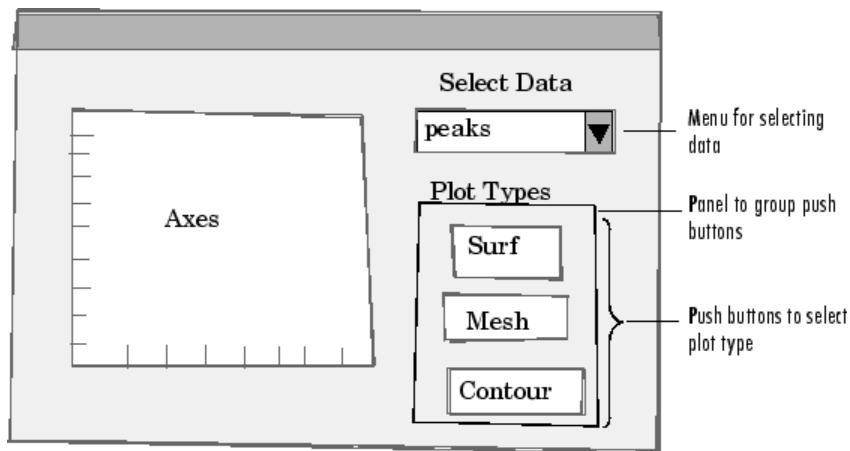
- “GUI の設計” (p.6-2)
- “GUIDE の起動” (p.6-4)
- “GUI テンプレートの選択” (p.6-5)
- “GUI サイズの設定” (p.6-15)
- “GUI にコンポーネントを追加” (p.6-19)
- “コンポーネントの整列” (p.6-87)
- “タブの順序の設定” (p.6-96)
- “メニューの作成” (p.6-99)
- “ツールバーの作成” (p.6-120)
- “オブジェクト階層の表示” (p.6-134)
- “クロスプラットフォーム互換性のための設計” (p.6-135)

GUI の設計

実際の GUI を作成する前に、GUI にさせたいこと、および、どのような動作をさせたいかを決めるることは重要です。GUI を紙に描き、ユーザーが見たり、行うアクションについて予測することは役立ちます。

メモ MATLAB では、1 度の関数呼び出しで作成できる標準的なダイアログ ボックスを選択できます。これらのダイアログ ボックスと作成に使用する関数についての詳細は、MATLAB 関数リファレンス ドキュメンテーションの“Predefined Dialog Boxes”を参照してください。

この例で使用する GUI には、ポップアップ メニューから選択するデータを、表面、メッシュ、あるいは等高線図のいずれかの形式で表示できる座標軸コンポーネントが含まれています。次の図は、設計のための出発点として用いるスケッチを示します。



このパネルは 3 つのプッシュ ボタンを含み、ユーザーの希望でプロットのタイプを選択できます。ポップアップ メニューは、MATLAB 関数に相当する 3 つの文字列、`peaks`、`membrane`、`sinc` を含んでいます。プロットするデータをこのメニューから選択できます。

以下のような多くの Web サイトや市販の出版物で、GUI の設計のためのガイドラインが提供されています。

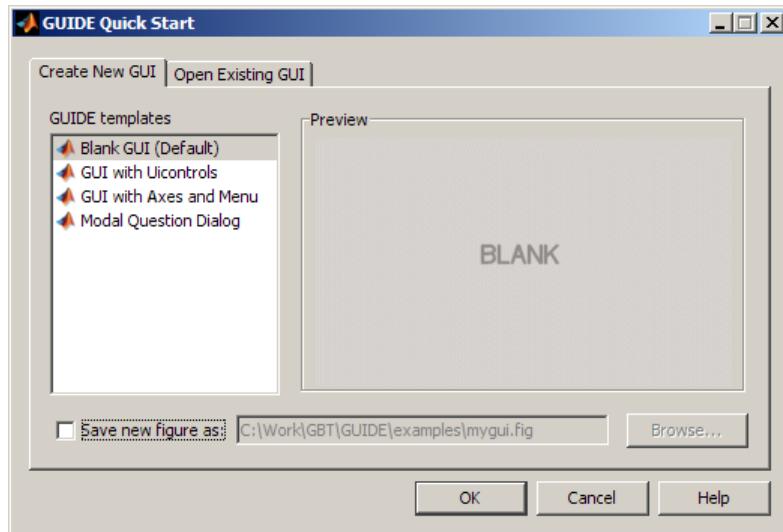
- ・ AskTog – 良い設計についてのエッセイとユーザー インターフェイス設計のための原則のリスト著者 Bruce Tognazzini は一流のユーザー インターフェイス デザイナーです。<http://www.asktog.com/basics/firstPrinciples.html>
- ・ Galitz, Wilbert, O., *Essential Guide to User Interface Design*. Wiley, New York, NY, 2002.
- ・ GUI Design Handbook – GUI コントロールの利用のための詳細なガイド <http://www.fast-consulting.com/desktop.htm>.
- ・ Johnson, J., *GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers*. Morgan Kaufmann, San Francisco, CA, 2000.
- ・ Usability Glossary – GUI 設計、有用性、関連トピックスについての広範囲の用語集。<http://www.usabilityfirst.com/glossary/main.cgi>.
- ・ UsabilityNet – 設計原理、ユーザー中心の設計、さらに他の有用性と設計に関するトピックスを説明します。http://www.usabilitynet.org/management/b_design.htm.

GUIDE の起動

GUIDE を起動する方法は多数あります。次の方法で GUIDE を起動できます。

- ・ コマンド ラインから `guide` とタイプする
- ・ [スタート] メニューから [MATLAB] > [GUIDE (GUI ビルダー)] を選択する
- ・ [MATLAB ファイル] メニューから [新規作成] > [GUI] を選択する
- ・ MATLAB ツールバーから [GUIDE] ボタンをクリックする

これにより、次の図の [GUIDE のクイック スタート] ダイアログ ボックスが表示されます。



[GUIDE のクイック スタート] ダイアログ ボックスは、2つのタブ ダイアログを含みます。

- ・ [新しい GUI を作成] – 新しい GUI に対するテンプレートを選択することによって、新しい GUI の作成を開始できます。GUI を保存するための名前も指定できます。
テンプレートについての詳細は、“GUI テンプレートの選択” (p.6-5)を参照してください。
- ・ 既存の GUI を開く – GUIDE で既存の GUI を開くことができます。現在のフォルダーカから GUI を選択するか、あるいは他のディレクトリをブラウズできます。

GUI テンプレートの選択

このセクションの内容…

“テンプレートへのアクセス” (p.6-5)

“テンプレートの説明” (p.6-6)

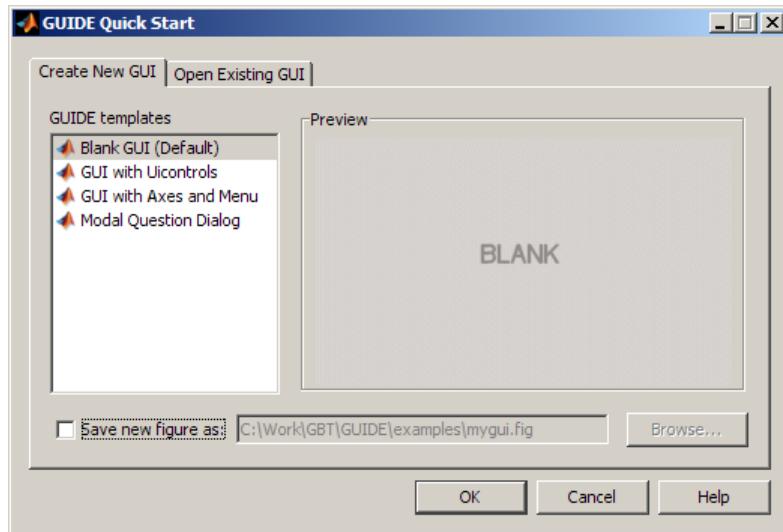
テンプレートへのアクセス

GUIDE は、ユーザーの GUI を作成するために修正できるいくつかのテンプレートを提供します。テンプレートは、完全な機能をもつ GUI で、そのコールバックはすでにプログラムされています。

2 通りの方法で、テンプレートにアクセスできます。

- GUIDE の起動詳細は、“GUIDE の起動” (p.6-4)を参照してください。
- GUIDE がすでに開いている場合、レイアウトエディターで [ファイル] メニューから [新規作成] 選択します。

いずれの場合も、GUIDE は、次の図に示すように、[新しい GUI を作成] タブが選択された状態で [GUIDE のクイックスタート] ダイアログ ボックスを表示します。このタブには、利用できるテンプレートのリストが含まれます。



テンプレートを使用するために、以下を行います。

- 1 左のペインでテンプレートを選択します。右のペインにプレビューが表示されます。
- 2 オプションで、[新規 Figure を別名で保存]を選択して、右側のフィールドに名前を入力して、GUI に名前を付けます。GUIDE は、GUI をレイアウトエディターに開く前に GUI を保存します。この時点で GUI に名前を付けないと、GUI を保存し、最初に実行するときに名前を付けるように GUIDE がプロンプトを表示します。
- 3 [OK] をクリックすると、レイアウトエディターに GUI テンプレートが開きます。

テンプレートの説明

GUIDE は、完全な機能をもつ 4 つのテンプレートを提供します。これらについて、次の節で説明します。

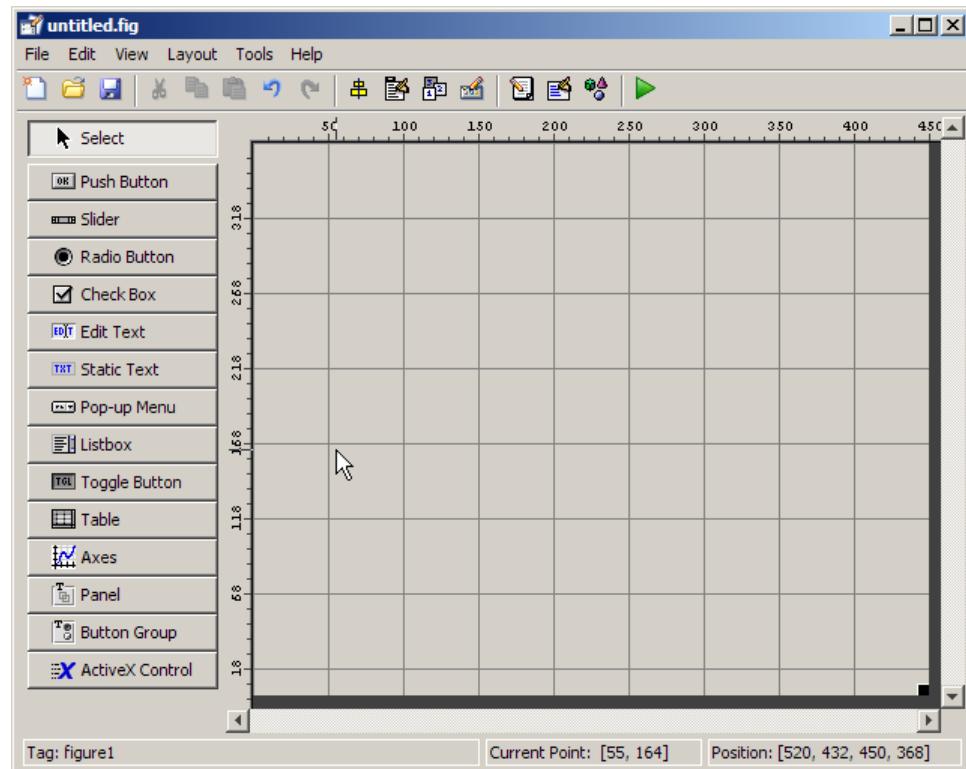
- ・ “空白 GUI” (p.6-8)
- ・ “Uicontrol をもつ GUI” (p.6-9)
- ・ “座標軸とメニュー付きの GUI” (p.6-10)
- ・ “モーダル クエスチョン ダイアログ” (p.6-13)

“初期状態では” GUI テンプレートのいずれもメニュー バーまたはツールバーを含みません。いずれも、MATLAB デスクトップにドックできません。ただし、これらのコントロールを与えてカスタマイズするために、これらの GUIDE の既定値をオーバーライドできます。詳細は、“メニューの作成” (p.6-99) と “ツールバーの作成” (p.6-120) の節を参照してください。

メモ テンプレート GUI の動作を参考するために、M ファイルコードを表示し、それらのコールバックを見ることができます。ユーザー自身の目的のためにコールバックを修正することもできます。これらのテンプレートのいずれかに対して M ファイルを表示するには、レイアウト エディターでテンプレートを開き、ツールバーの [M ファイルエディター] ボタン をクリックします。コールバックの利用についての詳細は、章 8, “GUIDE GUI のプログラミング”を参照してください。

空白 GUI

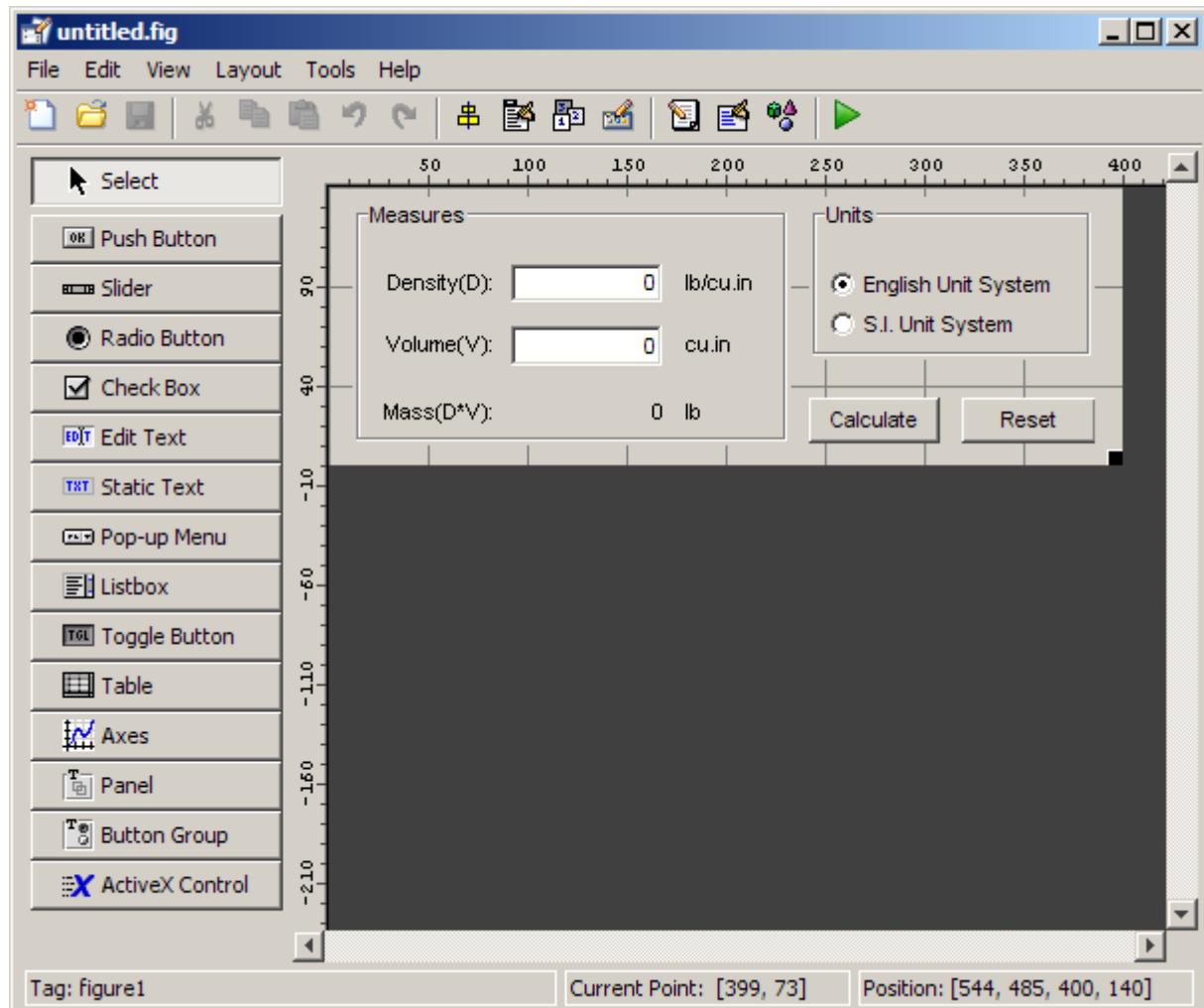
次の図は、空白 GUI のテンプレートが表示されたものです。



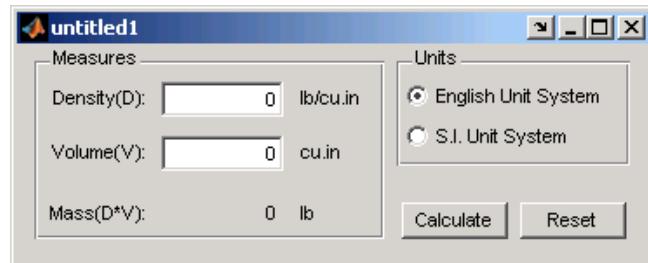
GUI を構築する出発点として他のテンプレートが適当でない場合、あるいは空の GUI から始めたい場合には、空白 GUI を選択してください。

Uicontrol をもつ GUI

次の図は、レイアウトエディターに表示された、ユーザーインターフェイスコントロール(uiicontrol)付きの GUIに対するテンプレートを示します。ユーザーインターフェイスコントロールは、プッシュボタン、スライダー、ラジオボタン、チェックボックス、エディットテキスト、スタティックテキスト、リストボックス、トグルボタンなどを含みます。



[実行] ボタン▶をクリックして GUI を実行すると、次の図のような GUI が現れます。

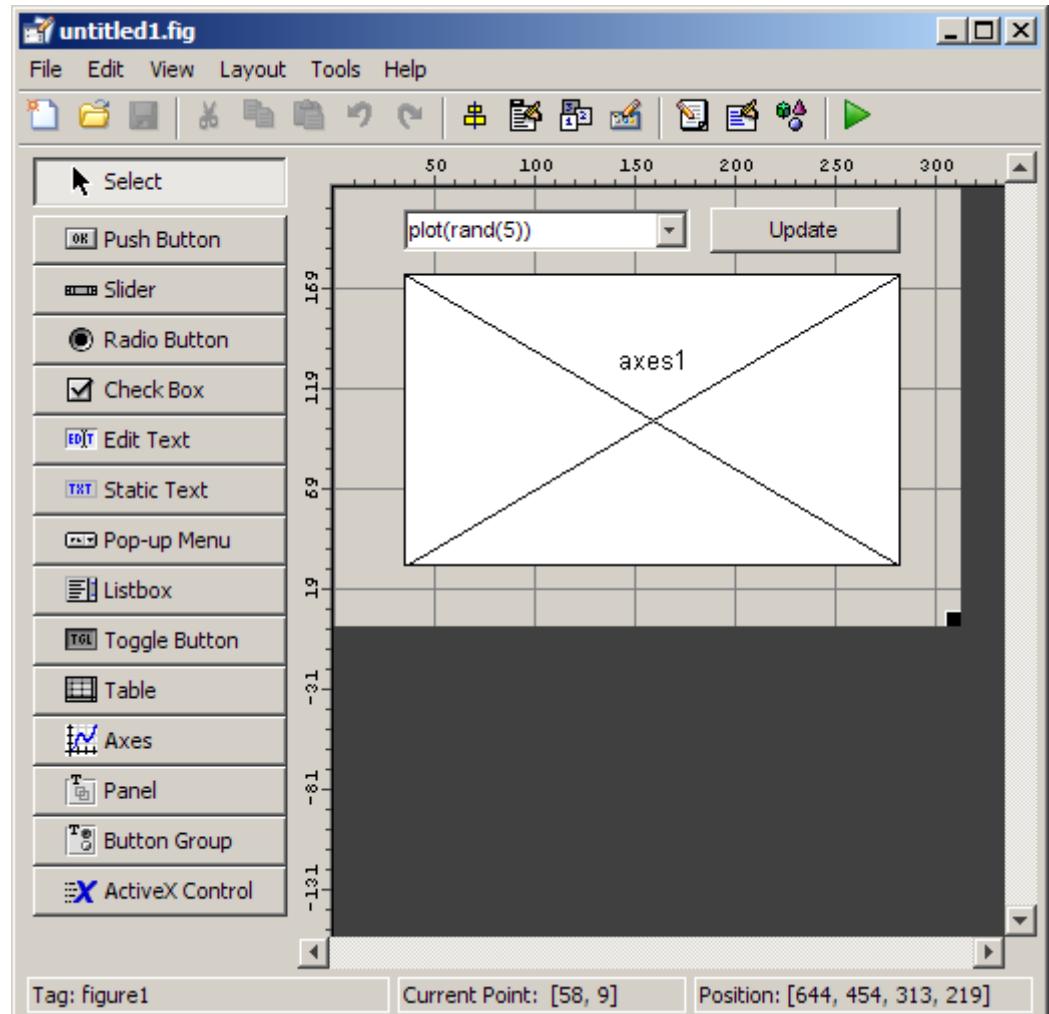


ユーザーがオブジェクトの密度と体積の値を入力し、[Calculate] ボタンをクリックすると、GUI はオブジェクトの質量を計算し、結果を $\text{Mass}(D \cdot V)$ の隣に表示します。

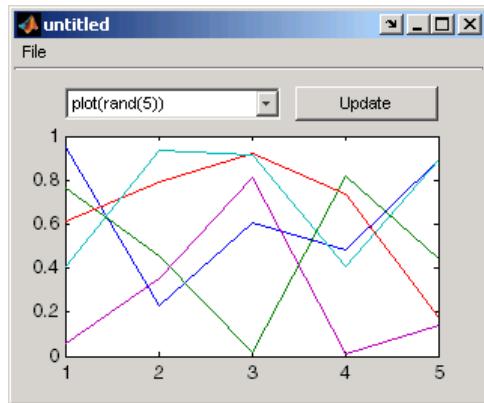
これらのユーザー インターフェイス コントロールに対する M ファイルのコードを表示するには、レイアウト エディターでテンプレートを開き、ツールバーの [M ファイル エディター] ボタン をクリックします。

座標軸とメニュー付きの GUI

次の図は、座標軸とメニューをもつ GUI のテンプレートです。



ツールバー上の [実行] ボタン▶をクリックして GUI を実行すると、GUI は、次の図のように、MATLAB コマンド `rand(5)` で生成される 5 つの線のプロットを表示します。次の図は、例を示します。



ポップアップメニューで、他のプロットを選択することができます。[更新] ボタンをクリックすると、座標軸上に現在選択されているプロットを表示します。

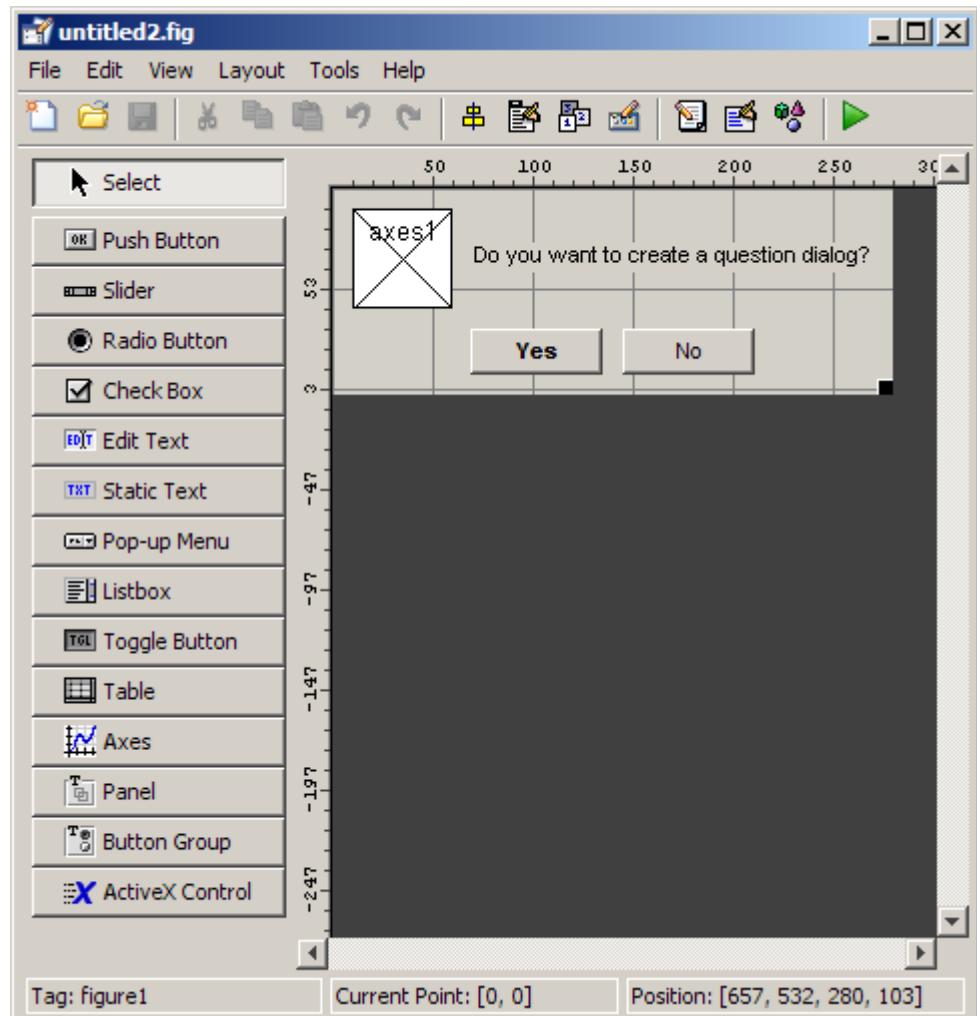
この GUI は、また、3 つの項目をもつ [ファイル] メニューをもちます。

- ・ [開く] を選択すると、コンピューター上のファイルを開くためのダイアログ ボックスを表示します。
- ・ [印刷] は [印刷] ダイアログ ボックスを開きます。[印刷] ダイアログ ボックスで [OK] をクリックすると、プロットを印刷します。
- ・ [閉じる] を選択すると、GUI が閉じます。

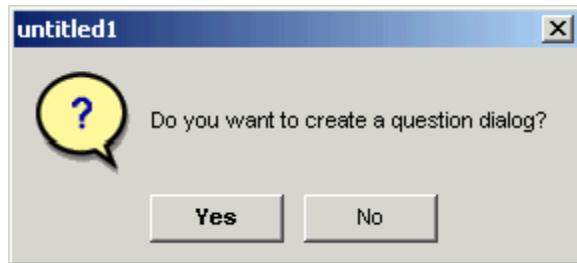
これらのメニュー選択に対する M ファイルのコードを表示するには、レイアウト エディターでテンプレートを開き、ツールバーの [M ファイル エディター] ボタン をクリックします。

モーダル クエスチョン ダイアログ

次の図は、レイアウトエディターに表示されるモーダル クエスチョン ダイアログ テンプレートです。



GUI を実行すると、次のように、ダイアログ ボックスが表示されます。



クリックされるボタンに応じて、GUI はテキスト文字列 Yes または No を返します。

ユーザー GUI が文字列を返すか、あるいはモーダルとしたい場合、このテンプレートを選択します。

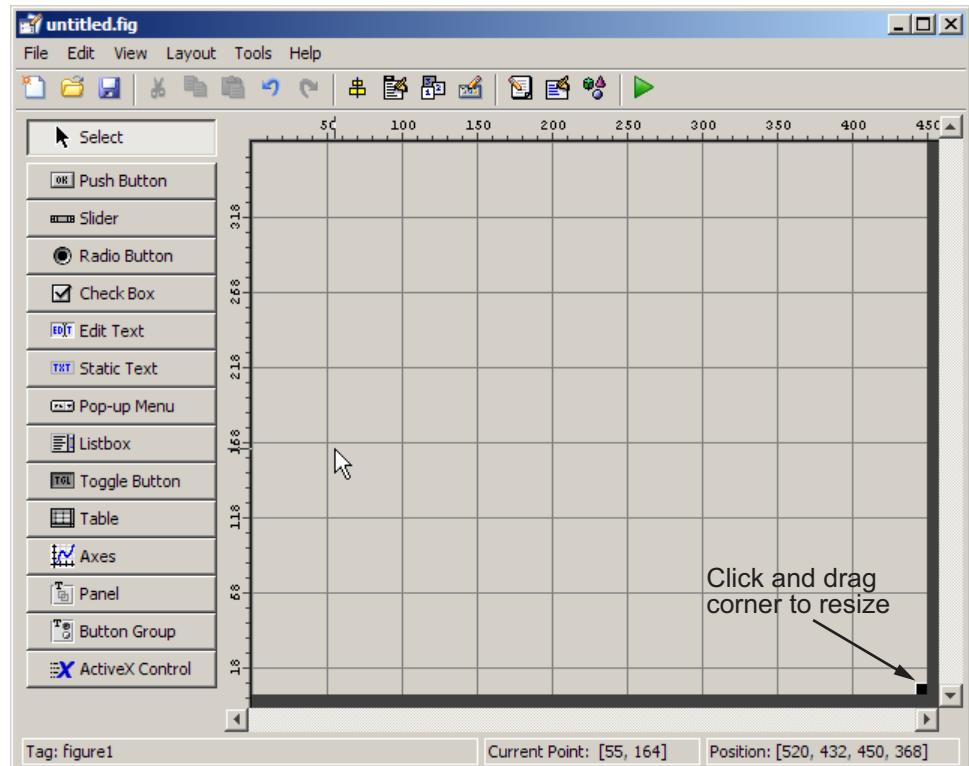
モーダル GUI はブロックされています。このことは、GUI が実行を返すまで、現在の M ファイルが実行を停止することを意味します。つまり、いずれかのボタンがクリックされるまで、ユーザーは他の MATLAB ウィンドウとやりとりできません。

メモ モーダルのダイアログ ボックス (WindowStyle を 'modal' に設定した Figure) は、メニューまたはツールバーを表示できません。

これらの機能に対する M ファイルのコードを表示するには、レイアウト エディターでテンプレートを開き、ツールバーの [M ファイル エディター] ボタン をクリックします。他の GUI についてこのテンプレートを使用する例は、“操作確認のためのモーダル ダイアログの使用”(p.10-98)を参照してください。詳細は、Figure の「WindowStyle」プロパティを参照してください。

GUI サイズの設定

レイアウトエディター内のグリッドエリアをサイズ変更することで GUI のサイズを指定します。右下隅をクリックして、GUI が適切なサイズになるまでドラッグします。必要な場合は、ウィンドウをさらに大きくしてください。

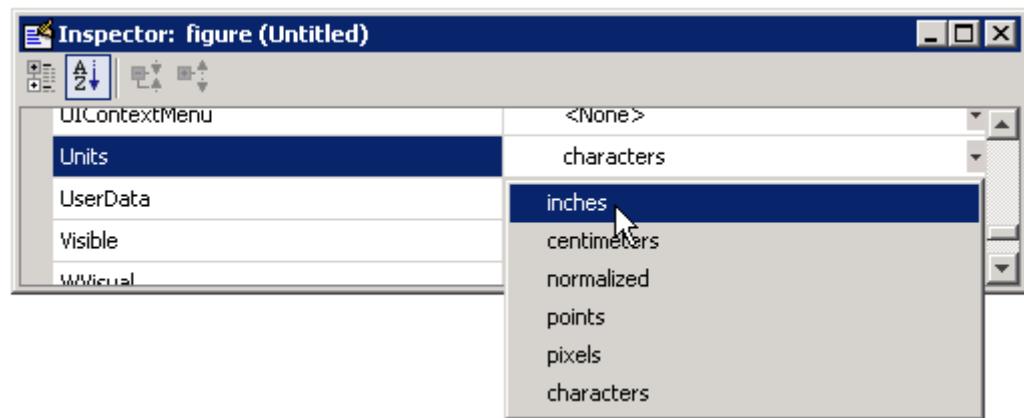


隅のハンドルをドラッグすると、右下隅に表示される値は GUI の現在の位置をピクセル単位で示します。

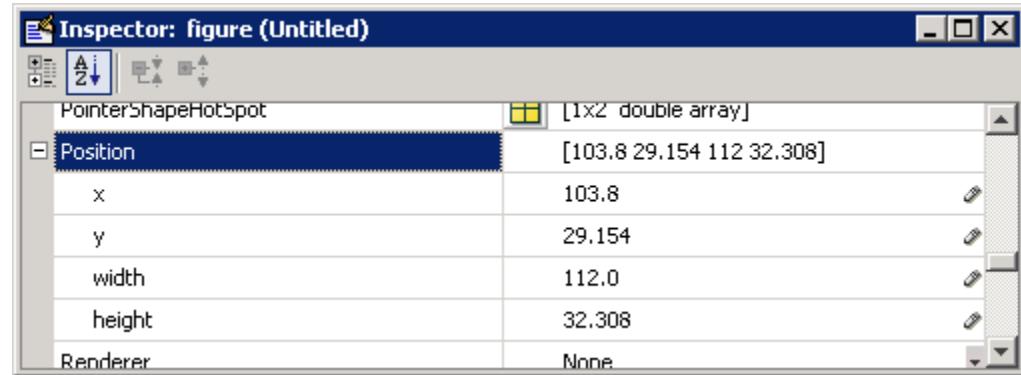
メモ 以下の手順の多くでは、GUI とそのコンポーネントのプロパティを設定するために、プロパティインスペクターを使用する方法を示します。このツールとそのコンテキストセンシティブ ヘルプを使用する方法は、“プロパティインスペクター”(p.6-90)を参照してください。

GUI の位置あるいはサイズを厳密な値に設定したい場合、次のことを行います。

- 1 [表示] メニューから [プロパティインスペクター] を選択するか、あるいは [プロパティインスペクター] ボタンをクリックします。
- 2 Units プロパティまでスクロールして、現在の設定が characters または normalized であるかに注意してください。Units の隣のボタンをクリックしてから、ポップアップメニューから [インチ] の設定を変更します。



- 3 プロパティインスペクターで、Position の隣の + 記号をクリックします。コンポーネントの Position プロパティの要素が表示されます。



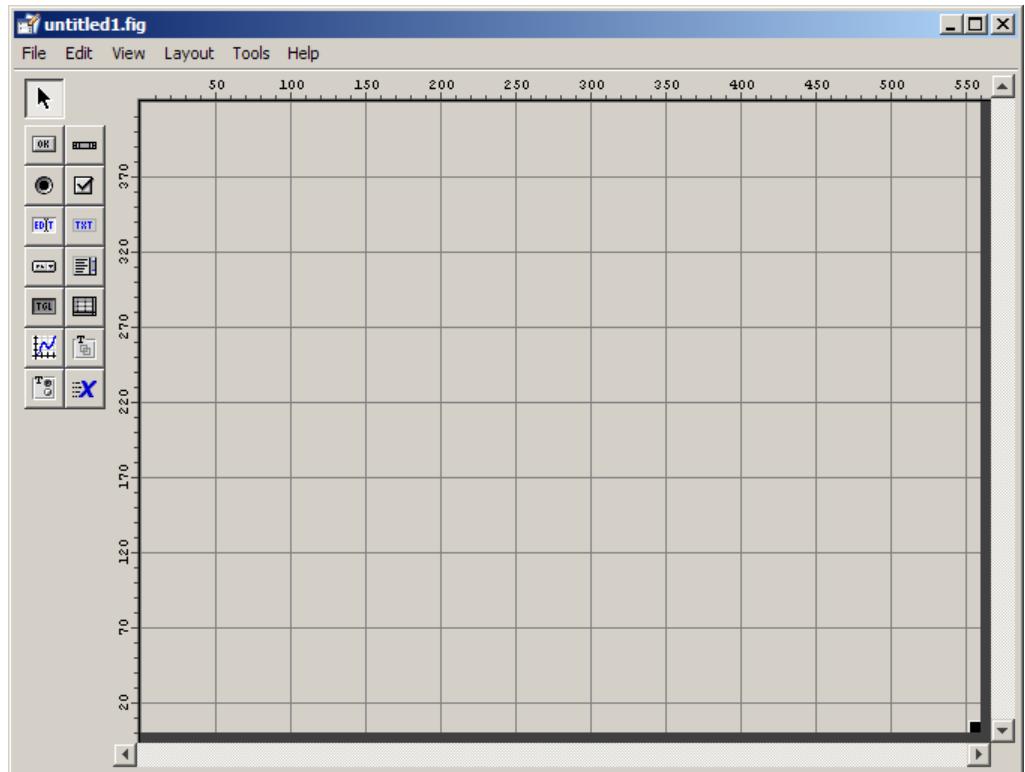
4 GUI の左下隅を表示させたい点の x 座標、y 座標と GUI の幅、高さを入力します。

5 Units プロパティを前の設定、characters または normalized にリセットします。

メモ Units プロパティを characters (サイズ変更不可能な GUI) または normalized (サイズ変更可能な GUI) に設定すると、GUI はプラットフォームに依存せず同じような外見を与えられます。詳細は、“クロスプラットフォーム互換の Units”(p.6-137)を参照してください。

レイアウト エリアを最大化する

GUIDE ツールバー、ステータスバー、これら両方を非表示にすると、レイアウトエディター内のスペースを最大化することができます。これを行うには、[表示] メニューから [ツールバーの表示] および/または [ステータスバーの表示] を非選択にします。コンポーネントパレットにツールアイコンだけ表示すると、さらにスペースができます。コンポーネントパレットにツールアイコンのみを表示するには、GUIDE の [ファイル] メニューから [設定] を選択して、[コンポーネントパレットに名前を表示] を非選択にします。これらすべてのことを行うと、レイアウトエディターは次のようになります。



GUI にコンポーネントを追加

このセクションの内容…

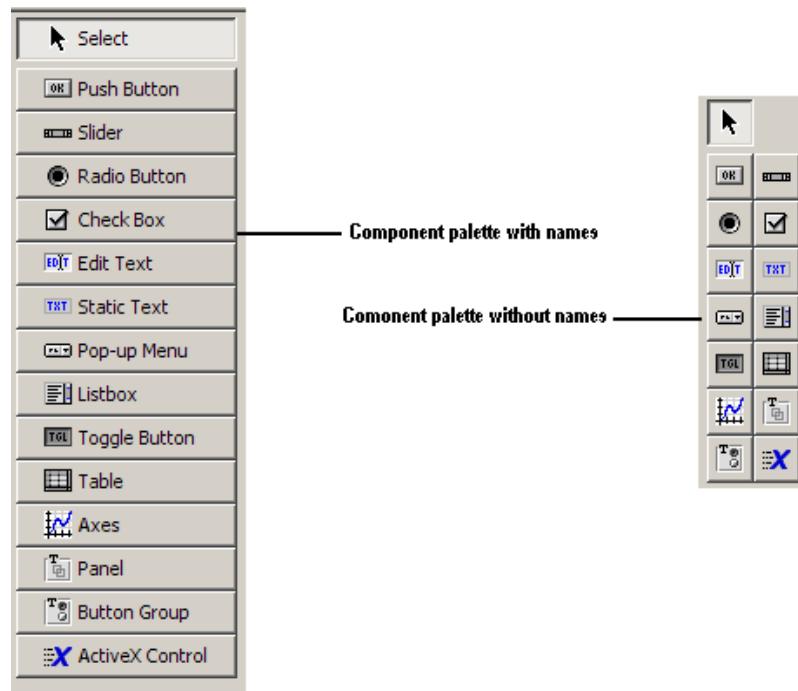
- “利用可能なコンポーネント” (p.6-20)
- “多くのコンポーネントをもつ GUI の取り扱い” (p.6-24)
- “GUIDE レイアウト エリアへのコンポーネントの追加” (p.6-31)
- “ユーザー インターフェイス コントロールの定義” (p.6-38)
- “パネルとボタン グループ定義” (p.6-55)
- “座標軸を定義する” (p.6-60)
- “テーブルを定義する” (p.6-64)
- “ActiveX コントロールの追加” (p.6-75)
- “レイアウト エリア内のコンポーネントの取り扱い” (p.6-78)
- “コンポーネントの位置決めと移動” (p.6-81)
- “コンポーネントのサイズ変更” (p.6-84)

以下は、興味のある他のトピックスです。

- ・ “コンポーネントの整列” (p.6-87)
- ・ “タブの順序の設定” (p.6-96)

利用可能なコンポーネント

レイアウトエディターの左側のコンポーネントパレットには、ユーザーの GUI に追加できるコンポーネントを収容しています。コンポーネントを名前と共にあるいは名前なしで表示できます。



最初にレイアウトエディターを開くと、コンポーネントパレットにはアイコンのみ含まれます。GUI コンポーネントの名前を表示するには、[ファイル] メニューから [プレファレンス] を選択し、[コンポーネントパレットに名前を表示] の隣のボックスをチェックし、[OK] をクリックします。

GUI へのメニューの追加についての詳細は、“メニューの作成”(p.6-99)を参照してください。

メモ ここでは、GUI のレイアウトをするためのコンポーネントの利用方法についての情報を提供します。これらのコンポーネントのプログラミングについての詳細は、章 8, “GUIDE GUI のプログラミング”を参照してください。

コンポーネント	アイコン	説明
プッシュ ボタン		プッシュ ボタンがクリックされると、アクションを実行します。たとえば、[OK] ボタンは、設定を適用し、ダイアログ ボックスを閉じます。プッシュ ボタンをクリックすると、押された状態に見え、マウスをはなすと、プッシュ ボタンの外見が押されていない状態に戻ります。
スライダー		スライダーは、設定した範囲内の数値を、ユーザーが、スライダーまたはサム(thumb)と呼ばれるスライディングバーを移動することにより、設定可能にするものです。ユーザーは、スライダーをクリックしてドラッグしたり、溝または矢印をクリックすることによって、スライダーを移動します。スライダーの位置は、指定した範囲のパーセンテージで表示します。
ラジオ ボタン		ラジオ ボタンは、チェック ボックスに似ていますが、一般的に、関連するラジオ ボタンのグループ内で、互いに排他的になります。つまり、1つのボタンを選択すると、以前に選択されたボタンは非選択になります。ラジオ ボタンをアクティブにするために、オブジェクト上でマウス ボタンをクリックしてください。表示は、ボタンの状態を示します。ボタン グループを用いて、ラジオ ボタンの排他的な選択を管理できます。
チェック ボックス		チェック ボックスは、チェックされたときに動作を起こし、状態が、チェックされている、あるいはチェックされていないかを表示します。チェック ボックスは、たとえば、ツールバーを表示するなど、複数の独立な設定を提供するときに役立ちます。
エディット テキスト		エディット テキスト コンポーネントは、テキスト文字列を入力したり、修正することができるフィールドです。テキストを入力したい場合、エディット テキストを使用してください。ユーザーは数字を入力できますが、等価な数値に変換しなければなりません。

コンポーネント	アイコン	説明
スタティック テキスト		スタティック テキストは、テキストの表示をコントロールします。スタティック テキストは、一般に、他のコントロールのラベル付けに使用したり、ユーザーに対する指示を与えるたり、あるいはスライダーに関連する値を表示するために使用されます。ユーザーはスタティック テキストを対話で変更できません。
ポップアップ メニュー		ポップアップ メニューは、三角形の印を押すと、選択のリストを表示して開きます。
リスト ボックス		リスト ボックスは、項目のリストを表示し、1つあるいは複数の項目を選択することを可能にします。
トグル ボタン		トグル ボタンは、on あるいは off の状態を示し動作します。トグル ボタンをクリックすると、押された状態に見え、on であることを示します。マウス ボタンをはなすと、トグル ボタンを 2 度目にクリックするまでトグル ボタンは押された状態になります。2 度目にクリックすると、ボタンは off であることを示す上がった状態になります。ボタン グループを用いて、トグル ボタンの排他的な選択を管理できます。
テーブル		テーブル コンポーネントを作成するために、テーブル ボタンを使用します。このコンポーネントの利用の詳細は、関数 <code>uitable</code> を参照してください。
座標軸		座標軸により、ユーザー GUI がグラフィックス（たとえば、グラフィックスとイメージ）を表示することが可能になります。他の各グラフィックス オブジェクトのように、座標軸は、その動作と外見の多くの側面をコントロールするため、各プロパティを設定することができます。Axes オブジェクトの詳細は、MATLAB「グラフィックス」ドキュメンテーションの“Axes Properties”と <code>plot</code> 、 <code>surf</code> 、 <code>line</code> 、 <code>bar</code> 、 <code>polar</code> 、 <code>pie</code> 、 <code>contour</code> 、 <code>mesh</code> コマンドを参照してください。さらに詳しいリストとして、MATLAB 関数リファレンスドキュメンテーションの「関数一覧別」を参照してください。

コンポーネント	アイコン	説明
パネル		<p>パネルは、GUI コンポーネントをグループ化します。パネルは、関連するコントロールを視覚的にグループ化して、ユーザー インターフェイスの理解を容易にします。パネルはタイトルと各境界をもつことができます。</p> <p>パネルの子は、座標軸やユーザー インターフェイス コントロールと同様、パネルやボタン グループもなることができます。パネル内の各コンポーネントの位置は、パネルに相対的に解釈されます。パネルを移動する場合、その子もパネルとともに移動し、パネル 上での位置を保ちます。</p>
ボタン グループ		ボタン グループはパネルに似ていますが、ラジオ ボタンやトグル ボタンに対する排他的な選択を管理するために使用されます。
ツールバー		プッシュ ボタンとトグル ボタンを含むツールバーを作成できます。ツールバー ボタンを作成するには、GUIDE のツールバー エディターを使用します。[保存] と[印刷]などのあらかじめ定義されたボタン、あるいはユーザー独自のアイコンとコールバックをカスタマイズするボタンから選択します。
ActiveX® コンポーネント		<p>ActiveX コンポーネントにより、GUI に ActiveX コントロールを表示することができます。これらは、Microsoft® Windows® プラットフォーム上でのみ利用できます。</p> <p>ActiveX コントロールは、Figure すなわち GUI 自身に対してのみ子になることができます。これは、パネルやボタン グループの子になることはできません。</p> <p>例として、このドキュメンテーションの“ActiveX コントロール”(p.8-48)を参照してください。ActiveX コントロールの詳細は、MATLAB「外部インターフェイス」ドキュメンテーションの“Creating COM Objects”を参照してください。</p>

多くのコンポーネントをもつ GUI の取り扱い

GUIDE に GUI を開き、10 種類を超えるデモを行う GUI の例を実行させると、GUI のコンポーネントの外見と動作を確かめることができます。GUI を実行すると、そのコンポーネントのコールバックはすべて GUI を使ってユーザーの動作を伝え、表示するプロットを更新するものもあります。controlsuite と呼ばれる GUI は、前の節“利用可能なコンポーネント”(p.6-20) の表に一覧表示されたコンポーネント (ActiveX コントロールは除く) すべてを含みます。これは、GUIDE に開く FIG-ファイル (controlsuite.fig) と、MATLAB エディターに開く M ファイル (controlsuite.m) で構成されます。

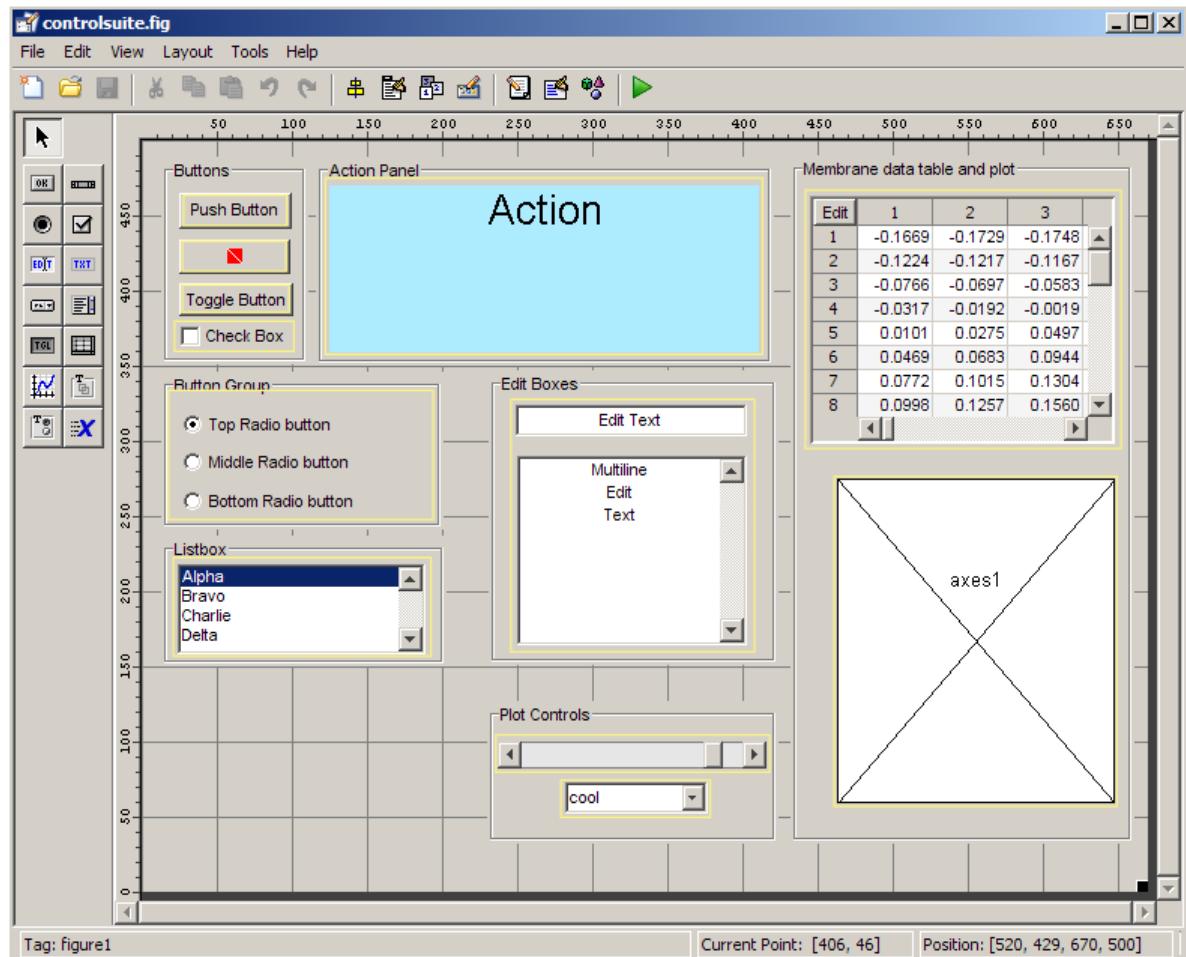
controlsuite のレイアウトと GUI M ファイルの表示

MATLAB ヘルプ ブラウザーでこれを読み頂いている場合、以下のリンクをクリックすると、controlsuite の完成した例が GUIDE レイアウトエディターと MATLAB エディターに表示されます。

- GUIDE レイアウトエディターに controlsuite の GUI を表示するにはここをクリックします。
- MATLAB エディターに controlsuite の GUI M ファイルを表示するにはここをクリックします。
- controlsuite の GUI を実行するにはここをクリックします。

メモ controlsuite GUI を試す場合は、はじめに FIG-ファイルと M ファイルのコピーを、MATLAB パス上の書き込み可能なフォルダーに保存して、これらを取り扱います。

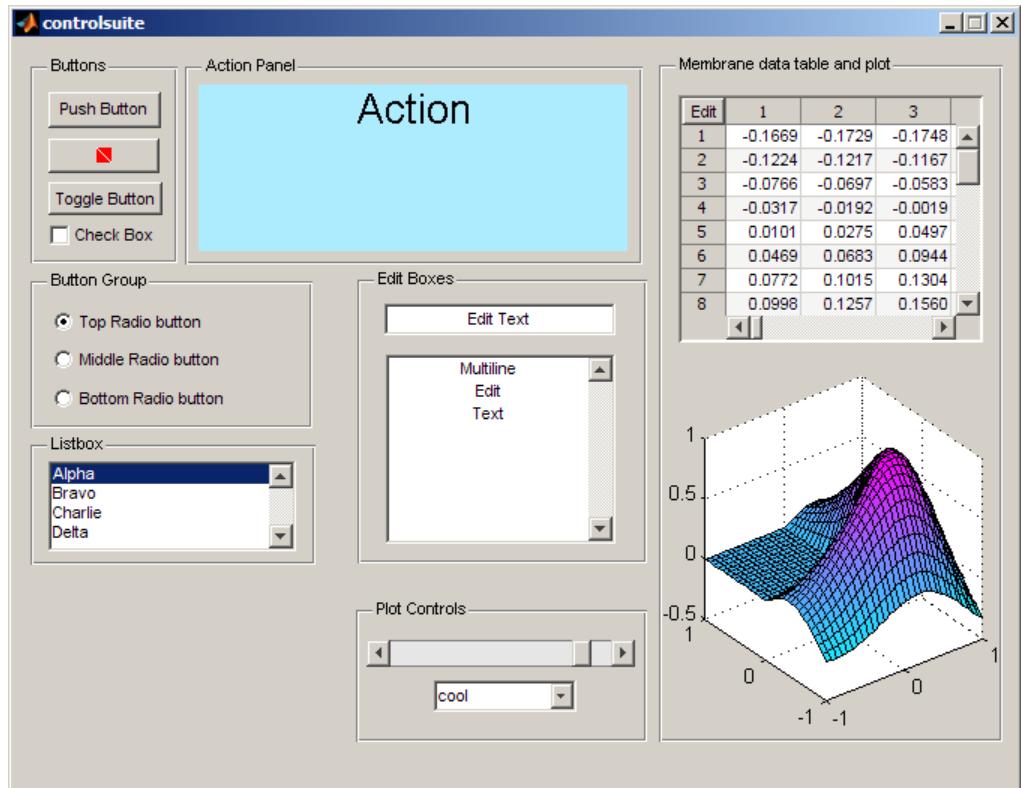
GUIDE に controlsuite.fig を開くと、レイアウトエディターに次のように表示されます。“例:GUIDE GUI コンポーネントのプログラミング”(p.8-30) の節に述べるように、そのタイプのコンポーネントがどのようにプログラムされたかについて参照するには、次の図で黄色にアウトラインされたいずれかのコントロールをクリックします。コントロールのいくつかについては、この節の後半でも説明します。



GUI には、7つのパネル内にまとめられた 15 個のコントロールが含まれます。[Action Panel] には、コントロールを操作するときにユーザーが行う動作を記述するために変化する、スタティック テキスト コンポーネントが含まれます。たとえば、[Listbox] パネルで [Charlie] を選択すると、[Action Panel] に [Charlie] という語が配置されます。右側の Membrane データ テーブルとプロット パネルは、下部の座標軸に 3-D 表面プロット（関数 `membrane` で生成される）を表示し、上部のテーブルにそのプロットに対するデータを表示します。ユーザーは、下中央の [プロット コントロール] パネルで、2 つのコントロールを使用して、方位角の表示とカラーマップをコントロールできます。

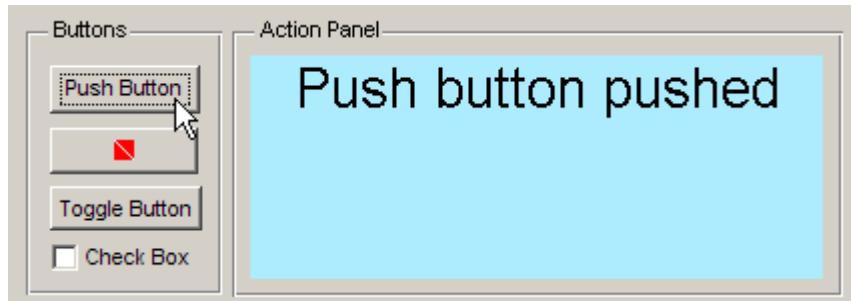
controlsuite GUI の実行

レイアウトエディターの[Figure を実行] ボタンをクリックすると、GUI が開いて、初期化され、次の図に示すようになります。



次の節では、いくつかのコントロールと実行するコード（それらのコールバック）を説明します。controlsuite M ファイルがどのように GUI をコントロールするかを確かめるために、コードのセクションを調べ、下記のコールバックへのリンクをクリックします。

プッシュ ボタン. ユーザーが [プッシュ ボタン] ボタンをクリックすると、結果は次のように [Action Panel] に表示されます。



メッセージを生成する M コードは、下記に示す pushbutton1_Callback の一部です。

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.statusText, 'String', 'Push button pushed')
```

このコールバックは、pushbutton1 がクリックされるとアクティブになります。これは、関数 set を使用して、textStatus というステティック テキスト フィールドに文字列 'Push button pushed' を配置します。controlsuite のコントロールはすべてこの動作を実行しますが、他のことも行うものがあります。

トグル ボタン. [トグル ボタン] ボタンをクリックすると、次のようにになります。



このボタン togglebutton1 のコールバックは、以下の M コードを含みます。

```
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1

isDown = get(hObject,'Value');

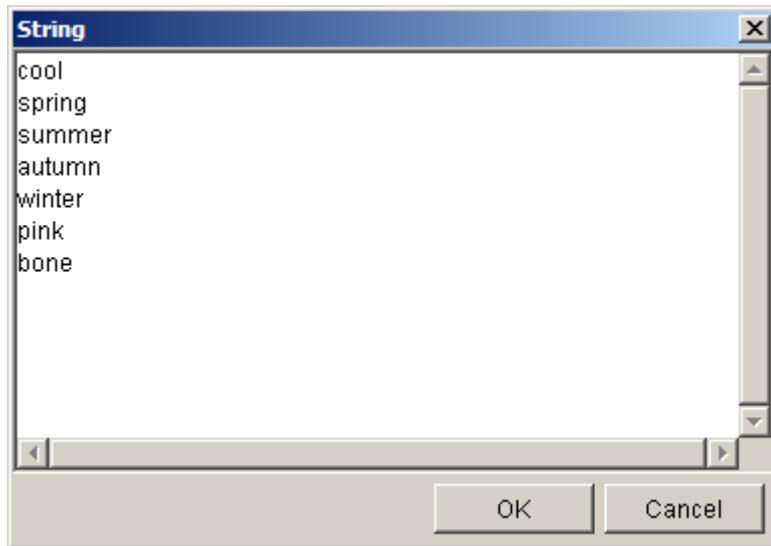
if isDown
    set(handles.statusText, 'string', 'Toggle down')
else
    set(handles.statusText, 'string', 'Toggle up')
end
```

ブロックは変数 `isDown` をテストします。これは、トグル ボタンの `Value` プロパティに設定され、値が `true` であるか `false` であるかに基づき、別のメッセージを書き込みます。

プロットのコントロール. [Plot Controls] パネルの 2 つのコンポーネントは、[Action Panel] のテキストに再び書き込むのと同様に、関数 `peaks` のプロットにも影響します。

- ・ 組み込みのカラーマップを名前により選択するpopupmenu メニュー
- ・ z 軸の周りに表示を回転させるスライダー

`popupmenu1` という名前のpopupmenu は、その `String` プロパティに 7 つのカラーマップの名前の一覧を含みます。これは、その編集アイコンをクリックすることで、プロパティ インスペクターに設定できます。 文字列を編集ダイアログに入力して [OK] をクリックすると、7 つのカラーマップの名前すべてを同時に設定します。



ポップアップのコールバックはその動作をコントロールします。GUIDE は、これだけのコールバックを生成します。

```
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

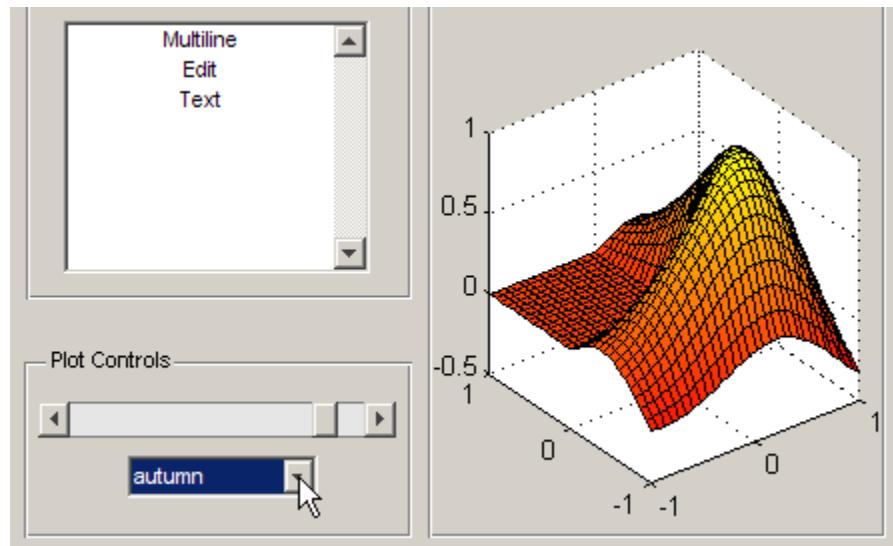
% Hints: contents = get(hObject,'String') returns popupmenu1
%        contents as cell array
%        contents{get(hObject,'Value')} returns selected item
%        from popupmenu1
```

コールバックのコードは、以下のステートメントを追加します。

```
contents      = get(hObject,'String');
selectedText = contents{get(hObject,'Value')};
colormapStatus = [selectedText ' colormap'];
set(handles.statusText, 'string', colormapStatus);
colormap(selectedText)
```

String データは、セル配列として読み出され、contents に割り当てられます。Value プロパティは、カラーマップの名前を取り出すようにユーザーが選択した contents の

メンバーにインデックスを付けます。その名前である、`selectedText` は、メッセージに構成され、スタティック テキスト フィールド `textStatus` に配置され、カラーマップをリセットする関数 `colormap` で使用されます。たとえば、ユーザーがポップアップメニューから [autumn] を選択すると、GUI からの以下の切り取りで示すように、表面は黄色、オレンジ、赤の色調に変化します。



スライダー コントロールは、方位角の表示を設定します。ユーザーがその“サム”を移動するか、その矢印をクリックすると、プロットが回転します。その名前は `slider1` であり、そのコールバックは `slider1_Callback` です。

データ テーブル. 右上隅のテーブルは、`uitable` コンポーネントです。GUI が作成されると、テーブルの `CreateFcn` は同じ `membrane` データを、その下にある座標軸内のプロットが表示するテーブルに読み込みます。

このテーブルは既定では編集可能ではありませんが、その左上隅の小さな [編集] トグル ボタンをクリックすると、ユーザーはテーブルの値を 1 つずつ編集できます。このボタンは、テーブルの最上部にありますが、実際にはテーブルの一部ではありません。テーブルの `CellEditCallback` は、表面プロットを更新し、Action Panel の編集の結果を表示して、テーブル セルそれぞれの編集に応答します。再び [編集] ボタンをクリックすると、テーブルは編集できなくなります。この動作の詳細は、`controlsuite M` ファイルの `togglebutton2_Callback` を参照してください。

uitable をグラフィックスと組合せる方法を述べた他の例として、“テーブルのデータを対話的に調べる GUI”(p.10-32) を参照してください。

“例:GUIDE GUI コンポーネントのプログラミング”(p.8-30) の controlsuite と以下の節で使用される GUI コンポーネントを取り扱う方法の詳細は、以下の節で参照してください。

- ・ “ユーザー インターフェイス コントロールの定義”(p.6-38)
- ・ “パネルとボタン グループ定義”(p.6-55)
- ・ “座標軸を定義する”(p.6-60)
- ・ “テーブルを定義する”(p.6-64)

GUIDE レイアウト エリアへのコンポーネントの追加

このトピックでは、GUIDE レイアウト エリア にコンポーネントを配置する方法を述べ、各コンポーネントに一意的な識別子を与えます。

メモ GUI へのメニューの追加についての詳細は、“メニューの作成”(p.6-99)を参照してください。ツールバーの取り扱いについての詳細は、“ツールバーの作成”(p.6-120)を参照してください。

1 ユーザーの設計に従い、レイアウト エリアにコンポーネントを配置します。

- ・ パレットからコンポーネントをドラッグして、レイアウト エリアにドロップします。
- ・ パレットのコンポーネントをクリックしてから、カーソルをレイアウト エリアに移動させます。カーソルが十字の形に変わります。コンポーネントを既定値のサイズで追加するには再びクリックします。あるいは、コンポーネントを追加するときにサイズを決めるには、クリックしながらドラッグします。

一旦、レイアウト エリアに GUI コンポーネントを定義してから、それを選択すると、そのコンポーネントがプロパティ インスペクターに自動的に表示されます。プロパティ インスペクターが開いていないか、または表示されていない場合、コンポーネントをダブルクリックするとインスペクターが起動し、そのコンポーネントをフォーカスします。

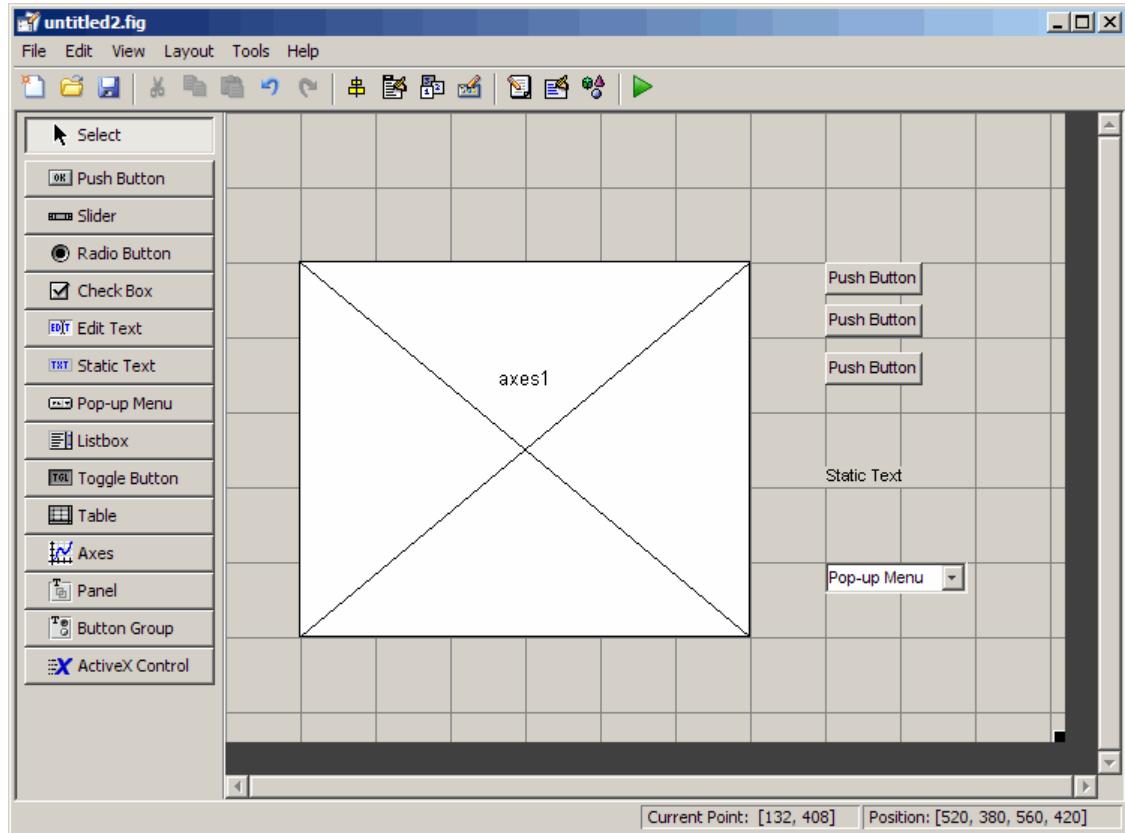
次の表に一覧表示されたコンポーネントは、さらに考慮すべきことがあります。詳細は、以下に記載されている節を参照してください。

追加するコンポーネント	説明
パネルまたはボタン グループ	“パネルまたはボタン グループへのコンポーネントの追加”(p.6-34)を参照。
メニュー	“メニューの作成”(p.6-99)を参照。
ツールバー	“ツールバーの作成”(p.6-120)を参照。
ActiveX コントロール	“ActiveX コントロールの追加”(p.6-75)を参照。

グリッドの利用についての詳細は、“グリッドとルーラ”(p.6-94)を参照してください。

- 2 各コンポーネントに一意的な識別子を割り当てます。コンポーネントの Tag プロパティの値に設定して行います。詳細は、“各コンポーネントへの識別子の割り当て”(p.6-37)を参照してください。
- 3 適当なプロパティを設定して、各コンポーネントの外観を指定します。次のトピックでは、特定の情報を説明します。
 - ・ “ユーザー インターフェイス コントロールの定義”(p.6-38)
 - ・ “パネルとボタン グループ定義”(p.6-55)
 - ・ “座標軸を定義する”(p.6-60)
 - ・ “テーブルを定義する”(p.6-64)
 - ・ “ActiveX コントロールの追加”(p.6-75)

これは、レイアウトエディター内の GUI の例です。レイアウトエディター内のコンポーネントは、アクティブではありません。章 7, “GUIDE GUI の保存と実行” では、機能する GUI の実行方法について説明します。



Place コンポーネントに座標軸を利用する

GUIDE レイアウトエディターの下部のステータスバーは、以下を表示します。

- Current Point – レイアウトエディターのグリッドエリアの左下隅に対するマウスの現在位置。
- Position – 選択されたコンポーネントの Position プロパティ。[左端からの距離、下端からの距離、幅、高さ] の 4 要素のベクトル。距離は、親の Figure、パネ

ル、ボタン グループに対する距離です。値は、ピクセル単位で与えられます。ルーラも、ピクセル表示です。

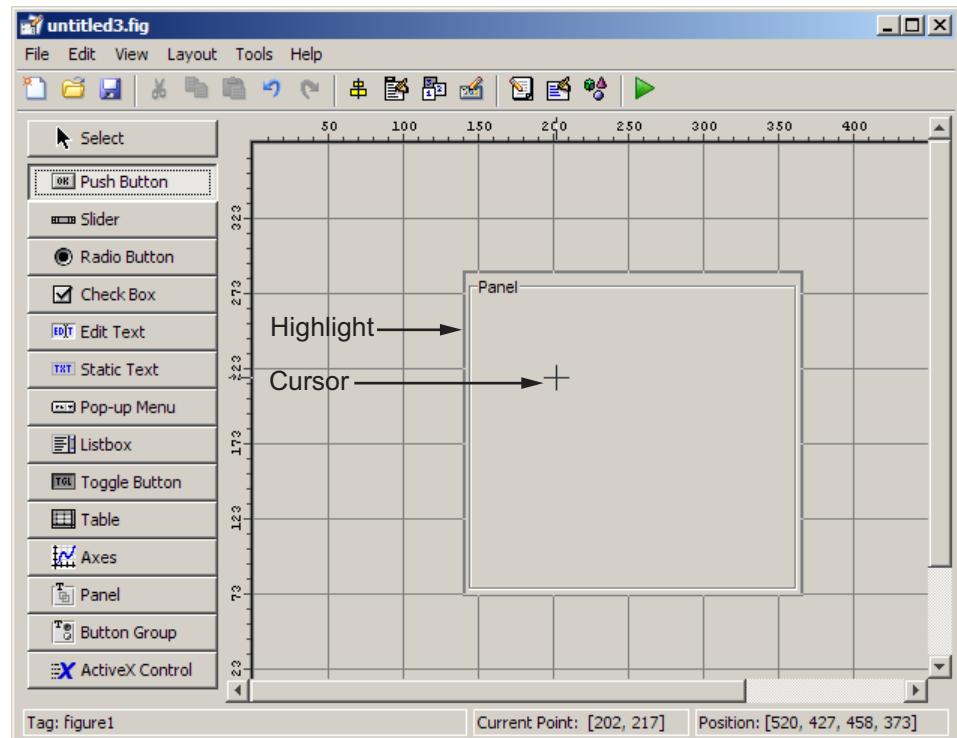
1 つのコンポーネントを選択して移動すると、位置ベクトル（左からの距離、下からの距離）の最初の 2 要素は、コンポーネントの移動とともに更新されます。コンポーネントをサイズ変更する場合、位置ベクトル（幅、高さ）の最後の 2 要素は、ユーザーがサイズを変更すると更新されます。コンポーネントの左下隅の位置が変化するように、ユーザーがコンポーネントをサイズ変更すると、最初の 2 要素も変更されます。コンポーネントが選択されていない場合、位置ベクトルは、Figure の位置ベクトルです。

詳細は、“座標表示の利用”(p.6-81)を参照してください。

パネルまたはボタン グループへのコンポーネントの追加

パネルまたはボタン グループにコンポーネントを追加するには、コンポーネント パレットでコンポーネントを選択して、希望するパネルまたはボタン グループの上にカーソルを移動します。カーソルの位置は、コンポーネントの親を決めます。

次の図に示すように、GUIDE が親をハイライトします。ユーザーがコンポーネントをロップしたり、カーソルをクリックする場合、ハイライトは、そのコンポーネントがハイライトされたパネル、ボタン グループ、または、Figure の子になることを示します。



メモ パネルやボタン グループの各コンポーネントに Tag プロパティの値を設定することによって、これらの一意的な識別子を割り当てます。詳細は、“各コンポーネントへの識別子の割り当て”(p.6-37)を参照してください。

パネルとボタン グループに既存のコンポーネントを含める。パネルまたはボタン グループに新規のコンポーネントを追加したり、既存のコンポーネントをドラッグすると、コンポーネントは、完全にまたは部分的に囲まれていても、パネルまたはボタン グループの、メンバーまたは子に自動的になります。ただし、そのコンポーネントがパネルまたはボタン グループに完全には含まれていないと、レイアウトエディターにクリップされるように見えます。GUI を実行すると、パネルとボタン グループにまたがって表示されます。このコンポーネントはパネルの子であり、パネルに従い動作します。オブジェクトブラウザを使用して、パネルまたはボタン グループの子オブジェクトを確認することができます。“オブジェクト階層の表示”(p.6-134)に、方法を説明します。

既存のコントロールをグループ化するために、新規のパネルまたはボタン グループを GUI に追加できます。そのようなコントロールを、新規のパネルやボタン グループに含めるには、以下を行います。ここでは、パネルについて説明しますが、ボタン グループ内のコンポーネントに対しても同じことを行います。

- 1 新規のパネルまたは新規のボタン グループ ツールを選択して、目的のサイズと位置をもつ長方形をドラッグ アウトします。

パネルは、Axes、Table、または他のパネルやボタン グループでない限り、その境界内のどのコントロールも表示します。ネストしたいパネルのみがオーバーラップされ、オーバーラップが完全であることを確認します。

- 2 レイアウトにこのような問題があるときには、表示させない新規のパネルをコンポーネントの裏に置きます。これは、[レイアウト] メニューの [最背面へ移動] または [背面へ移動] を使用して行うことができます。このパネルにコンポーネントを追加したり、またはコンポーネントをドラッグすると、このパネルは他のパネルの後ろに自動的に置かれます。

ここで、プロパティインスペクターを使用して、パネルの Tag プロパティと String プロパティを、目的のものに設定すると良いでしょう。

- 3 [表示] メニューからオブジェクトブラウザを開き、追加したパネルを見つけます。このツールを使い、グループ化するために目的としたすべてのコントロールが含まれていることを確かめます。欠けているものがあれば、以下の手順を実行します。

- 4 含めようとするコントロールをドラッグしますが、パネル内でここでは目的の位置に合わせません。さらに、パネルを用いてそれらをグループ化するように、すでに適切な位置にある、コントロールをわずかに移動させます。

パネルは、ユーザーがコントロールを移動するときに、コントロールを含むことを示す強調表示を行います。オブジェクトブラウザは、関係を確認するために更新します。パネルを移動するとき、その子のコントロールはパネルとともに移動します。

ヒント 周囲のパネルまたはボタン グループとともにコントロールを登録するためには、1、2 ピクセルでもマウスを用いてコントロールを移動する必要があります。複数のコントロールを選択して矢印キーを使用して移動することはできません。オブジェクトブラウザを使用して、そのコントロールが適切に入れ子になっていることを確かめます。

パネルとボタン グループを GUI に組み込む方法の詳細は、“パネルとボタン グループ定義”(p.6-55)を参照してください。

各コンポーネントへの識別子の割り当て

各コンポーネントに意味のある文字列の識別子を割り当てるには、Tag プロパティを使用します。

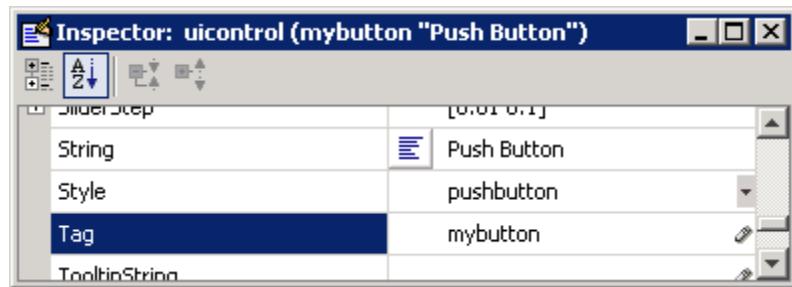
レイアウトエリアにコンポーネントを配置する場合、GUIDE は既定値を Tag プロパティに割り当てます。GUI を保存する前に、この値を GUI 内でコンポーネントの役割を表す文字列で置き換えます。

Tag に割り当てる文字列の値は、コンポーネントを識別するために M ファイルコードで使用され、GUI で一意でなければなりません。Tag を設定するには、以下を行います。

1 [表示] メニューから [プロパティ インスペクター] を選択するか、あるいは [プロパティ インスペクター] ボタンをクリックします。

2 レイアウトエリアで、Tag を設定したいコンポーネントを選択します。

- 3 プロパティ インスペクターで、[Tag] を選択してからその値を、識別子として使用したい文字列で置き換えます。次の図で、Tag は mybutton に設定されます。



ユーザー インターフェイス コントロールの定義

ユーザー インターフェイス コントロールは、プッシュ ボタン、トグル ボタン、スライダー、ラジオ ボタン、エディット テキスト コントロール、スタティック テキスト コントロール、ポップアップ メニュー、チェック ボックス、リスト ボックスを含みます。

ユーザー インターフェイス コントロールを定義するためには、いくつかのプロパティを設定する必要があります。このためには、以下のことを行います。

- 1 プロパティ インスペクターを使用して、該当するプロパティを修正します。[表示] メニューから [プロパティ インスペクター] を選択するか、あるいは [プロパティ インスペクター] ボタンをクリックして、プロパティ インスペクターを開きます。
- 2 レイアウト エリアで、定義しているコンポーネントを選択します。

以下のトピックスでは、ユーザー インターフェイス コントロールの一般に利用されるプロパティについて述べ、各コントロールに対する簡単な例を提供します。

- ・ “一般に利用するプロパティ” (p.6-39)
- ・ “プッシュ ボタン” (p.6-40)
- ・ “スライダー” (p.6-42)
- ・ “ラジオ ボタン” (p.6-43)
- ・ “チェック ボックス” (p.6-45)
- ・ “エディット テキスト” (p.6-46)

- ・ “スタティック テキスト” (p.6-48)
- ・ “ポップアップ メニュー” (p.6-49)
- ・ “リスト ボックス” (p.6-51)
- ・ “トグル ボタン” (p.6-53)

メモ これらのコンポーネントの詳細は、“利用可能なコンポーネント” (p.6-20)を参照してください。これらのコンポーネントのプログラミングの基本的な例として、“例:GUIDE GUI コンポーネントのプログラミング” (p.8-30)を参照してください。

一般に利用するプロパティ

次の表では、ユーザー インターフェイス コントロールを説明するために必要となる、最も一般に利用されるプロパティを示します。特定のコントロールの説明では、そのコントロールに固有のプロパティもリストします。

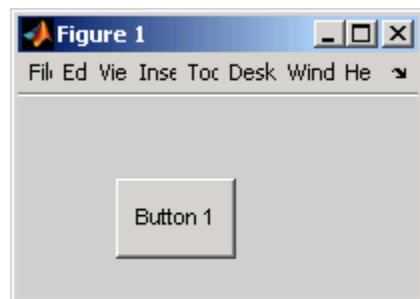
プロパティ	値	説明
Enable	on、inactive、off。既定値は on です。	コントロールがユーザーに利用可能であるかどうかを決めます。
Max	スカラー。既定値は 1 です。	最大値。説明はコンポーネントのタイプに依存します。
Min	スカラー。既定値は 0 です。	最小値。説明はコンポーネントのタイプに依存します。
Position	4 要素ベクトル。[左端からの距離、下端からの距離、幅、高さ]。	コンポーネントのサイズとその親に対する相対的位置。
String	文字列。文字列のセル配列または文字配列になることもあります。	コンポーネントラベル。リスト ボックスとポップアップ メニューの場合は、項目のリストです。

プロパティ	値	説明
Units	characters、centimeters、inches、normalized、pixels、points。既定値はcharacters です。	Position プロパティ ベクトルを処理するために用いられる測定の単位。
Value	スカラーまたはベクトル。	コンポーネントの値。説明はコンポーネントのタイプに依存します。

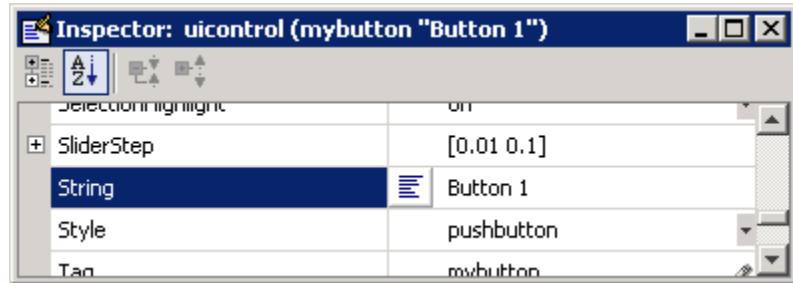
プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB ドキュメンテーションの「Uicontrol プロパティ」を参照してください。GUI の動作をコントロールするために必要なプロパティは、章 8, “GUIDE GUI のプログラミング”に説明します。

プッシュ ボタン

この図に示すように、ラベル Button 1 が付いたプッシュ ボタンを作成するには、



- String プロパティに付けたいラベルを設定して、プッシュ ボタンのラベルを指定します。この場合は、Button 1 にします。



ラベルに “&” 文字を表示するには、文字列に 2 つの & 文字を使用します。
remove、default、factory (大文字と小文字の区別あり) は予約語です。これらの
いずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ (\) を付
けます。たとえば、\\$remove は remove を与えます。

プッシュボタンのテキストは 1 行に限られます。2 行以上指定すると、最初の行の
みが表示されます。指定された String を収められないような小さなプッシュボタ
ンを作成すると、MATLAB は省略記号を用いて文字列を短くします。



- コンポーネントの位置やサイズを正確な値に設定したい場合、その Position プロ
パティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コン
ポーネントのサイズ変更”(p.6-84)を参照してください。
- プッシュボタンにイメージを追加するには、ボタンの CData プロパティに、トゥル
ー カラー イメージを定義する RGB 値の $m \times n \times 3$ 配列を割り当てます。これは、
GUI M ファイルの opening 関数内のプログラムで、行わなければなりません。た
とえば、配列 img は (rand で生成される) 0 から 1 までの乱数の値を使用して、
 $16 \times 64 \times 3$ トゥルーカラー イメージを定義します。

```
img = rand(16, 64, 3);
set(handles.pushbutton1, 'CData', img);
```

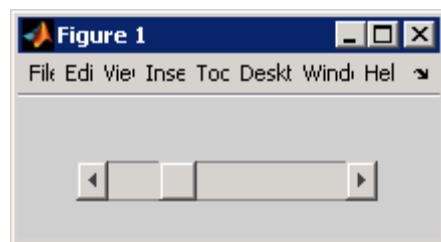
pushbutton1 はプッシュボタンの Tag プロパティです。



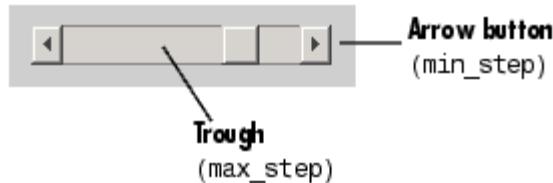
メモ “アイコン エディター” (p.15-60)に述べるアイコン エディターを用いてユーザー独自のアイコンを作成します。行列 X の変換と対応するカラーマップ、すなわち、 (X, MAP) イメージから RGB (トゥルーカラー) 形式への変換の詳細は、関数 `ind2rgb` を参照してください。

スライダー

この図に示すように、スライダーを作成するには、



- スライダーの `Min` プロパティをスライダーの最小値に、`Max` プロパティを最大値に設定して、スライダーの範囲を指定します。`Min` プロパティは、`Max` より小さくなければなりません。
- `Value` プロパティを適切な数値に設定することによってスライダーが作成される場合、スライダーが示す値を指定します。この数値は、`Min` 以上で `Max` 以下でなければなりません。指定した範囲外を `Value` に指定すると、スライダーは表示されません。
- ユーザーが最小ステップのために矢印ボタンをクリックしたり、最大ステップのためにスライダーの溝をクリックするときに、`SliderStep` プロパティを設定することによって、スライダーの `Value` が変化する量をコントロールします。`SliderStep` を 2 要素ベクトル $[min_step, max_step]$ として定義します。範囲 $[0, 1]$ の各値は、範囲のパーセンテージを示します。



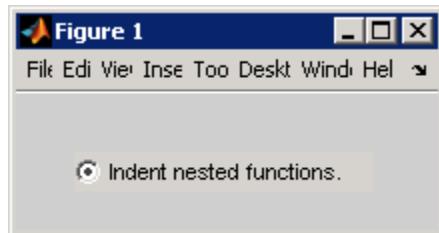
- コンポーネントの位置やサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。

メモ Mac® プラットフォームでは、水平方向のスライダーの高さに制限があります。位置ベクトルに設定した高さがこの制限を超えると、スライダーの表示される高さは、許容される最大値になります。位置ベクトルの要素 height は変更されません。

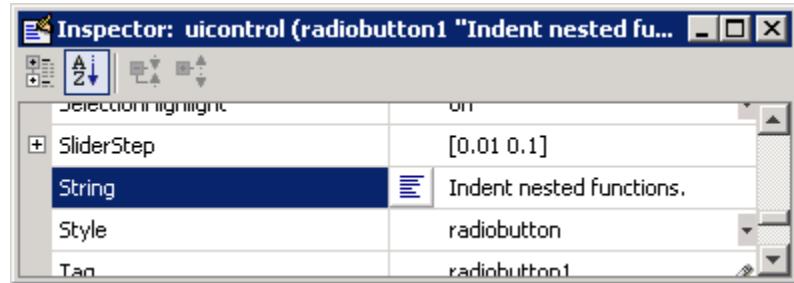
メモ スライダー コンポーネントには、説明のテキストのやデータ入力機能がありません。スライダーにラベリングするには、“スタティック テキスト”(p.6-48) コンポーネントを使用します。“エディット テキスト”(p.6-46) コンポーネントを使用して、スライダーに対して値を与えることができます。

ラジオ ボタン

この図に示すように、ラベル Indent nested functions が付いたラジオ ボタンを作成するには、



- String プロパティに付けたいラベルを設定して、ラジオ ボタンのラベルを指定します。この場合は、Indent nested functions とします。



ラベルに“&”文字を表示するには、文字列に2つの&文字を使用します。
remove、default、factory(大文字と小文字の区別あり)は予約語です。これらの
いずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ(\\$)を付
けます。たとえば、\\$removeはremoveを与えます。

ラジオボタンのテキストは1行に限られます。2行以上指定すると、最初の行のみが表示されます。指定されたStringを収められないような小さなプッシュボタンを作成すると、MATLABは省略記号を用いて文字列を短くします。



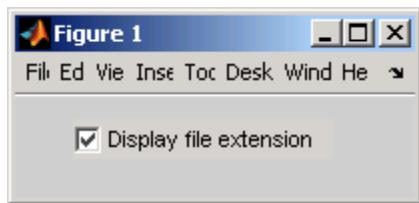
- ラジオボタンのValueプロパティを、ラジオボタンのMaxプロパティの値(既定値は1)に設定すると、選択されたボタンをもつラジオボタンを作成します。ラジオボタンを選択されていない状態にするには、ValueをMin(既定値は0)に設定します。これに対応して、ユーザーが、ラジオボタンを選択すると、MATLABはValueをMaxに設定し、選択しないと、ValueをMinに設定します。
- コンポーネントの位置やサイズを正確な値に設定したい場合、そのPositionプロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。
- ラジオボタンにイメージを追加するには、ボタンのCDataプロパティに、トゥルーカラーイメージを定義するRGB値の $m \times n \times 3$ 配列を割り当てます。これは、GUIDE Mファイルのopening関数内のプログラムで、行わなければなりません。たとえば、配列imgは(randで生成される)0から1までの乱数の値を使用して、 $16 \times 24 \times 3$ トゥルーカラーイメージを定義します。

```
img = rand(16, 24, 3);
set(handles.radiobutton1, 'CData', img);
```

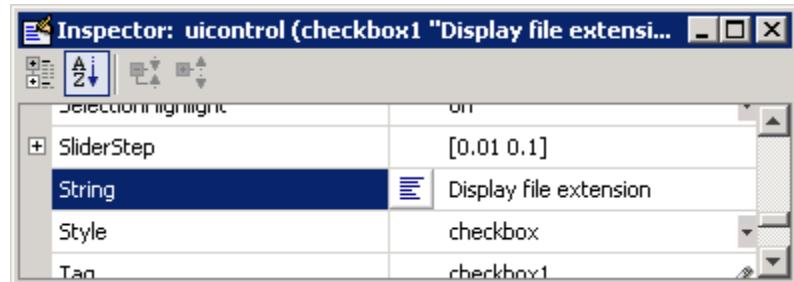
メモ ラジオ ボタンとトグル ボタンの排他的な選択を処理するには、これらをボタン グループに置きます。詳細は、“ボタン グループ”(p.6-58)を参照してください。

チェック ボックス

この図に示すように、ラベル Display file extension が付いた、初めにチェックされているチェック ボックスを作成するには、以下のことを行います。



- String プロパティに付けたいラベルを設定して、チェック ボックスのラベルを指定します。この場合は、Display file extension とします。



ラベルに “&” 文字を表示するには、文字列に 2 つの & 文字を使用します。
remove、default、factory (大文字と小文字の区別あり) は予約語です。これらの
いずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ (\) を付
けます。たとえば、\\$remove は remove を与えます。

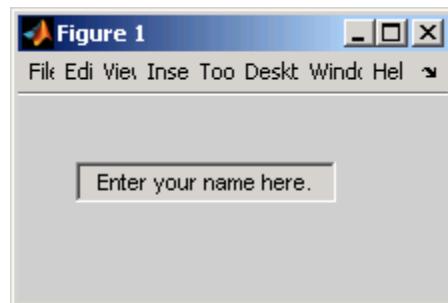
チェック ボックスのテキストは 1 行に限られます。指定された String を収めら
れないような幅の狭いコンポーネントを指定すると、MATLAB は省略記号を
用いて文字列を短くします。



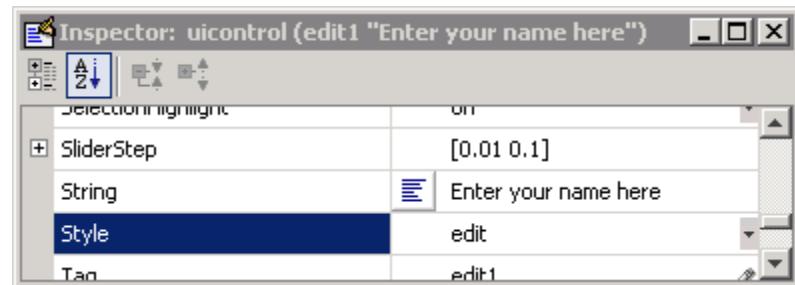
- Value プロパティを Max プロパティの値（既定値は 1）に設定すると、チェックされた状態のチェックボックスを作成します。チェックボックスがチェックされていない状態にするには、Value を Min（既定値は 0）に設定します。これに対して、ユーザーがチェックボックスをチェックすると、MATLAB は Value を Max に設定し、チェックしないと、Min に設定します。
- コンポーネントの位置やサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”（p.6-81）と“コンポーネントのサイズ変更”（p.6-84）を参照してください。

エディットテキスト

次の図に示すように、初めにテキスト Enter your name here を表示するエディットテキストコンポーネントを作成するには、以下を行います。



- String プロパティに表示したい文字列を設定して、エディットテキストコンポーネントを作成するときに表示されるテキストを指定します。この場合は、Enter your name here とします。



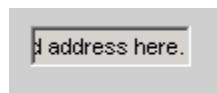
ラベルに “&” 文字を表示するには、文字列に 2 つの & 文字を使用します。
remove、default、factory（大文字と小文字の区別あり）は予約語です。これらの

いずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ(¥)を付けています。たとえば、¥remove は remove を与えます。

- 複数行の入力を可能にするには、Max と Min プロパティの差が 1 よりも大きくなるようにこれらのプロパティを指定します。たとえば、Max = 2、Min = 0 とします。Max 既定値は 1 です。Min 既定値は 0 です。MATLAB は、必要であれば、文字列を折り返し、スクロールバーを追加します。すべてのプラットフォームで、ユーザーが Tab キーを用いて複数行のテキストボックスを入力すると、編集カーソルが以前の位置に配置され、テキストはハイライトされません。



Max-Min が 1 以下の場合、エディット テキスト コンポーネントは 1 行のみの入力ができます。指定された文字列を収められないような幅の狭いコンポーネントを指定すると、MATLAB は文字列の一部のみを表示します。矢印キーを使用して、文字列内でカーソルを移動させることができます。すべてのプラットフォームにおいて、ユーザーが Tab キーを利用して 1 行のテキストボックスを入力すると、コンテンツ全体が強調表示され、カーソルは文字列の端(右側)にあります。

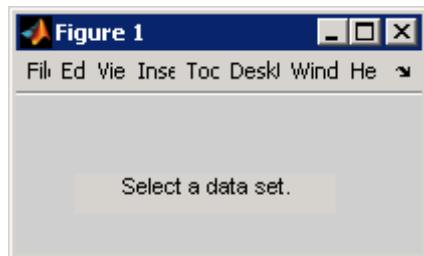


- コンポーネントの位置やサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。
- プロパティインスペクターの FontName の項目に、システムにあるフォントの名前を入力して、エディット ボックスで表示するテキストフォントを指定します。Microsoft Windows プラットフォームでは、既定値は MS Sans Serif です。Macintosh® と UNIX® プラットフォームでは、既定値は Helvetica です。

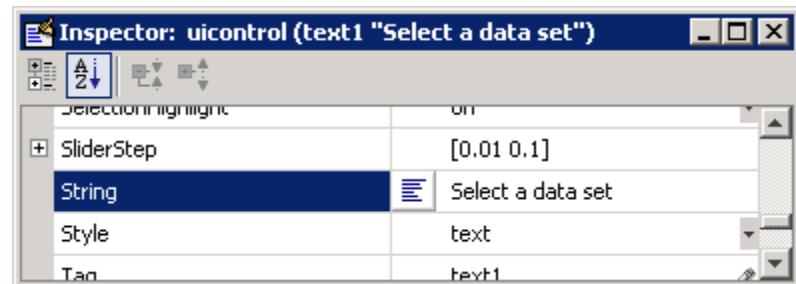
ヒント このフォントが利用できるかどうかを見つけるために、MATLAB プロンプトで `uisetfont` と入力します。リストボックスを含むダイアログが表示されます。利用可能なフォントがこのリストボックスに表示され、ユーザーが選択できます。フォントを選択すると、その名前と他の特性が構造体に返されます。この構造体から文字列 `FontName` をコピーして、プロパティインスペクターに貼り付けることができます。一覧表示されたフォントには、ユーザーのシステムで GUI のユーザーが利用できないものもあります。

スタティック テキスト

この図に示すように、テキスト `Select a data set` を表示するスタティック テキストコンポーネントを作成するには、次のようにします。



- コンポーネントの `String` プロパティに表示したいテキストを設定して、コンポーネントに表示されるテキストを指定します。この場合は、`Select a data set` にします。



リスト項目に "&" 文字を表示するには、文字列に 2 つの & 文字を使用します。`remove`、`default`、`factory`（大文字と小文字の区別あり）は予約語です。これらのいずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ（¥）を付けます。たとえば、¥remove は `remove` を与えます。

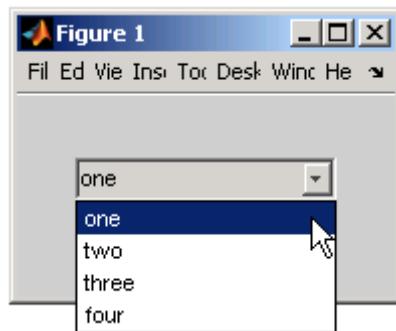
コンポーネントに指定した String を収めるのに十分な幅がない場合、MATLAB は文字列を覆します。



- コンポーネントの位置やサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。
- その FontName、FontWeight、FontAngle、FontSize、FontUnits プロパティを含む、テキストフォントを指定できます。詳細は、以前のトピックス “エディット テキスト”(p.6-46) と、プログラミングによる方法では “フォントの特性の設定”(p.11-18) の節を参照してください。

ポップアップ メニュー

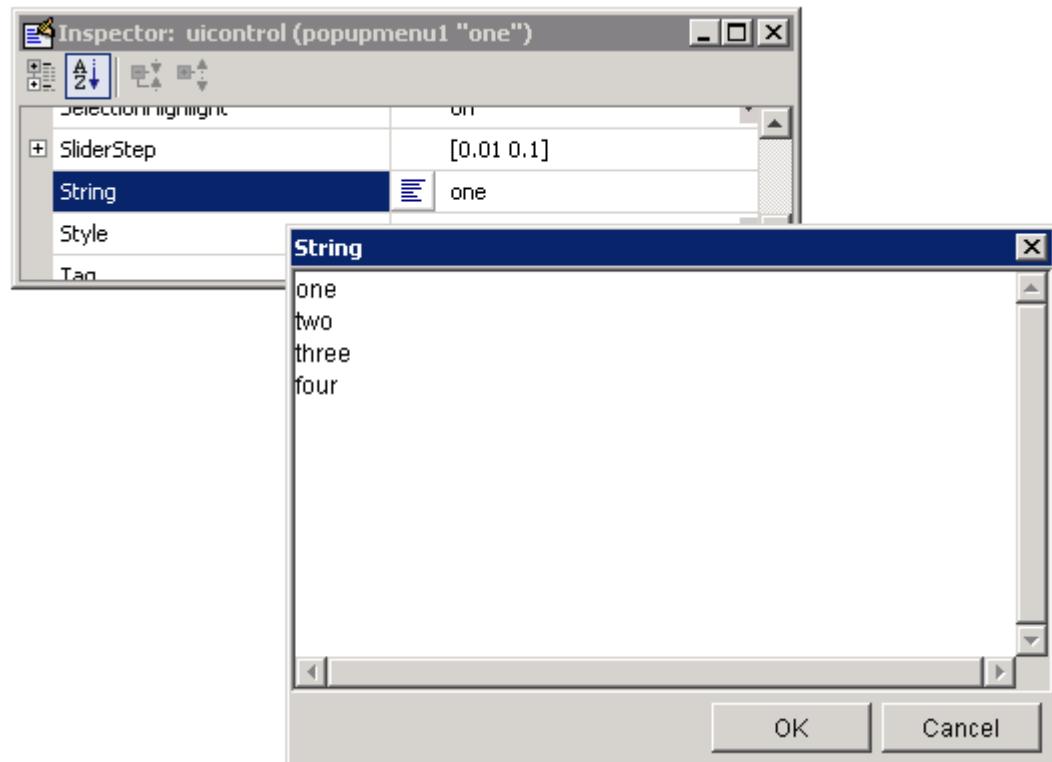
次の図に示すように、項目 one、two、three、four をもつ、(ドロップダウン メニューまたはコンボ ボックスとも呼ばれる) ポップアップ メニューを作成するには、次のようにします。



- String プロパティに表示したい項目を設定することによって、表示されるポップアップ メニュー項目を指定します。プロパティ名の右の



ボタンをクリックすると、プロパティ インスペクター エディターが開きます。



メニュー項目に“&”文字を表示するには、文字列に2つの&文字を使用します。
remove、default、factory(大文字と小文字の区別あり)は予約語です。これらの
いずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ(\\$)を付
けます。たとえば、\\$removeはremoveを与えます。

コンポーネントの幅が狭く、指定した文字列が1つ以上表示できない場合、
MATLABは省略記号を用いてこれらの文字列を短くします。

- コンポーネントが作成されるときに1つの項目を選択するには、Valueを選択
したリスト項目のインデックスを示すスカラーに設定します。ここで1はリストの
最初の項目に対応します。Valueを2に設定すると、メニューが作成されたとき
に、次のように見えます。

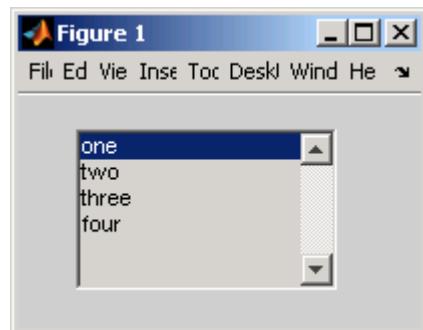


- コンポーネントの位置とサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。ポップアップメニューの高さは、フォントのサイズで決まります。位置ベクトルに設定する高さは無視されます。

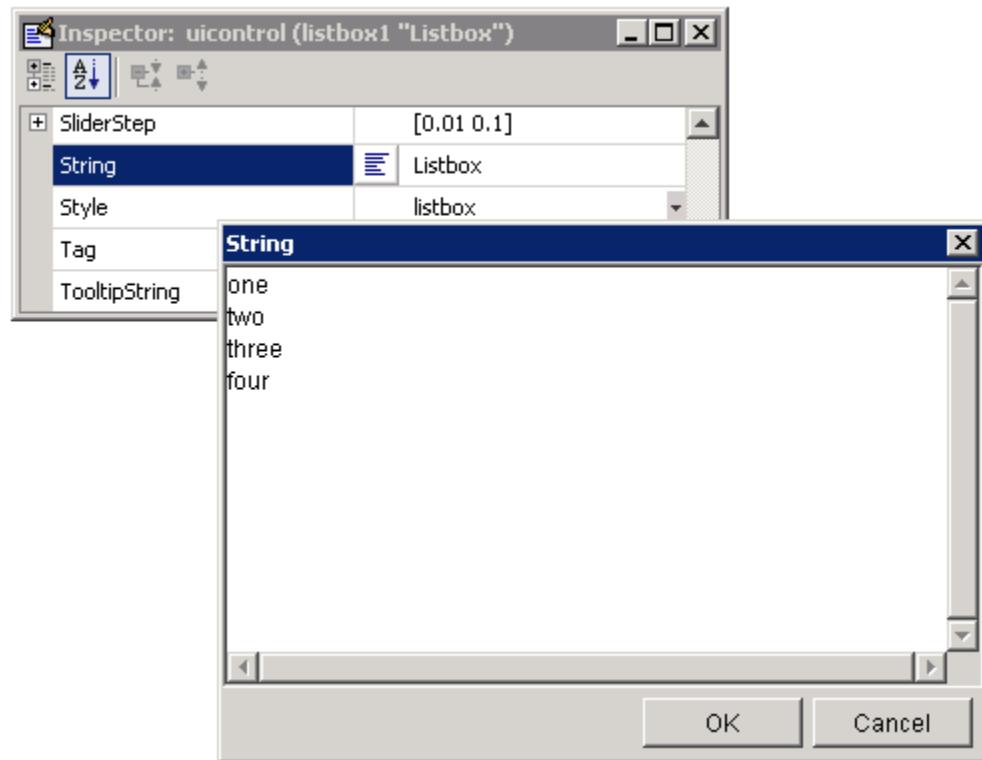
メモ ポップアップメニューは、ラベルを提供しません。ポップアップメニューをラベルするには、“スタティックテキスト”(p.6-48)コンポーネントを使用します。

リストボックス

次の図に示すように、項目 one、two、three、four をもつ、リストボックスを作成するには、以下を行います。



- String プロパティに表示したいリストを設定することによって、表示される項目のリストを指定します。プロパティインスペクターを使用して、リストを入力します。プロパティ名の右の  ボタンをクリックすると、エディターが開きます。



ラベルに “&” 文字を表示するには、文字列に 2 つの & 文字を使用します。
`remove`、`default`、`factory`（大文字と小文字の区別あり）は予約語です。これらの
 いずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ (\) を付
 けます。たとえば、`\$remove` は `remove` を与えます。

コンポーネントの幅が狭く、指定した文字列が 1 つ以上表示できない場合、
 MATLAB は省略記号を用いてこれらの文字列を短くします。

- Max および Min プロパティと共に Value プロパティを使用して、選択を指定します。
 - コンポーネントが作成されるときに 1 つの項目を選択するには、Value を選
 択したリスト項目のインデックスを示すスカラーに設定します。ここで 1 は
 リストの最初の項目に対応します。
 - コンポーネントが作成されたときに複数の項目を選択するには、Value を選
 択された項目のインデックスのベクトルに設定します。Value = [1, 3] は、
 次のような選択になります。



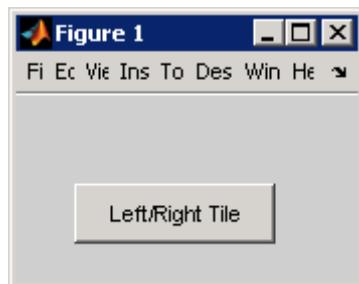
複数項目の選択を可能にするには、Max と Min プロパティの差が 1 よりも大きくなるようにこれらのプロパティを指定します。たとえば、Max = 2、Min = 0 とします。Max 既定値は 1 です。Min 既定値は 0 です。

- 最初の選択を行いたくない場合、複数の選択が可能になるように Max と Min プロパティを設定します。つまり、 $\text{Max} - \text{Min} > 1$ とし、Value プロパティを空行列 [] に設定します。
- リストボックスが小さく、すべてのリスト項目を表示できない場合、ListBoxTop プロパティを、コンポーネントが作成されたときにトップに表示したい項目のインデックスに設定できます。
- コンポーネントの位置やサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。

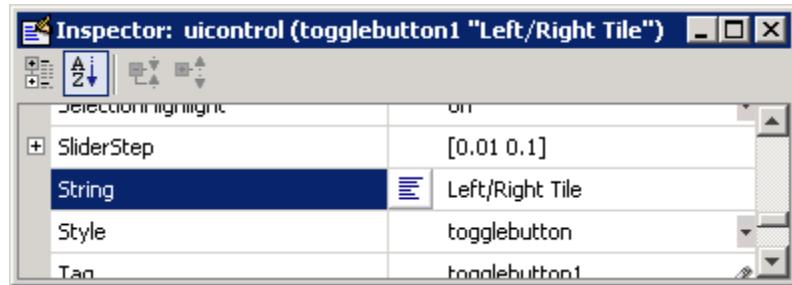
メモ リストボックスは、ラベルを提供しません。リストボックスにラベリングするには、“スタティック テキスト”(p.6-48) コンポーネントを使用します。

トグル ボタン

この図に示すように、ラベル Left/Right Tile が付いたトグル ボタンを作成するには、



- String プロパティに付けたいラベルを設定して、トグル ボタンのラベルを指定します。この場合は、Left/Right Tile とします。



ラベルに “&” 文字を表示するには、文字列に 2 つの & 文字を使用します。
remove、default、factory (大文字と小文字の区別あり) は予約語です。これらの
いずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ (\) を付
けます。たとえば、\\$remove は remove を与えます。

トグル ボタンのテキストは 1 行に限られます。2 行以上指定すると、最初の行の
みが示されます。指定された String を収められないような小さなトグル ボタンを
作成すると、MATLAB は省略記号を用いて文字列を短くします。



- トグル ボタンの Value プロパティを、ラジオ ボタンの Max プロパティの値（既定値は 1）に設定すると、選択されたボタンをもつラジオ ボタンを作成します。トグル ボタンを選択されていない（上がった）状態にするには、Value を Min（既定値は 0）に設定します。これに対応して、ユーザーが、トグル ボタンを選択すると、MATLAB は Value を Max に設定し、選択しないと、Value を Min に設定します。次の図は、押された状態のトグル ボタンを示します。



- コンポーネントの位置やサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動” (p.6-81) と “コンポーネントのサイズ変更” (p.6-84) を参照してください。

- トグル ボタンにイメージを追加するには、ボタンの CData プロパティに、トゥルーカラー イメージを定義する RGB 値の $m \times n \times 3$ 配列を割り当てます。これは、GUI M ファイルの opening 関数内のプログラムで、行わなければなりません。たとえば、配列 img は (rand で生成される) 0 から 1 までの乱数の値を使用して、 $16 \times 64 \times 3$ トゥルーカラー イメージを定義します。

```
img = rand(16, 64, 3);
set(handles.togglebutton1, 'CData', img);
```

togglebutton1 はトグル ボタンの Tag プロパティです。



メモ ラジオ ボタンとトグル ボタンの排他的な選択を処理するには、これらをボタン グループに置きます。詳細は、“ボタン グループ”(p.6-58)を参照してください。

パネルとボタン グループ定義

パネルとボタン グループは、GUI コンポーネントをグループにまとめて配置するコンテナです。パネルやボタン グループを移動すると、その子も共に移動し、パネルやボタン グループ上で相対的な位置を保ちます。

パネルとボタン グループを定義するには、いくつかのプロパティを設定する必要があります。このためには、以下のことを行います。

- 1 プロパティ インスペクターを使用して、該当するプロパティを修正します。[表示] メニューから [プロパティ インスペクター] を選択するか、あるいはプロパティ インスペクター アイコンをクリックして、プロパティ インスペクターを開きます。
- 2 レイアウト エリアで、定義しているコンポーネントを選択します。

メモ コンポーネントの詳細は、“利用可能なコンポーネント”(p.6-20)を参照してください。これらのコンポーネントのプログラミングの基本的な例として、“例:GUIDE GUI コンポーネントのプログラミング”(p.8-30)を参照してください。

以下のトピックスでは、パネルやボタン グループの一般に利用されるプロパティについて述べ、各コンポーネントに対する簡単な例を提供します。

- ・ “一般に利用するプロパティ”(p.6-56)
- ・ “パネル”(p.6-57)
- ・ “ボタン グループ”(p.6-58)

一般に利用するプロパティ

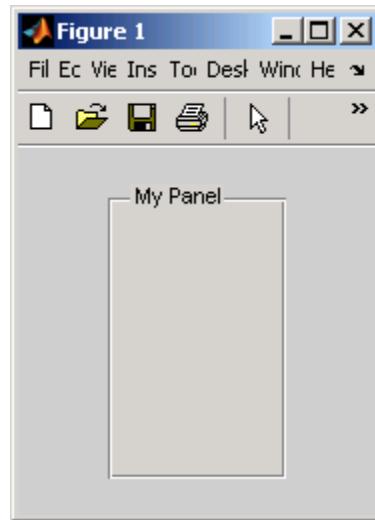
次の表では、パネルまたはボタン グループを記述するために必要となる、最も一般に利用されるプロパティを示します。

プロパティ	値	説明
Position	4要素ベクトル。[左端からの距離、下端からの距離、幅、高さ]。	コンポーネントのサイズとその親に対する相対的な位置。
Title	文字列	コンポーネントラベル。
TitlePosition	lefttop、center top、right top、left bottom、center bottom、right bottom。既定値は lefttop です。	パネルまたはボタン グループについて、タイトルの文字列の位置。
Units	characters、centimeters、inches、normalized、pixels、points。既定値は characters です。	Position プロパティ ベクトルを処理するために用いられる測定の単位。

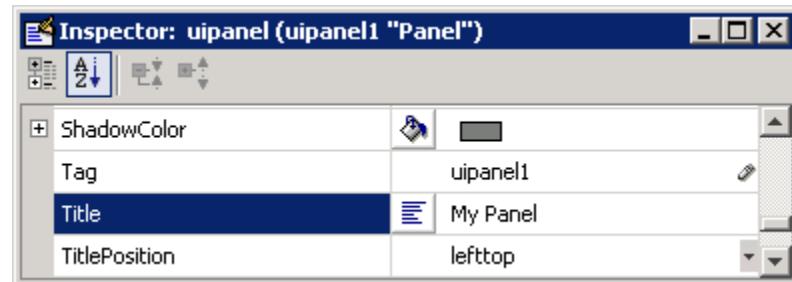
プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB リファレンス ドキュメンテーションの「Uipanel プロパティ」と「Uibuttongroup プロパティ」を参照してください。GUI の動作をコントロールするために必要なプロパティは、章 8、"GUIDE GUI のプログラミング" に説明します。

パネル

この図に示すように、タイトル My Panel が付いたパネルを作成するには、

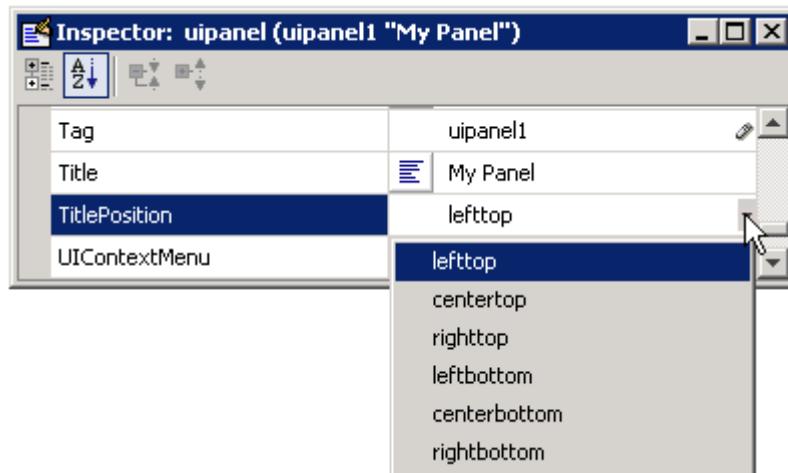


- Title プロパティに付けたい文字列を設定して、パネルのタイトルを指定します。この場合は、My Panel にします。



タイトルに “&” 文字を表示するには、文字列に 2 つの “&” 文字を使用します。remove、default、factory(大文字と小文字の区別あり)は予約語です。これらのいずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ(¥)を付けます。たとえば、¥remove は remove を与えます。

- ・ ポップアップメニューから利用できる TitlePosition プロパティの値を 1 つ選択してパネルのタイトルの位置を指定します。この場合は、lefttop にします。パネルの上、下、左、中央、右にタイトルを配置することも可能です。

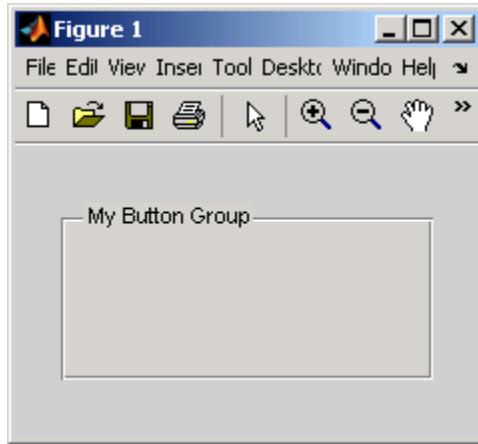


- ・ パネルの位置やサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。

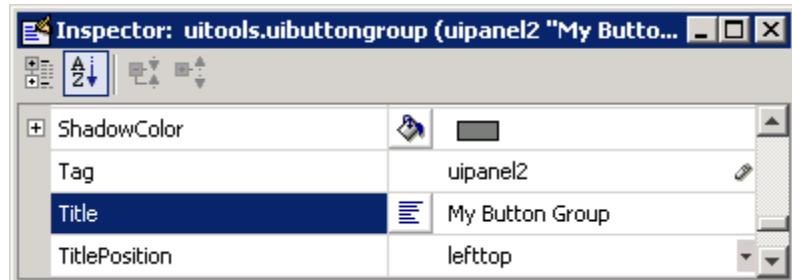
メモ さらにヒントや手法については、“パネルまたはボタン グループへのコンポーネントの追加”(p.6-34)と uipanel のリファレンスドキュメンテーションを参照してください。

ボタン グループ

この図に示すように、タイトル My Panel が付いたパネルを作成するには、

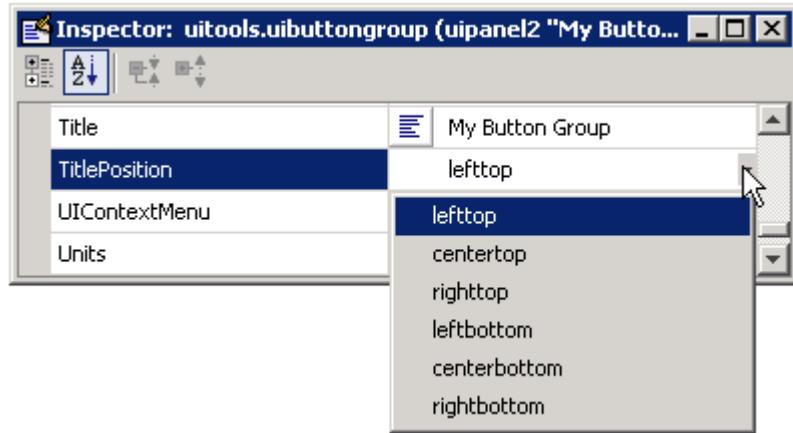


- ・ Title プロパティに付けたい文字列を設定して、パネルのタイトルを指定します。この場合は、My Panel にします。



タイトルに “&” 文字を表示するには、文字列に 2 つの “&” 文字を使用します。
`remove`、`default`、`factory` (大文字と小文字の区別あり) は予約語です。これらのいずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ (\) を付けてください。たとえば、`\remove` は `remove` を与えます。

- ・ ポップアップメニューから利用できる TitlePosition プロパティの値を 1 つ選択してボタン グループのタイトルの位置を指定します。この場合は、lefttop にします。ボタン グループの上、下、左、中央、右にタイトルを配置することも可能です。



- ボタン グループの位置やサイズを正確な値に設定したい場合、その Position プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。

メモ さらにヒントや手法については、“パネルまたはボタン グループへのコンポーネントの追加”(p.6-34) と uibuttongroup のリファレンスドキュメンテーションを参照してください。

座標軸を定義する

次のようなコマンド(plot、surf、line、bar、polar、pie、contour、meshなどのコマンド)を使用する場合、座標軸により、ユーザー GUI が、グラフィックス(たとえば、グラフィックスとイメージ)を表示することが可能になります。

座標軸を定義するには、いくつかのプロパティを設定する必要があります。このためには、以下のことを行います。

- 1 プロパティインスペクターを使用して、該当するプロパティを修正します。[表示] メニューから [プロパティインスペクター] を選択するか、あるいは [プロパティインスペクター] ボタンをクリックして、プロパティインスペクターを開きます。
- 2 レイアウトエリアで、定義しているコンポーネントを選択します。

メモ このコンポーネントの詳細は、“利用可能なコンポーネント”(p.6-20)を参照してください。

以下のトピックスでは、座標軸の一般的に使用されるプロパティについて述べ、簡単な例を提供します。

- ・ “一般に利用するプロパティ”(p.6-61)
- ・ “座標軸”(p.6-62)

一般に利用するプロパティ

次の表では、座標軸を説明するために必要となる、最も一般に利用されるプロパティを示します。

プロパティ	値	説明
NextPlot	add、replace、 replacechildren。既定値は replace です。	プロットでグラフィックスを追加するかどうか、グラフィックスを置き換えて Axes のプロパティを既定値にリセットする、あるいはグラフィックスのみを置き換える、を指定します。
位置	4要素ベクトル。[左端からの距離、下端からの距離、幅、高さ]。	コンポーネントのサイズとその親に対する相対的な位置。
単位	normalized、centimeters、characters、inches、pixels、points。既定値は normalized です。	位置ベクトルを処理するために用いられる測定の単位。

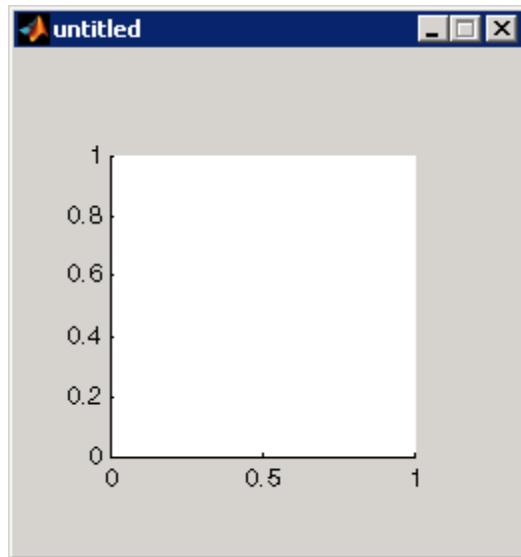
プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB ドキュメンテーションの「Axes プロパティ」を参照してください。GUI の動作をコントロールするために必要なプロパティは、章 8、"GUIDE GUI のプログラミング" に説明します。

Axes オブジェクトの詳細は、plot、surf、line、bar、polar、pie、contour、imagesc、mesh コマンドを参照してください。完全なリストとして、MATLAB 関数リファレンス ドキュメンテーションの“Function Reference”を参照してください。

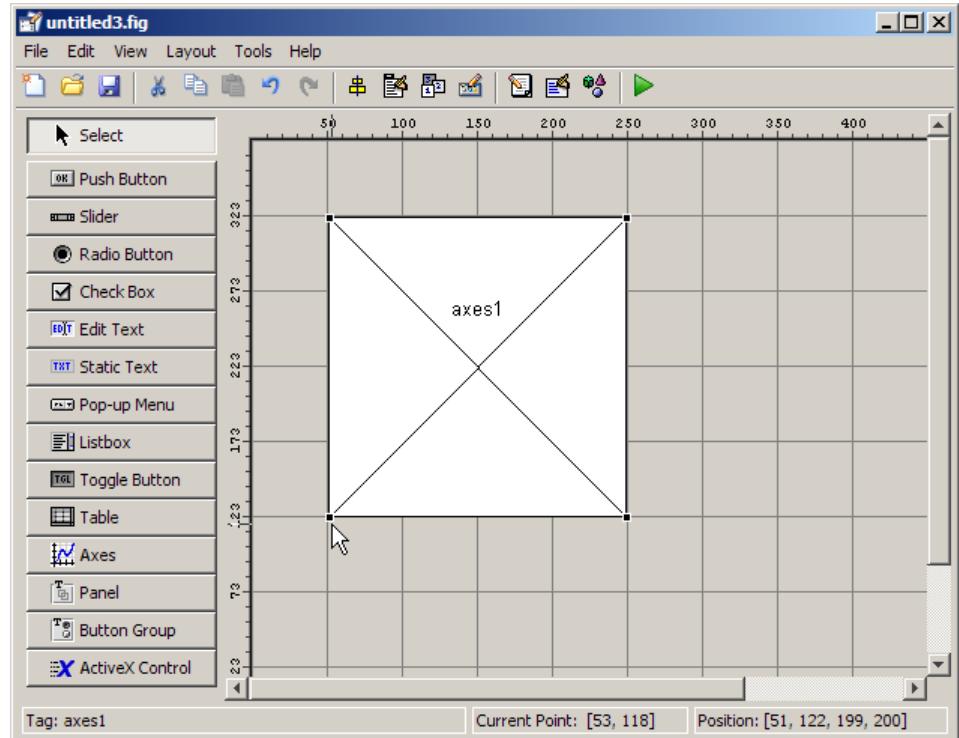
これらのグラフ作成関数の多くは、その NextPlot プロパティの設定に従い、既定で Axes プロパティをリセットします。これは、Axes の範囲を設定し直したり、Axes のコンテキストメニュー やコールバックを削除するなど、GUI の望ましくない動作を起こします。NextPlot Axes プロパティの詳細と設定は、次の節とさらに「プログラミングで作成する GUI」の節の“座標軸の追加”(p.11-37) を参照してください。

座標軸

この図に示すように、座標軸を作成するには、



- ・ 目盛りをレイアウト エディターに現れるボックスの外側に置くことができます。上述の座標軸は、レイアウト エディターで次のように見えます。座標軸の左と下に目盛りのためのスペースが配置されます。座標軸に描かれる関数は、目盛を適切に更新します。



- 座標軸コンポーネントにラベルするには、GUI M ファイルの関数 `title`、`xlabel`、`ylabel`、`zlabel`、`text` を使用してください。たとえば、

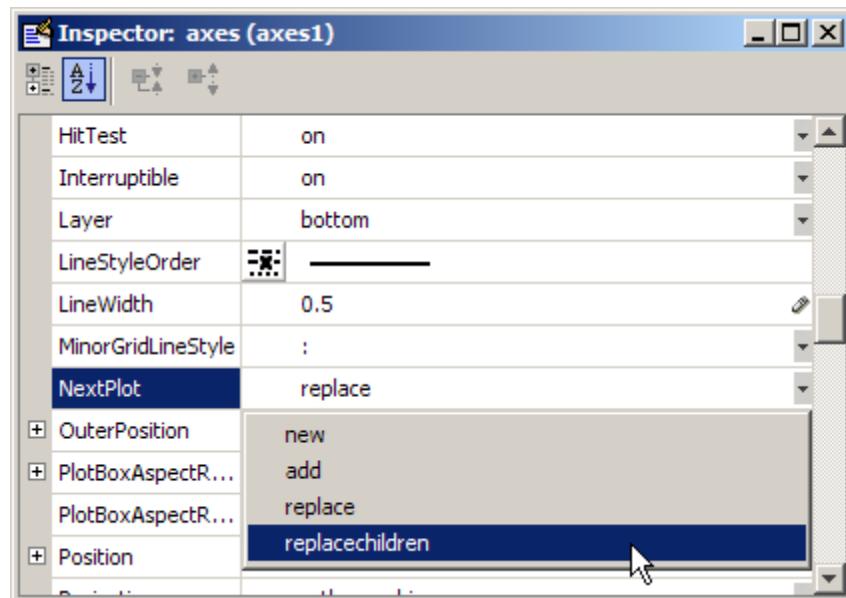
```
xlh = (axes_handle, 'Years')
```

は、X 軸に Years とラベルします。X 軸のラベルのハンドルは `xlh` です。座標軸ハンドルの決定についての詳細は、“コールバック構文と引数”(p.8-15)を参照してください。

`remove`、`default`、`factory`(大文字と小文字の区別あり)は予約語です。これらのいずれかコンポーネントテキストで使用するには、文字列の先頭にバックスラッシュ(\)を付けます。たとえば、`\$remove` は `remove` を与えます。

- 座標軸の位置やサイズを正確な値に設定したい場合、その `Position` プロパティを変更します。詳細は、“コンポーネントの位置決めと移動”(p.6-81)と“コンポーネントのサイズ変更”(p.6-84)を参照してください。

- Axes プロパティをカスタマイズすると、それらのいくつか（たとえば、コールバック、フォントの特性、Axis の範囲と目盛など）は、NextPlot プロパティが 'replace' の既定値をもつときに、グラフを Axes に書き込むたびに、既定値にリセットされます。カスタマイズされたプロパティを所望の状態に保つには、ここに示すように、プロパティインスペクターで、NextPlot を 'replacechildren' に設定します。



テーブルを定義する

テーブルを用いると、2 次元の表にデータを表示できます。プロパティインスペクターを使用して、オブジェクトのプロパティ値の取得や設定を行うことができます。

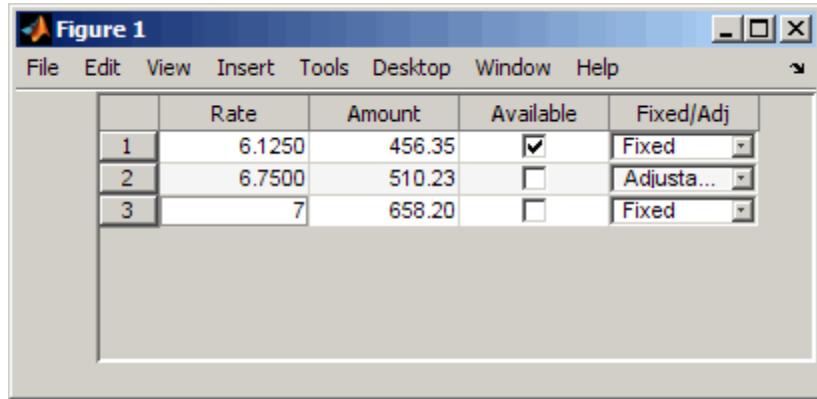
一般に利用するプロパティ

テーブルコンポーネントの最も一般に使用されるプロパティを、下記の表にリストします。これらは、テーブルプロパティエディターに現れる順でグループ化されています。テーブルプロパティの詳細は、uitable コマンドのドキュメンテーションを参照してください。

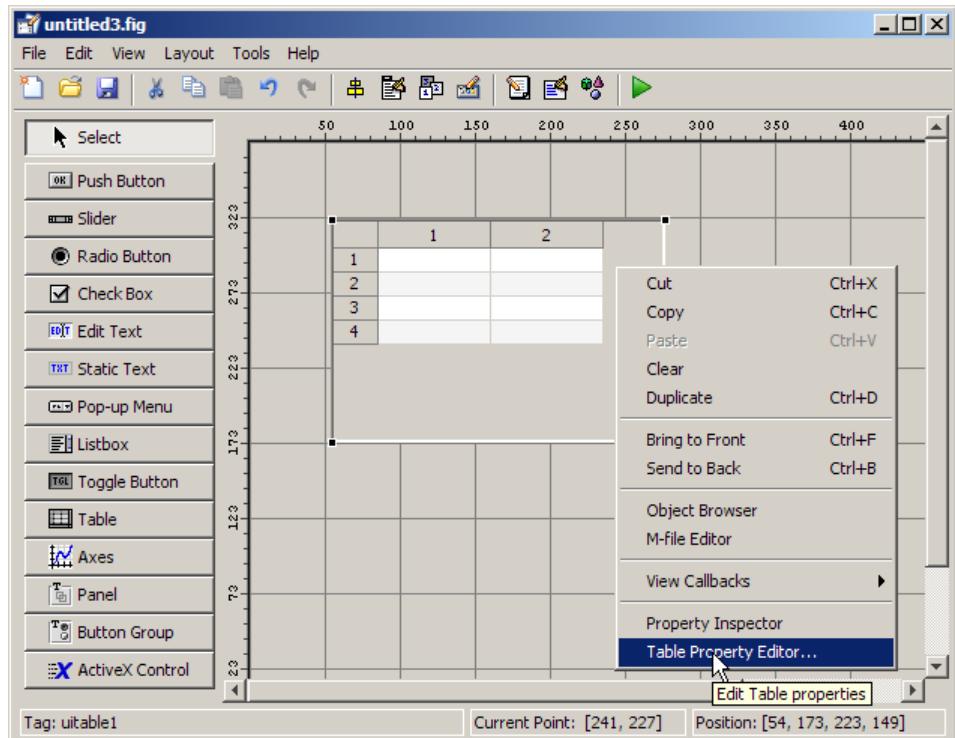
グループ	プロパティ	値	説明
列	ColumnName	文字列 {’番号’} 空行列 ([]) の $1 \times n$ セル配列	列ヘッダー ラベル
	ColumnFormat	文字列のセル配列	列の表示と編集可能であるかを決める
	ColumnWidth	$1 \times n$ セル配列または ’ auto’	各列のピクセル単位の幅、個々の列幅は ’ auto’ にも設定できます
	ColumnEditable	論理値 $1 \times n$ 行列 スカラー論理値 空行列 ([])	列内のデータを編集可能として決める
行	RowName	$1 \times n$ の文字列のセル配列	行ヘッダー ラベルの名前
色	BackgroundColor	$n \times 3$ の RGB triple 行列	セルの背景色
	RowStriping	{on} off	テーブルの行のストライプ配色
データ	Data	数値、論理値、文字データの行列またはセル配列	テーブル データ

テーブルの作成

図に示すように、GUIDE のテーブルを用いて GUI を作成するには、以下を行います。



レイアウト エディターにテーブル アイコンをドラッグして、テーブル内で右クリックします。ポップアップ コンテキストメニューから[テーブル プロパティ エディター]を選択します。テーブルだけを選択する場合、[ツール] メニューから [テーブル プロパティ エディター] を選択することもできます。



テーブル プロパティ エディターの利用。このようにして開くと、テーブル プロパティ エディターは [列] ペインを表示します。そのテーブル プロパティ エディター アイコン の 1 つをクリックすることでプロパティ インスペクターからクリックしたプロパティに適切なペインを開くための [テーブル プロパティ エディター] を開きます。

テーブル プロパティ エディターの左側のリストの項目をクリックすると、ペインのコンテンツを右側に変更します。項目を使用して、テーブルの [列]、[行]、[データ]、[色] オプションを指定して、コントロールをアクティブにします。

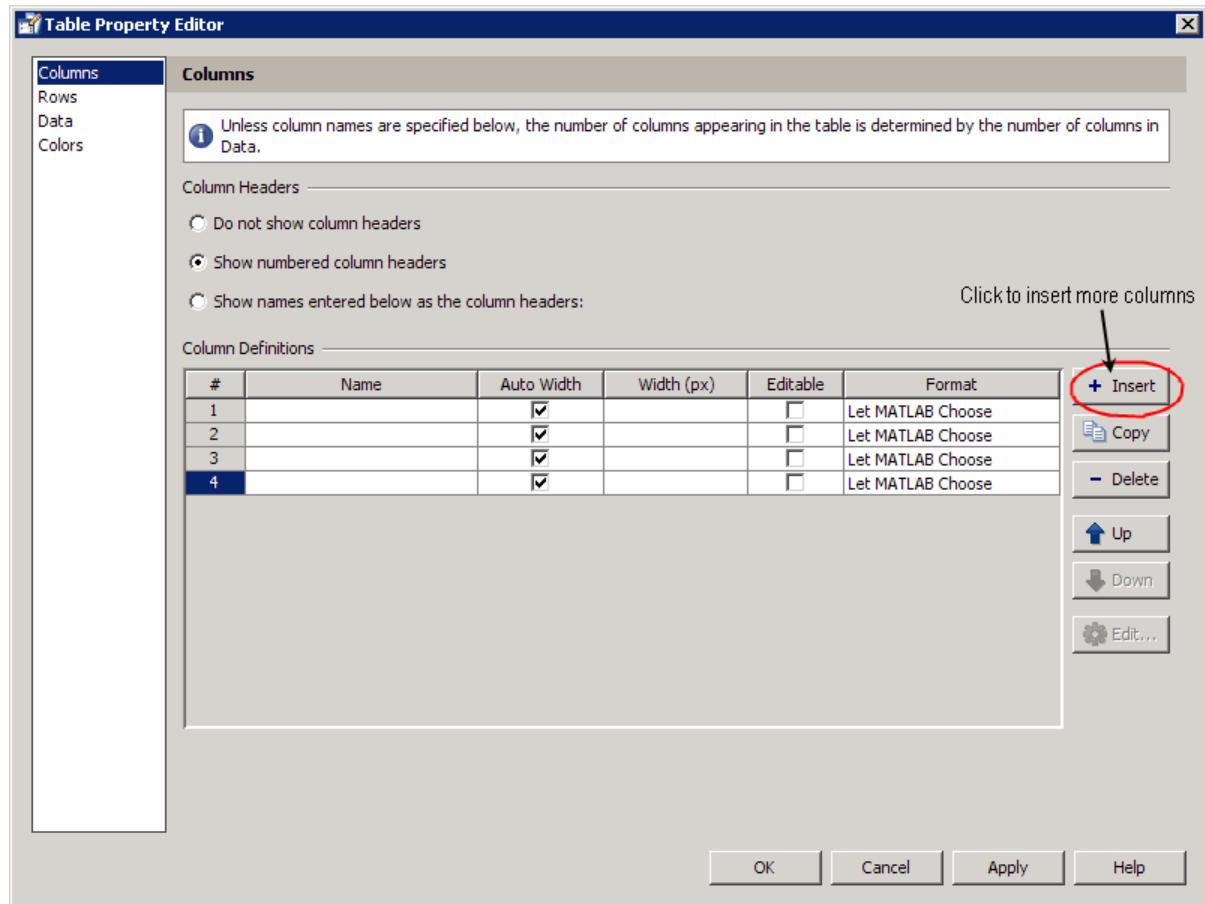
[列] ペインと [行] ペインはそれぞれ、列毎または行毎に、名前を入力しプロパティを設定できるデータ入力エリアがあります。一度に 1 行または 1 列のみ編集できます。これらのペインは、編集とナビゲートのための 5 つのボタンの垂直方向のグループを含みます。

ボタン	目的	アクセラレータ キー	
		Windows	Macintosh
[挿入]	現在の項目の下に、新規の列または行定義の項目を挿入します。	Insert	Insert
[削除]	現在の列または行の定義の項目(undo でない)を削除します。	Ctrl+D	Cmd+D
[コピー]	その下の新しい行に選択された項目のコピーを挿入します。	Ctrl+P	Cmd+P
[上]	選択された項目を 1 行上に移動します。	Ctrl+uparrow	Cmd+uparrow
[下]	選択された項目を 1 行下に移動します。	Ctrl+downarrow	Cmd+downarrow

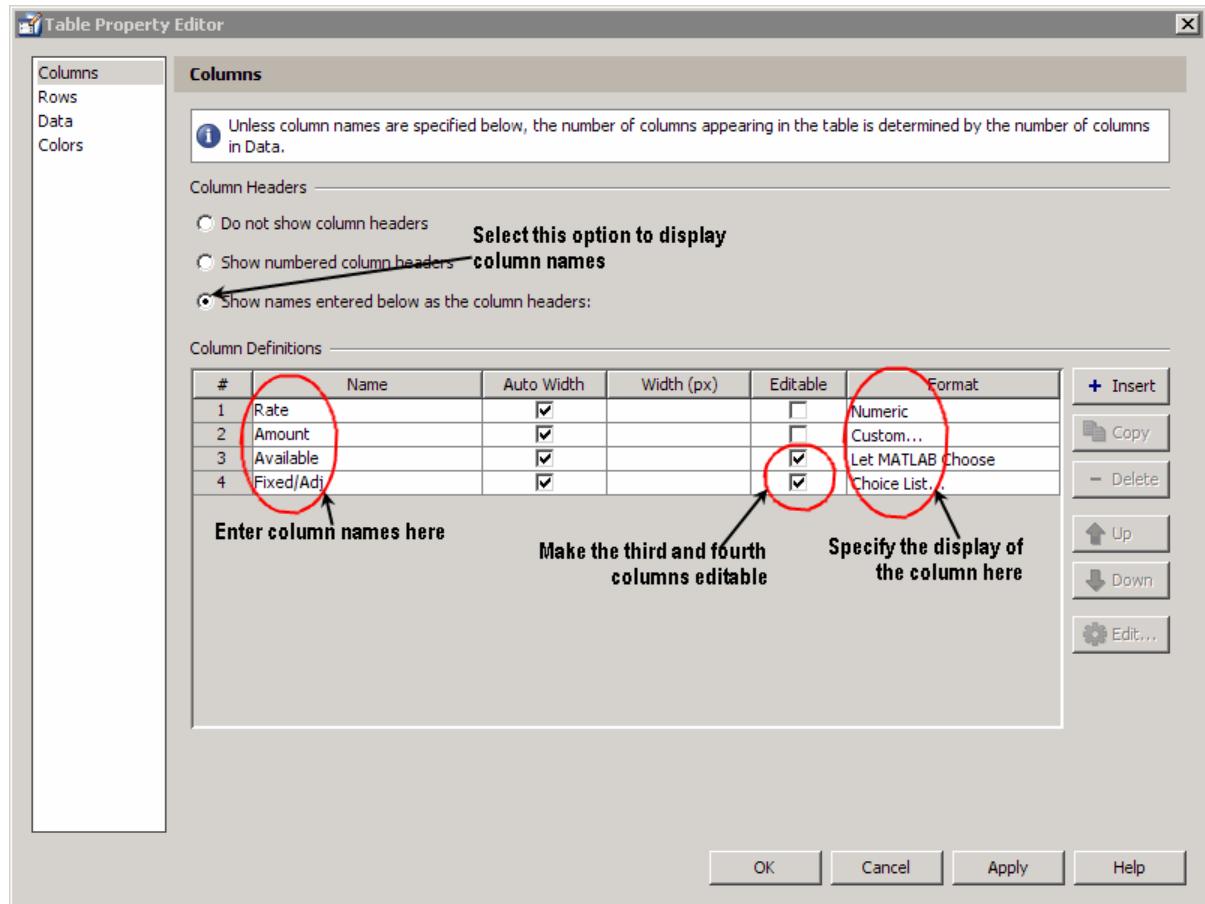
カーソルが日付入力エリアにあるときに限りキーボードと同じ動作をします。上記にリストされたものの他に、Ctrl+T または Cmd+T を入力すると、編集のためにカーソルを含むフィールド全体を選択します(フィールドがテキストを含む場合)。

テーブル プロパティ エディターでの変更をテーブルに保存するには、[OK] をクリックします。あるいは、[適用] をクリックすると、変更を送り、テーブル プロパティ エディターを用いて続行します。

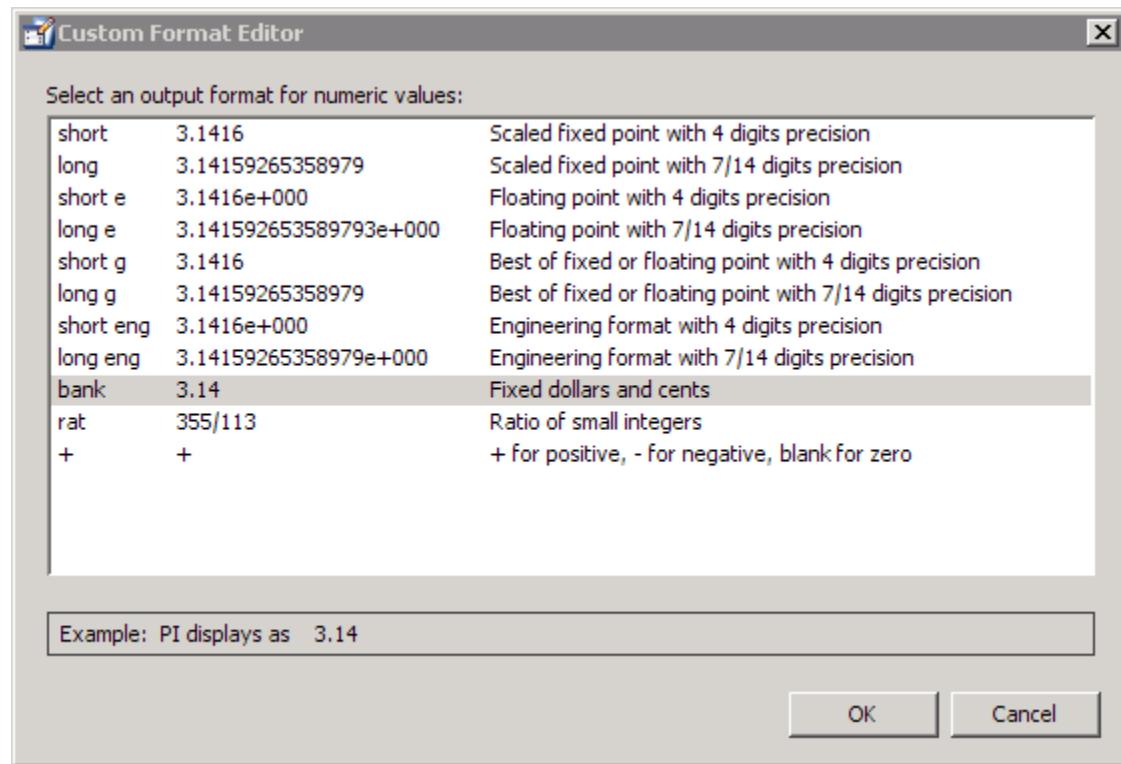
列のプロパティの設定. [挿入] をクリックして、さらに 2 列を追加します。



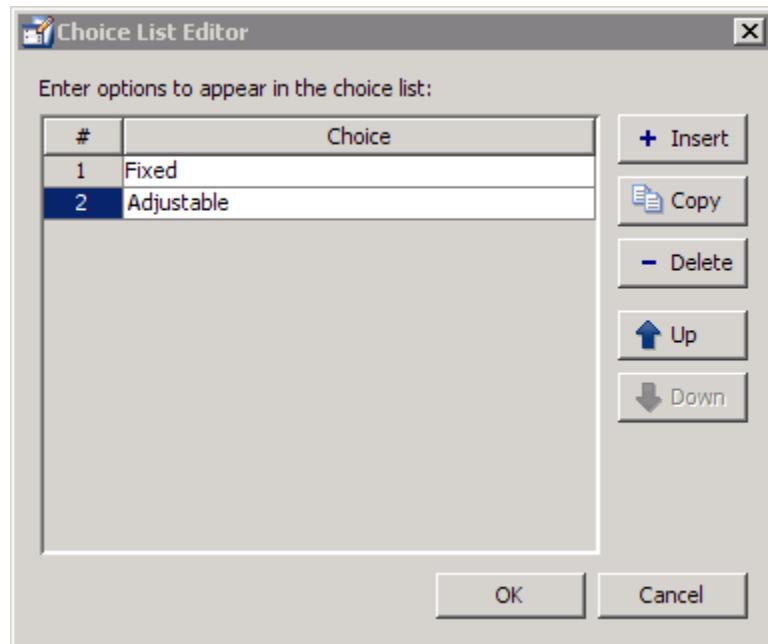
[列ヘッダーとして下記に入力された名前を表示] を選択し、[名前] グループに Rate、Amount、Available、Fixed/Adj を入力して ColumnName を設定します。Available および Fixed/Adj 列に対し、ColumnEditable プロパティを on に設定します。最後に、4 列に対して ColumnFormat を設定します。



Rate 列に対して、[数値] を選択します。Amount 列に対して、[カスタム] を選択して、カスタム形式エディターで bank を選択します。

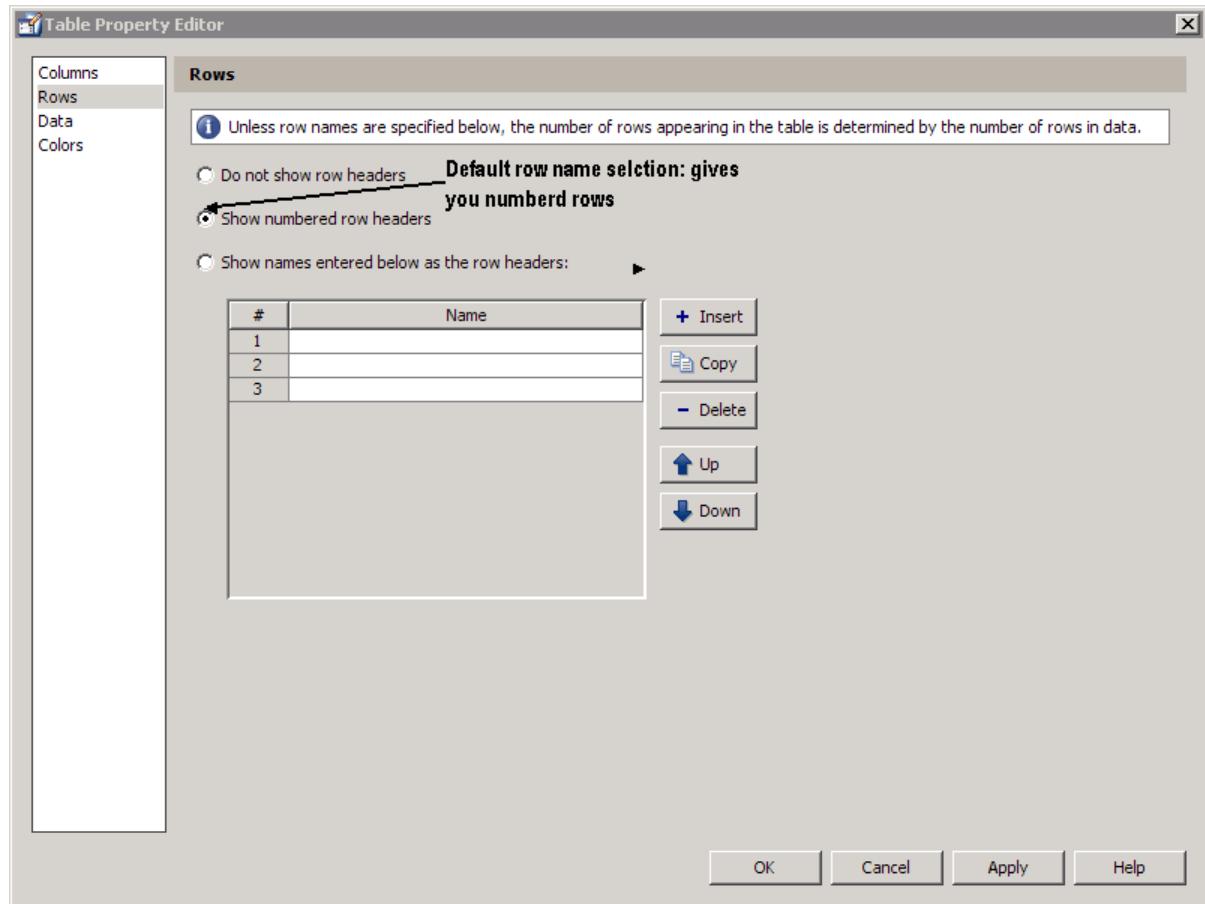


Available 列を既定値にします。これにより、MATLAB はテーブルの Data プロパティの値をもとに選択できます。Fixed/Adj 列に対して、[リストの選択] を選択して、ポップアップメニューを作成します。[リストエディターの選択] で [挿入] をクリックして、第 2 の選択を追加し、Fixed と Adjustable を 2 つの選択として入力します。



メモ ユーザーが選択リストから項目を選択するためには、リストの選択形式が設定されている列の ColumnEditable プロパティは 'true' に設定されなければなりません。ポップアップ コントロールは、列が編集可能である場合のみ表示されます。

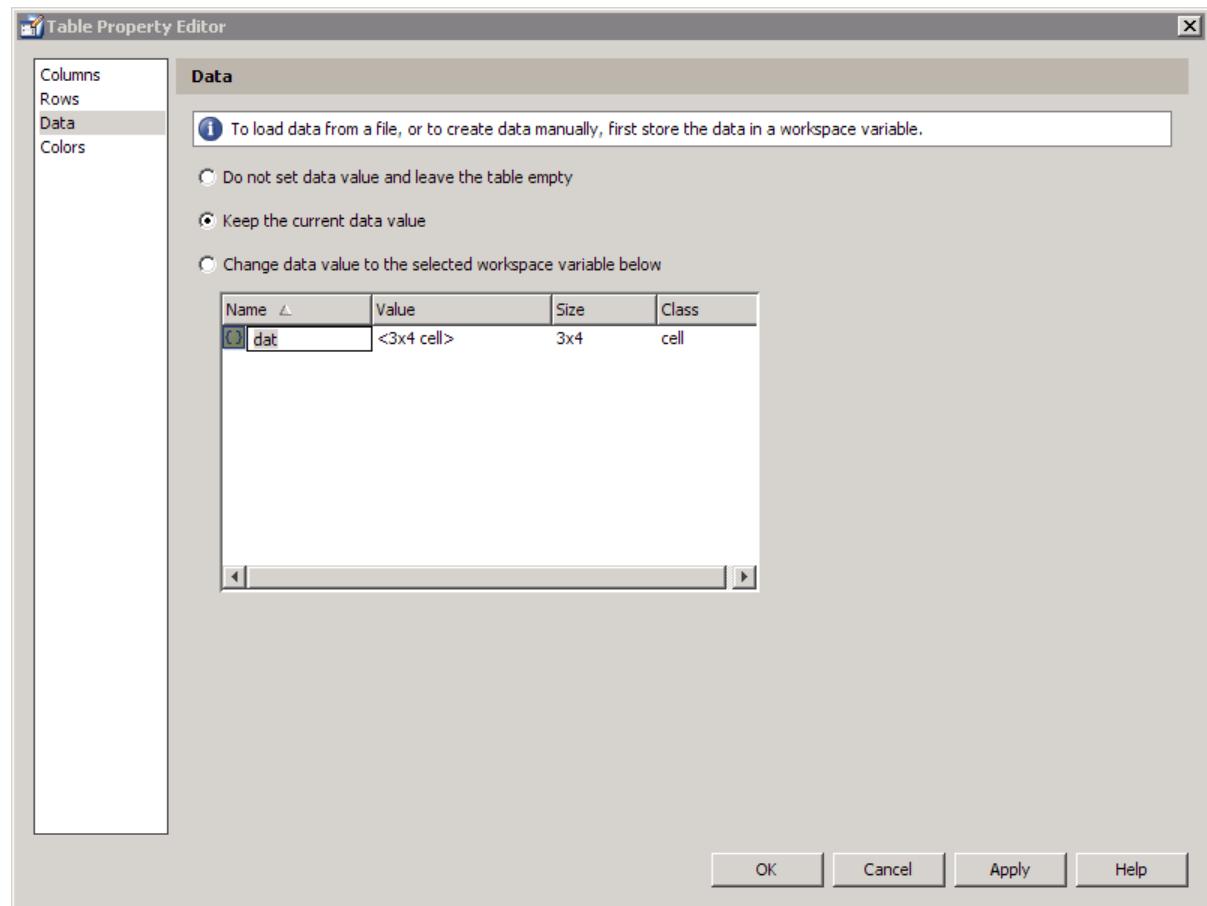
行のプロパティの設定. [行] タブにおいて、既定値の RowName、[番号付き行ヘッダー表示] にします。



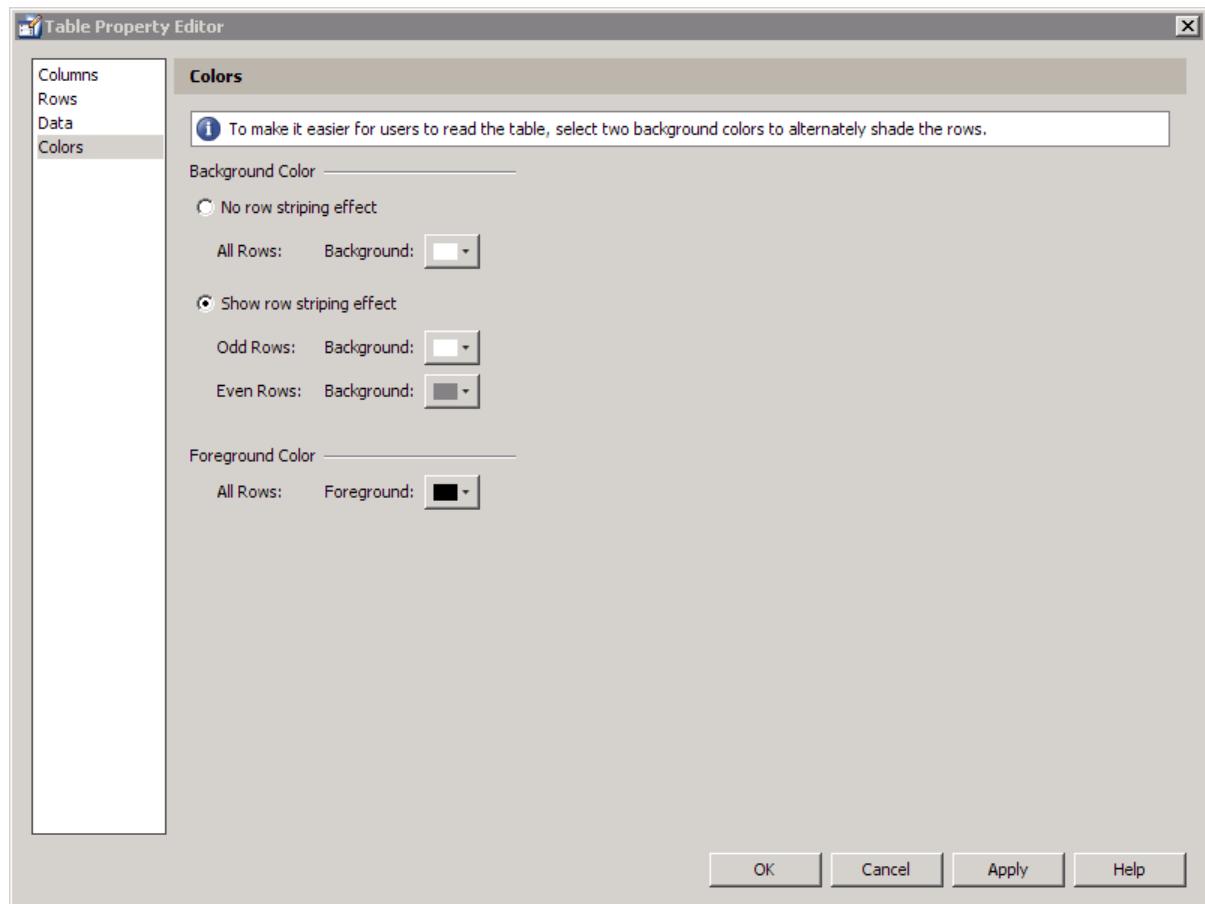
データのプロパティの設定. テーブルに置く Data の値を指定します。GUIDE で Data を指定する前に、MATLAB コマンド ウィンドウで Data を作成する必要があります。たとえば、次のように入力します。

```
dat = {6.125, 456.3457, true, 'Fixed'; ...
        6.75, 510.2342, false, 'Adjustable'; ...
        7,      658.2,    false, 'Fixed'};
```

テーブル プロパティ エディターで、ユーザーが定義したデータを選択し、[データ値を選択した以下のワークスペース変数に変更] を選択します。



色のプロパティの設定. [色] タブで、テーブルに対して `BackgroundColor` と `RowStriping` を指定します。

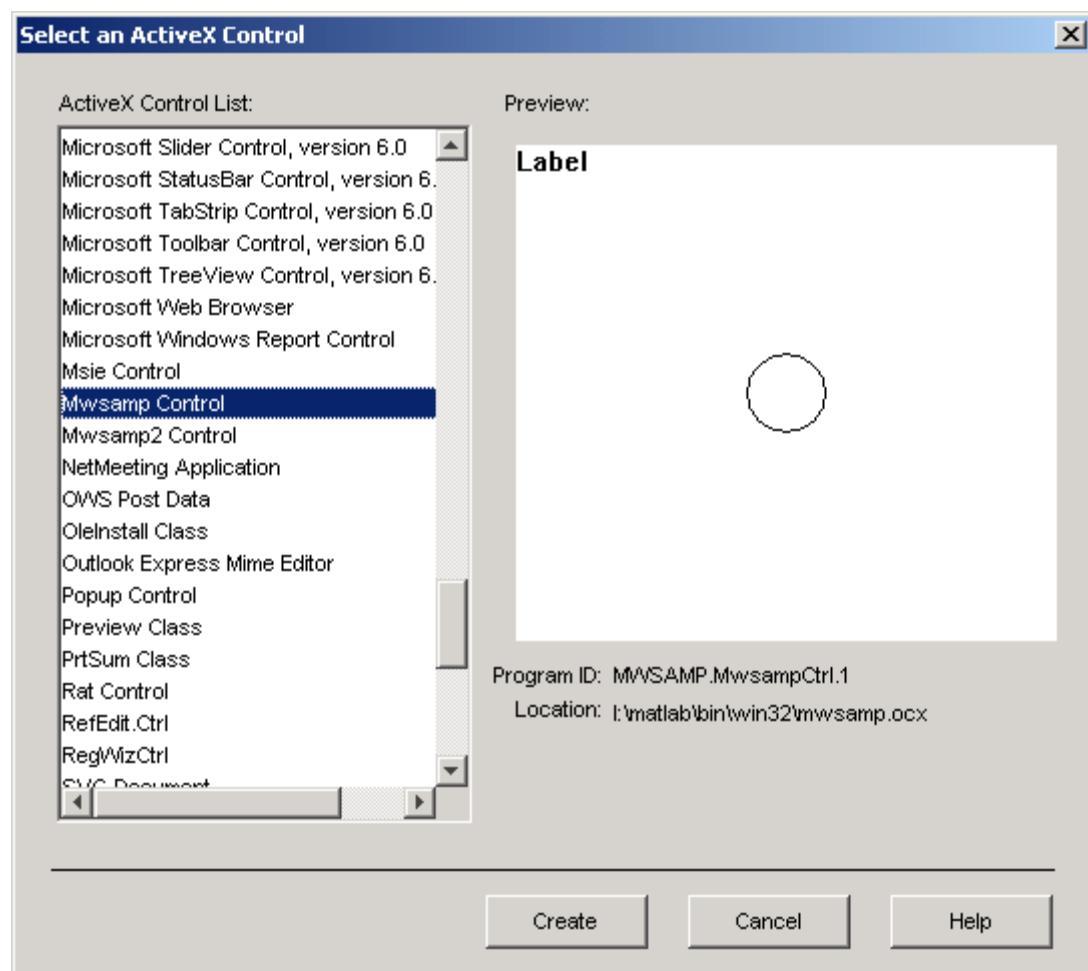


プロパティ インスペクターによって、テーブルのその他の `uitable` プロパティを変更できます。

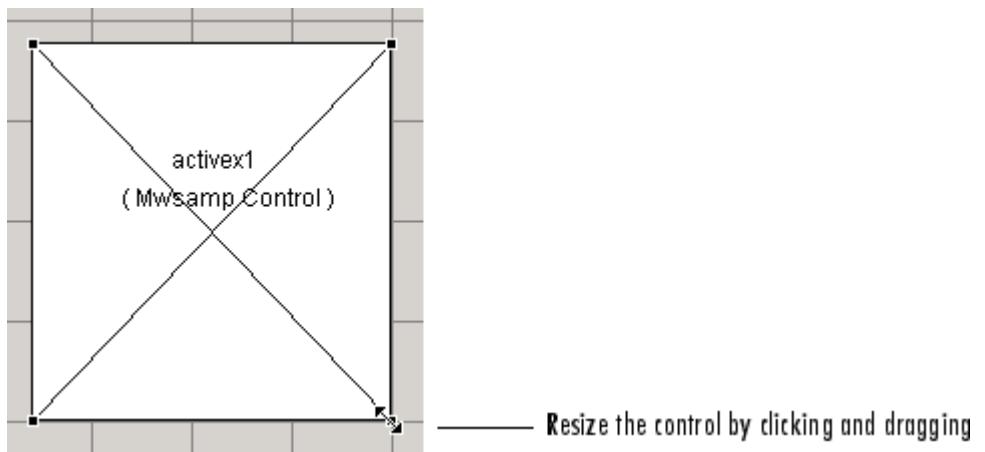
ActiveX コントロールの追加

コンポーネント パレットからレイアウト エリアに ActiveX コンポーネントをドラッグする場合、GUIDE は、ユーザー システム上に記録された ActiveX コントロールすべてをリストするダイアログ ボックスを開きます。

メモ MATLAB がユーザーのコンピューターにインストールされていない場合(たとえば、MATLAB をネットワーク上で実行している場合)、この例で述べる ActiveX コントロールが見つからないかもしれません。コントロールを登録するには、MATLAB「外部インターフェイス」ドキュメンテーションの“Registering Controls and Servers”を参照してください。



- 1 ActiveX コントロールを選択します。右のパネルは、選択したコントロールのレビューを示します。
- 2 [作成] をクリックします。コントロールがレイアウト エディターの小さなボックスとして現われます。
- 3 プレビュー ペインに示す四角のサイズに対するコントロールをサイズ変更します。これは、次の図に示すように、コントロールの隅をクリック、ドラッグして行うことができます。



ActiveX コントロールを選択すると、コンテキストメニューから、[ActiveX Property Editor] を右クリックして選択するか、または [ツール] メニューをクリックしてそれを選択することで、ActiveX プロパティエディターを開くことができます。

メモ [ActiveX プロパティエディター] の外見と何が含まれるかは、個々の ActiveX オブジェクトの作者が作成してオブジェクトの GUI に格納したものについて、ユーザーがどのようにコントロールするかで決まります。場合によっては、このメニュー項目を選択すると、コントロールなしの GUI を表示するか、あるいは GUI を全く表示しません。

サンプルの ActiveX コントロールと例については、“ActiveX コントロール”(p.8-48)を参照してください。

レイアウトエリア内のコンポーネントの取り扱い

このトピックは、レイアウトエリアでのコンポーネントの選択、移動、コピー、削除などについての基本的な情報を提供します。

- ・ “コンポーネントの選択” (p.6-78)
- ・ “コンポーネントのコピー、カット、クリア” (p.6-79)
- ・ “コンポーネントのペーストと複製” (p.6-79)
- ・ “前後位置” (p.6-80)

その他のトピックスとして以下のものがあります。

- ・ “コンポーネントの位置決めと移動” (p.6-81)
- ・ “コンポーネントのサイズ変更” (p.6-84)
- ・ “コンポーネントの整列” (p.6-87)
- ・ “タブの順序の設定” (p.6-96)

コンポーネントの選択

以下の方法でレイアウトエリアにコンポーネントを選択することができます。

- ・ 選択するために 1 つのコンポーネントをクリックします。
- ・ Figure のすべての子オブジェクトを選択するには、Ctrl+A を押します。これにより、パネルまたはボタン グループの子オブジェクトであるコンポーネントは選択されません。
- ・ コンポーネントを選択するには、カーソルをクリックしてドラッグし四角で囲みます。この選択範囲がパネルまたはボタン グループを囲む場合、パネルまたはボタン グループのみが選択され、その子は選択されません。選択範囲がパネルまたはボタン グループの一部を囲む場合、そのパネルまたはボタン グループの子オブジェクトである、四角内のコンポーネントのみが選択されます。
- ・ Shift と Ctrl キーを使用することで複数のコンポーネントを選択できます。

コンポーネントが親の境界の外側に位置する場合があります。そのようなコンポーネントはレイアウトエディターで表示されませんが、それを囲む長方形をドラッグするか、オブジェクトブラウザーで選択することによって選択されます。そのようなコンポーネントは、アクティブな GUI で表示されます。

オブジェクト ブラウザについての詳細は、“オブジェクト階層の表示”(p.6-134)を参照してください。

メモ 同じ親をもつ場合に限り、複数のコンポーネントを選択できます。Figure、パネル、ボタン グループの子オブジェクトを決定するために、オブジェクト ブラウザを使用します。

コンポーネントのコピー、カット、クリア

各コンポーネントをコピー、カット、クリアするには、既存のメニュー や ポップアップ メニュー、ツールバー アイコン、キーボードのキー、ショートカット キーを使用します。

コピー. コピーでは、クリップボード上に選択した各コンポーネントをコピーします。パネルやボタン グループのコピーは、その子を含みます。

カット. カットは、クリップボード上の選択した各コンポーネントをコピーし、レイアウト エリアからそれらを削除します。パネルやボタン グループをカットする場合、そのすべての子もカットします。

クリア. クリアでは、レイアウト エリアから選択したコンポーネントを削除します。クリップボード上に選択したコンポーネントをコピーしません。パネルやボタン グループをクリアする場合、そのすべての子もクリアします。

コンポーネントのペーストと複製

ペースト. コンポーネントを貼り付けるには、既存のメニュー や ポップアップ メニュー、ツールバー アイコン、ショートカット キーを使用します。GUIDE は、最後にマウス クリックのあった位置にクリップボードの内容をペーストします。マウス クリックの位置が、左上隅になります。

連続してペーストすると、各コピーを最後のコピーの右下に置きます。

複製. 複製したい 1 つまたは複数のコンポーネントを選択して、以下のいずれかを行います。

- ・ 上記のように選択したコンポーネントをコピーしペーストします。

- ・ [編集] メニューまたはポップアップメニューから [複製] を選択します。[複製] は、オリジナルの右下にコピーを置きます。
- ・ コンポーネントを右クリックし希望する位置までドラッグします。コンポーネントを置く際のカーソルの位置が、選択したコンポーネントすべての親を決めます。“パネルまたはボタン グループへのコンポーネントの追加”(p.6-34)の記述を参照してください。

前後位置

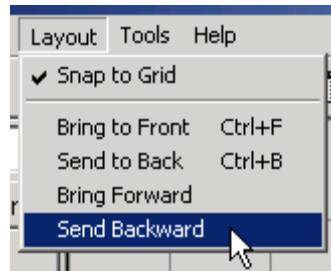
MATLAB Figure は、違う種類のコンポーネントの前後位置をコントロールする別々のスタックを保持します。

- ・ ボタン、スライダー、ポップアップメニューなどのユーザー インターフェイス コントロール
- ・ パネル、ボタン グループ、座標軸
- ・ ActiveX コントロール

コンポーネント同士が同じスタックにある場合に限り、重なっているコンポーネントの前後位置をコントロールすることができます。別のスタックにあり重なっているコンポーネントに対しては、次のようにになります。

- ・ ユーザー インターフェイス コントロールは、常に、パネル、ボタン グループ、座標軸、および、それらが重なっている ActiveX コントロールの一番上に現れます。
- ・ ActiveX コントロールは、重なったものの最前面になります。

レイアウト エディターは、前後の位置関係をコントロールできる 4 つの操作を提供します。次の図に示す [レイアウト] メニューからすべて利用できます。



- ・ [最前面へ移動]—非選択状態のオブジェクトの前面に、選択した(1つあるいは複数の)オブジェクトを移動します。(右クリックで現れるコンテキストメニュー、[レイアウト]メニュー、あるいは Ctrl+F ショートカットから利用できます)。
- ・ [最背面へ移動]—非選択状態のオブジェクトの後ろに、選択した(1つあるいは複数の)オブジェクトを移動します。(右クリックで現れるコンテキストメニュー、[レイアウト]メニュー、あるいは Ctrl+B ショートカットから利用できます)。
- ・ [前面へ移動]—選択したオブジェクトを、1つ前に移動します。すなわち、その直前のオブジェクトの前へ移動し、それに重なるすべてのオブジェクトの最も前への移動ではありません([レイアウト]メニューから利用できます)。
- ・ [背面へ移動]—選択したオブジェクトを、1つ後ろに移動します。すなわち、その直後のオブジェクトの後ろへ移動し、それに重なるすべてのオブジェクトの最も後ろへの移動ではありません([レイアウト]メニューから利用できます)。

メモ コンポーネントの前後位置を変更すると、タブの順序も変更されます。詳細は、“タブの順序の設定”(p.6-96)を参照してください。

コンポーネントの位置決めと移動

以下のいずれかの方法で、コンポーネントの位置決めと移動を行うことができます。

- ・ “座標表示の利用”(p.6-81)
- ・ “コンポーネントのドラッグ”(p.6-82)
- ・ “コンポーネントの移動に矢印キーを利用する”(p.6-83)
- ・ “コンポーネントの Position プロパティの設定”(p.6-83)

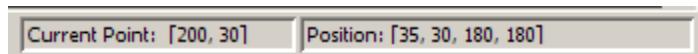
次は、重要な他のトピックスです。

- ・ “コンポーネントの整列”(p.6-87)

座標表示の利用

座標表示は、コンポーネントとマウス ポインタの位置を示します。これらの表示を用いて、手動で、コンポーネントの位置を決めて整列させます。レイアウトエディターの右下隅の座標表示は、選択されたコンポーネント(1つまたは複数)の位置を [x|left y|bottom width height] と表示します。これらの値は、コンポーネントに対して選択する座標の単位に関わらず、ピクセル単位で表示されます。

コンポーネントをドラッグまたはサイズ変更すると、それに応じて表示が更新されます。コンポーネントの位置表示の左の表示は、現在のマウスの位置をピクセル単位で表します。以下の表示の例は、位置 [35, 30, 180, 180] の選択されたコンポーネントを示します。x=35 と y=30 の左下隅に、180×180 ピクセルのオブジェクトがあり、マウスを位置 [200, 30] に置きます。



複数のオブジェクトを選択すると、[位置] 表示は、各オブジェクトが同じ x、y、width、height をもつ場合は、値が表示されます。その他の場合は、' MULTI' を表示します。たとえば、2 つのチェック ボックスを選択すると、一方の位置は [250, 140, 76, 20] ピクセルになり、他方の位置は [250, 190, 68, 20] ピクセルになります。[位置] の表示は、[250, MULTI, MULTI, 20] となります。

コンポーネントのドラッグ

移動したい 1 つまたは複数のコンポーネントを選択してから、適切な位置までドラッグ アンド ドロップします。コンポーネントを Figure からパネルまたはボタン グループに移動させることができます。パネルまたはボタン グループから、Figure あるいは他のパネルやボタン グループに、コンポーネントを移動することができます。

コンポーネントを置く際のカーソルの位置が、選択したコンポーネントすべての親も決めます。“パネルまたはボタン グループへのコンポーネントの追加”(p.6-34)の記述を参照してください。

1 つまたは複数の選択したコンポーネントがその親の境界の外に位置する場合があります。そのようなコンポーネントはレイアウトエディターで表示されませんが、それを囲む長方形をドラッグするか、オブジェクトブラウザーで選択することによって選択されます。そのようなコンポーネントは、アクティブな GUI で表示されます。

オブジェクトブラウザーについての詳細は、“オブジェクト階層の表示”(p.6-134)を参照してください。

メモ 複数のコンポーネントを選択するためには、それらが同じ親をもたなければなりません。つまり、それらが 同じ Figure、パネル、または、ボタン グループに含まれる必要があります。

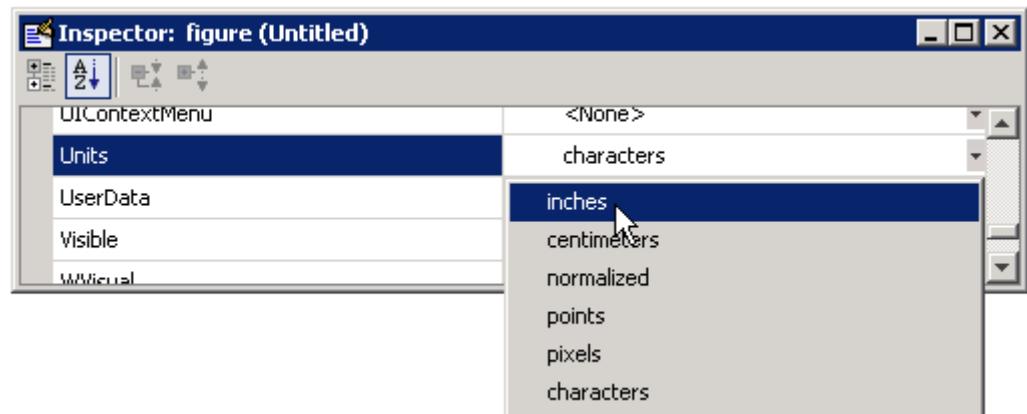
コンポーネントの移動に矢印キーを利用する

移動したい 1 つまたは複数のコンポーネントを選択してから、コンポーネントが適切な位置に移動するまで矢印キーを押し続けます。ただし、この方法で、各コンポーネントを他のオブジェクトの境界上まで移動したとしても、移動する前に属していた Figure、パネル、あるいはボタン グループの子オブジェクトのままであることに注意してください。

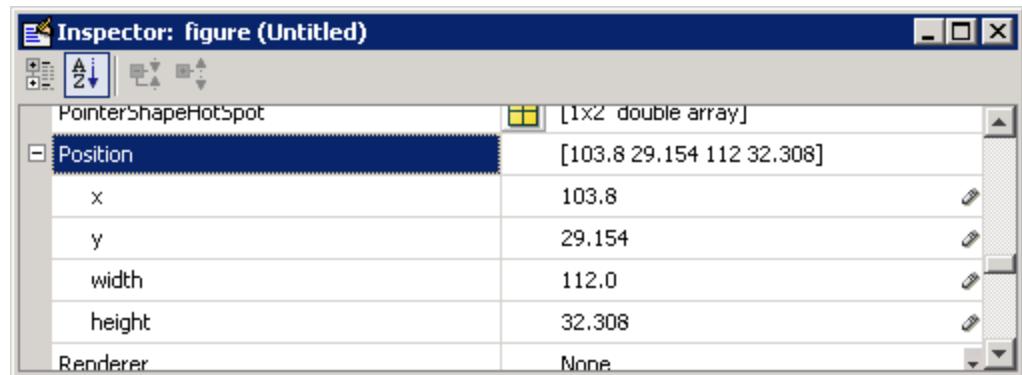
コンポーネントの Position プロパティの設定

移動したい 1 つまたは複数のコンポーネントを選択します。それから、[表示] メニューから、あるいは [プロパティインスペクター] ボタンをクリックすることによって、プロパティインスペクターを開きます。

- 1 プロパティインスペクターで Units プロパティまでスクロールし、現在の設定が characters または normalized であるか注意します。Units の隣のボタンをクリックしてから、ポップアップメニューから inches の設定を変更します。



2 Position の隣の + 記号をクリックします。プロパティインスペクターは、Position プロパティの要素を表示します。



3 選択に応じて、以下を行います。

- 1 つのコンポーネントのみを選択した場合、コンポーネントの左下隅を表示したい位置の点の x と y 座標を入力します。
- 2 つ以上のコンポーネントを選択した場合、x 座標または y 座標のいずれかを入力して、その座標についてコンポーネントを整列させます。

4 Units プロパティを前の設定、characters または normalized にリセットします。

メモ Units プロパティを characters (サイズ変更不可能な GUI) または normalized (サイズ変更可能な GUI) に設定すると、GUI はプラットフォームに依存せず同じような外見を与えられます。詳細は、“クロスプラットフォーム互換の Units” (p.6-137) を参照してください。

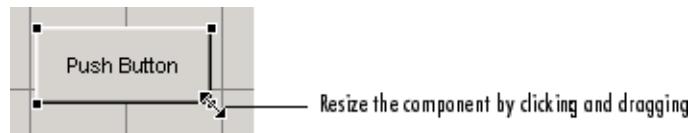
コンポーネントのサイズ変更

次のいずれかの方法で、コンポーネントをサイズ変更できます。

- “コンポーネントの隅をドラッグする” (p.6-85)
- “コンポーネントの Position プロパティの設定” (p.6-85)

コンポーネントの隅をドラッグする

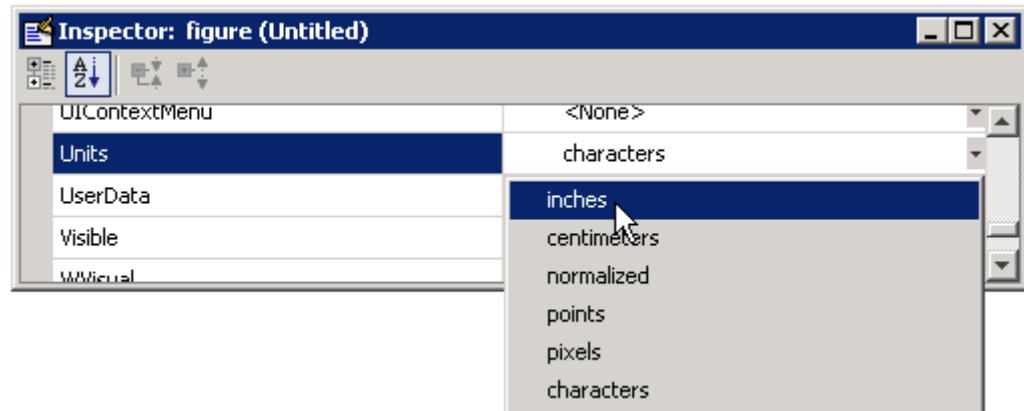
サイズ変更したいコンポーネントを選択します。コーナー ハンドルの 1 つをクリックして、コンポーネントが適切なサイズになるまでドラッグします。



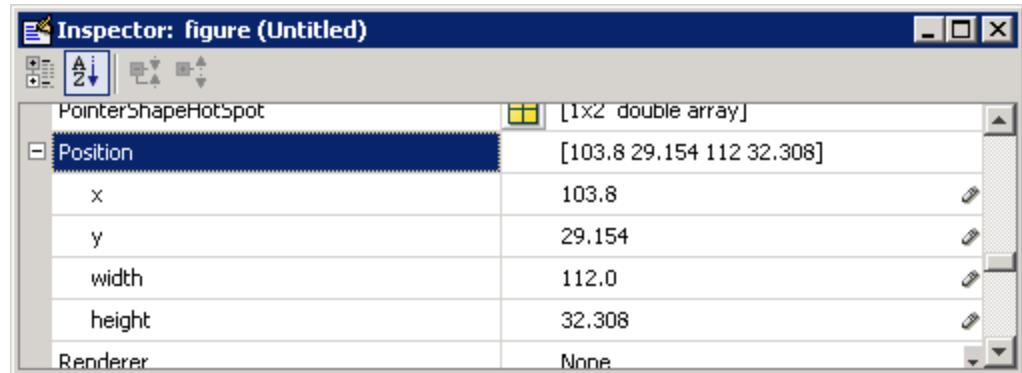
コンポーネントの Position プロパティの設定

サイズ変更したい 1 つまたは複数のコンポーネントを選択します。次に、[表示] メニューから、あるいは [プロパティインスペクター] ボタンをクリックすることによって、プロパティインスペクターを開きます。

- 1 プロパティインスペクターで Units プロパティまでスクロールし、現在の設定が characters または normalized であるか注意します。Units の隣のボタンをクリックしてから、ポップアップメニューから inches の設定を変更します。



2 Position の隣の + 記号をクリックします。プロパティインスペクターは、Position プロパティの要素を表示します。



3 コンポーネントに対する所望の width と height を入力します。

4 Units プロパティを前の設定、characters または normalized にリセットします。

メモ 複数のコンポーネントを選択するためには、それらが同じ親をもたなければなりません。つまり、それらが同じ Figure、パネル、または、ボタングループに含まれる必要があります。詳細は、“コンポーネントの選択”(p.6-78)を参照してください。Units プロパティを characters (サイズ変更不可能な GUI) または normalized (サイズ変更可能な GUI) に設定すると、GUI はプラットフォームに依存せず同じような外見を与えられます。詳細は、“クロスプラットフォーム互換の Units”(p.6-137)を参照してください。

コンポーネントの整列

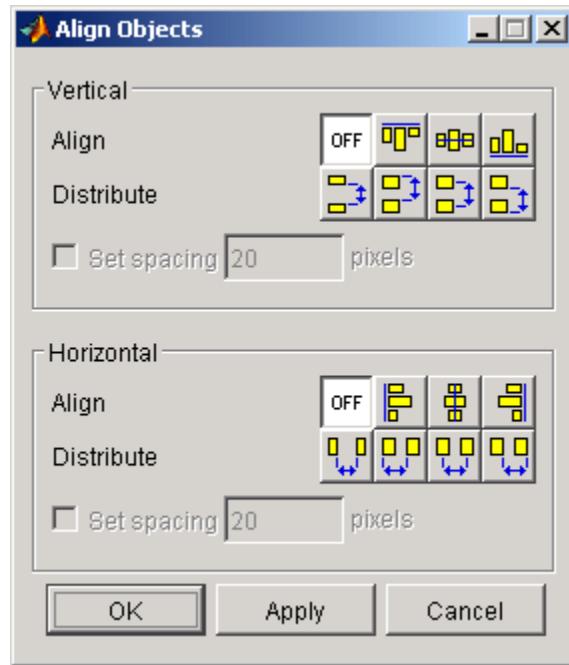
このセクションの内容…

- “配置 ツール” (p.6-87)
- “プロパティ インスペクター” (p.6-90)
- “グリッドとルーラ” (p.6-94)
- “ガイド ライン” (p.6-94)

配置 ツール

配置ツールで、オブジェクトを互いに配置することが可能になり、選択するオブジェクト間の間隔を調整します。指定する整列の操作は、[適用] ボタンを押すときに、選択したすべてのコンポーネントに適用されます。

メモ 複数のコンポーネントを選択するためには、それらが同じ親をもたなければなりません。つまり、それらが同じ Figure、パネル、または、ボタン グループに含まれる必要があります。詳細は、“コンポーネントの選択” (p.6-78) を参照してください。



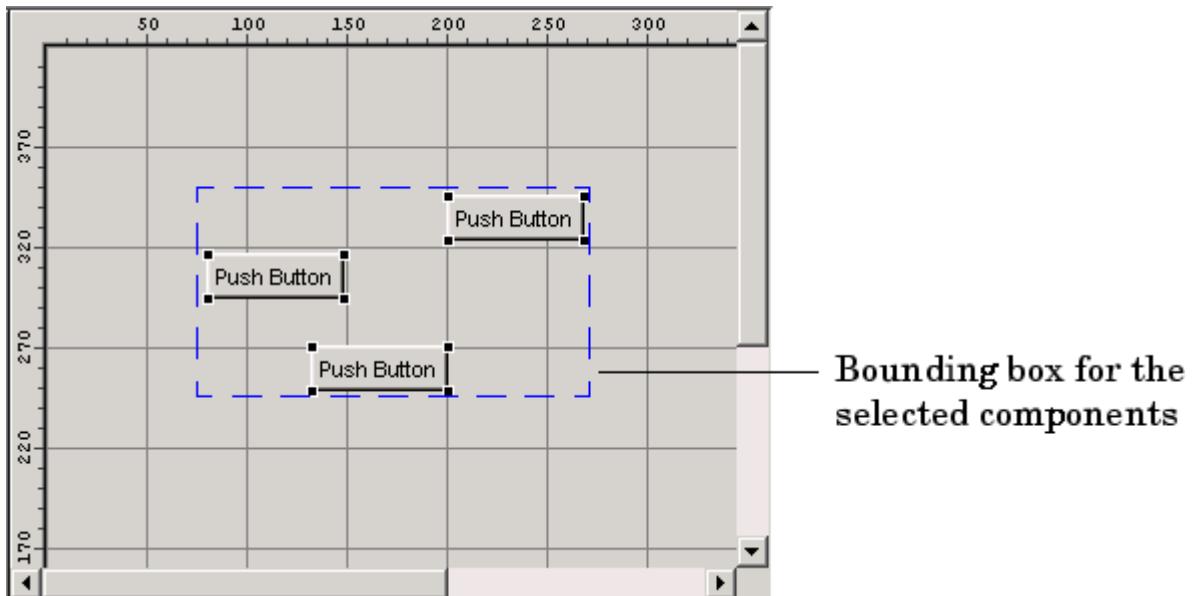
配列ツールは、2つのタイプの整列操作を提供します。

- ・ [整列] – 1つのリファレンス ラインに、選択されるすべてのコンポーネントを配置します。
- ・ [間隔] – 互いに関して、選択されるすべてのコンポーネントの間隔を、均等にとります。

両方のタイプの配置共に、水平方向と垂直方向に適用することができます。多くの場合、2つの別々の手順を利用して、垂直あるいは水平に、別々に、配置を適用する方がよいことに注意してください。

整列のオプション

垂直の整列オプションと、水平整列オプションの両方があります。各オプションは、選択するコンポーネントをリファレンス ラインに整列します。これは、選択するオブジェクトを囲む境界を示すボックスにより、決定されます。たとえば、レイアウト エリアの以下の図は、3 つの選択するプッシュ ボタンにより形成される境界を示すボックス（破線によって示される）を示します。



すべての整列オプション（垂直方向 top、center、bottom および 水平方向 left、center、right）は、境界を示すボックスの対応する端（あるいは中央）に関して、選択するコンポーネントを整列します。

間隔オプション

コンポーネントの間隔は、選択したグループのすべてのコンポーネント間に等間隔のスペースを追加します。間隔オプションは、2つの異なるモードで操作します。

- ・ 範囲内の選択されたコンポーネントを等間隔にします（既定値）。
- ・ 指定するピクセルの値で、選択されるコンポーネント間の間隔をとります（[間隔設定] をチェックし、ピクセル値を指定してください）。

両方のモードでは、整列ツール上のボタン ラベルで示されるように、間隔を測る方法を指定できます。これらのオプションは、次に關して測定される間隔を含みます。

- Vertical — 内部、最上部、中央、下部
- Horizontal — 内部、左、中央、右

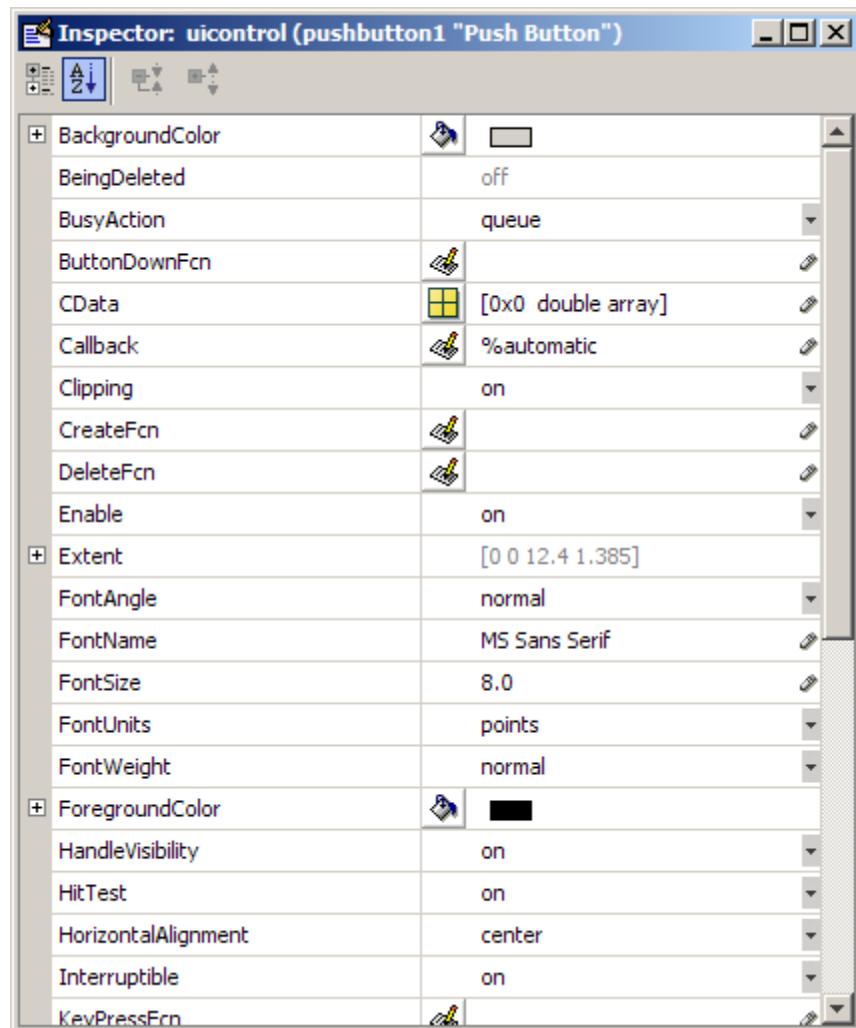
プロパティ インスペクター

プロパティ インスペクターについて

GUIDE では、通常 MATLAB で行えるように、プロパティ インスペクターを用いることで大部分のコンポーネントのプロパティを表示したり設定したりできます。GUIDE レイアウト エディターから開くには、以下のいずれかを行ってください。

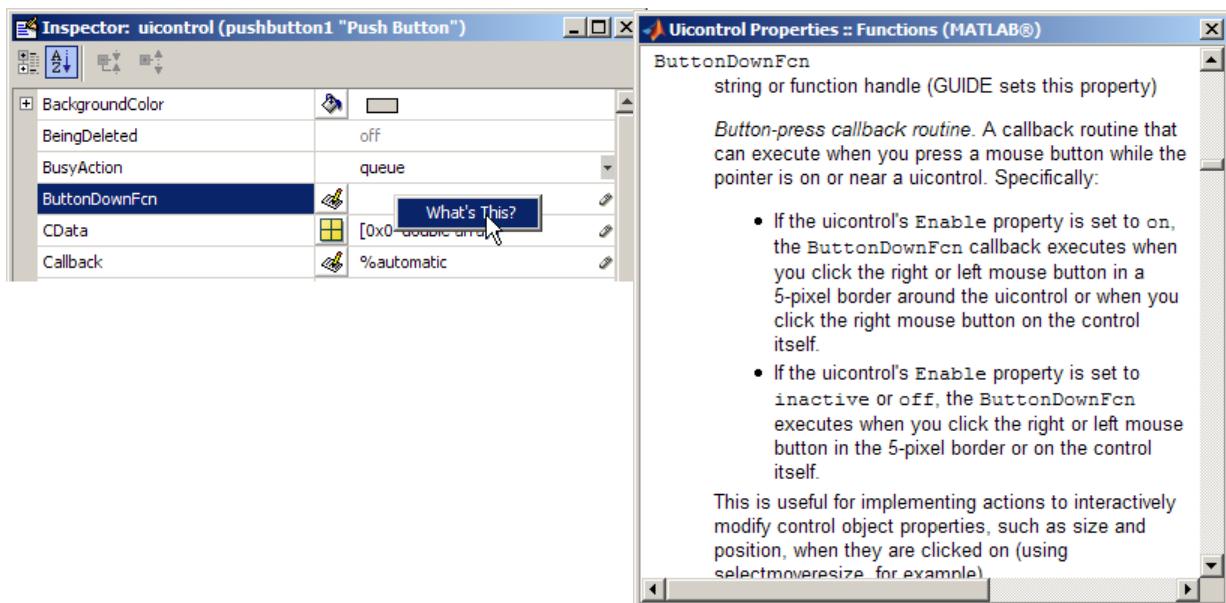
- 調べようとしているコンポーネントを選択、またはダブル クリックして、プロパティ インスペクターに開き手前に配置します。
- [表示] メニューから [プロパティ インスペクター] を選択します。
- [プロパティ インスペクター] ボタンをクリックします。

選択されたコンポーネントのプロパティを表示して、プロパティ インスペクターのウィンドウが開きます。たとえば、プッシュ ボタンのプロパティの表示は以下のようになります。



さらにプロパティを表示するためにスクロール ダウンします。その値を設定する場合、プロパティ値、またはその左にあるアイコンをクリックして、そのフィールドで直接、またはポップアップメニュー、テキストダイアログ、またはカラーピッカーなど、モーダルの GUI を使います。左マージンのプラス ボックスをクリックして、BackgroundColor、Extent と Position など、複数行のプロパティを広げます。

プロパティ インスペクターでは、個々のプロパティに対して状況依存ヘルプを見ることができます。プロパティ名または値を右クリックして、[これは何?] というメニュー項目を開きます。それをクリックすると、選択したプロパティに対する説明を表示するヘルプ ウィンドウが開きます。たとえば、右側には、左側に表示されたプロパティ インスペクターから得られた、プッシュ ボタン ButtonDownFcn に対する状況依存ヘルプがあります。



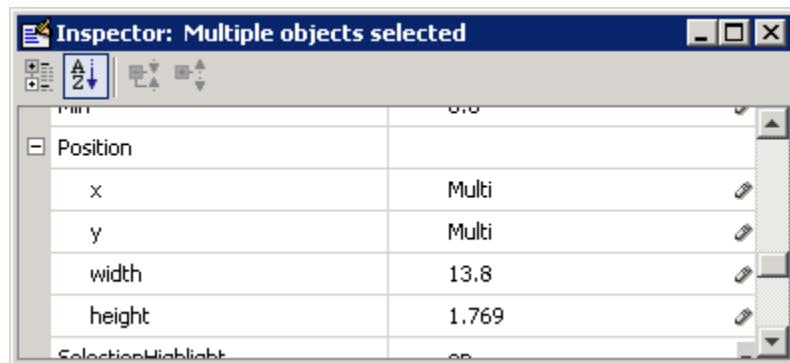
詳細は、MATLAB「グラフィックス」ドキュメンテーションの“Accessing Object Properties with the Property Inspector”を参照してください。

コンポーネントを整列するためにプロパティ インスペクターを利用する

プロパティ インスペクターによって、Position プロパティを設定してコンポーネントを整列できます。コンポーネントの Position プロパティは、GUI の位置とサイズを指定する 4 要素ベクトル [左端からの距離、下端からの距離、幅、高さ] です。値は、コンポーネントの Units プロパティによって指定される単位で与えられます。

- 1 整列させたいコンポーネントを選択します。詳細は、“コンポーネントの選択”(p.6-78)を参照してください。

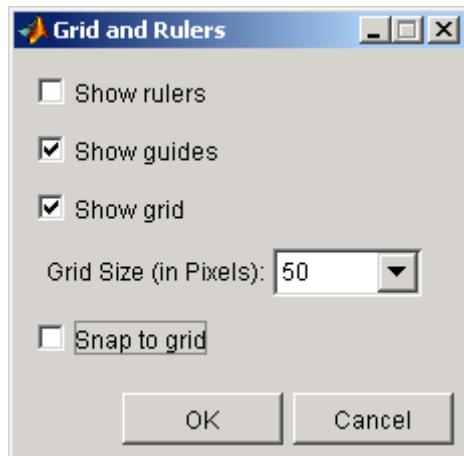
- 2 [表示] メニューから [プロパティ インスペクター] を選択するか、あるいは [プロパティ インスペクター] ボタンをクリックします。
- 3 プロパティ インスペクターで、Units プロパティをスクロールして、現在の設定に注意し、設定を inches に変更します。
- 4 Position プロパティまでスクロールします。値 null は、異なるコンポーネントに対する値が要素毎に異なることを意味します。次の図は、同じサイズの複数のコンポーネントに対する Position プロパティを示します。



- 5 左側を揃えるために x の値を変更します。y の値を変更して底辺を揃えます。たとえば、x を 2.0 に設定すると、コンポーネントの左側を GUI の左側から 2 インチに揃えます。
- 6 コンポーネントを整列する場合、Units プロパティをオリジナルの設定に戻すように変更します。

グリッドとルーラ

レイアウトエリアは、コンポーネントのレイアウトを容易にするために、グリッドとルーラを表示します。グリッドラインは、既定で、50-ピクセルの間隔があり、10から200までのピクセルの範囲の値から選択することができます。ユーザーは、オプションで、[グリッドにスナップ]を利用することができます。この操作では、グリッドラインの近くに移動されたオブジェクトをそのグリッドラインまでジャンプさせます。グリッドにスナップは、グリッド表示を使用して、あるいは使用せずに、動作します。



以下のことに対して、グリッドとルーラ ダイアログ [ツール] メニューから、[グリッドとルーラ] を選択することで、アクセスします) を使用してください。

- ・ ルーラ、グリッド、ガイドライン それぞれの表示、非表示のコントロール
- ・ グリッド間隔の設定
- ・ [グリッドにスナップ] を使用可能、あるいは使用不可とする

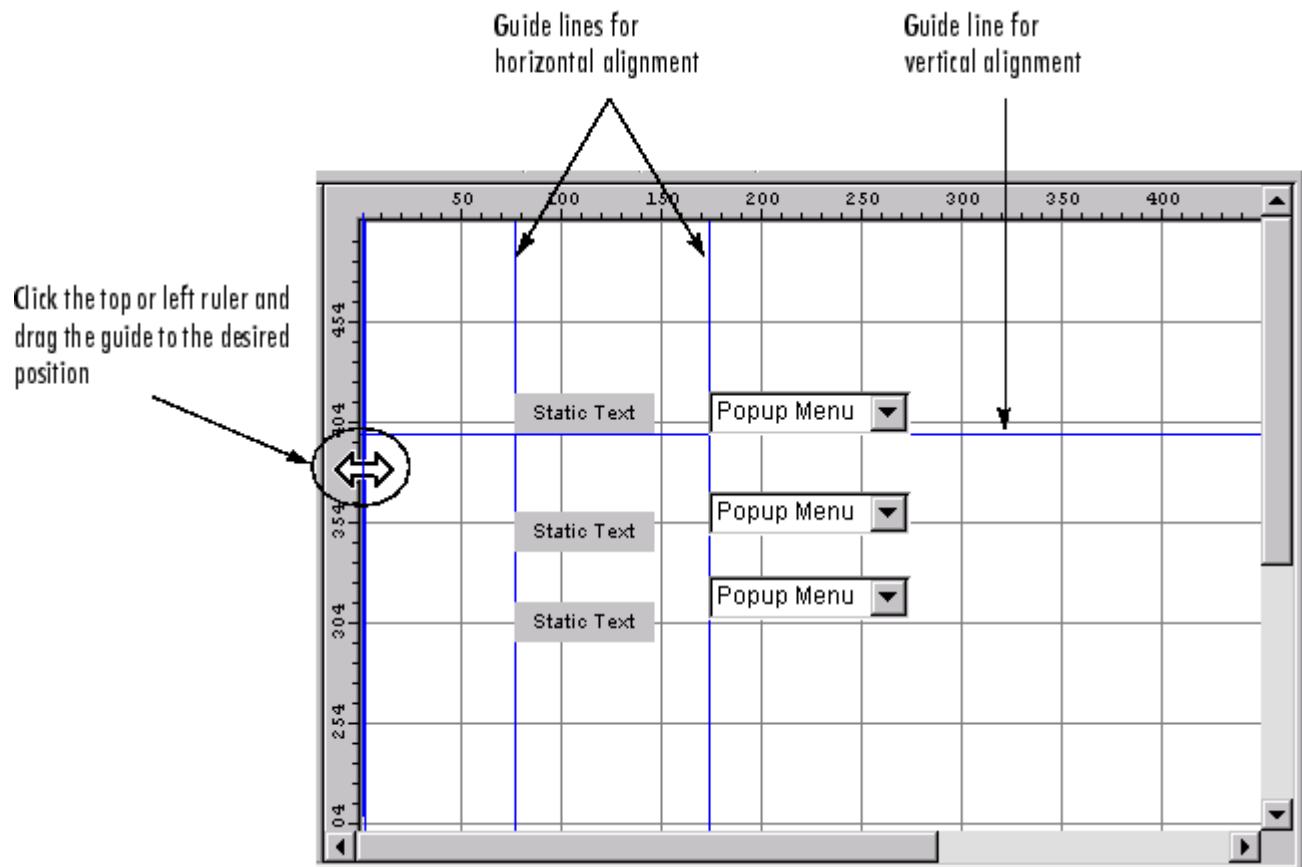
ガイドライン

レイアウトエディターは、ガイドラインに垂直 および 水平方向にスナップする機能を有しています。コンポーネントがラインの近くまで移動されると、そのラインにスナップします。

ガイドラインは、レイアウトエディターの任意の位置で、コンポーネントの配置に対して基準を定めたい場合、役立ちます。

ガイド ラインの生成

ガイド ラインを作成するために、上あるいは左のルーラ上をクリックし、ラインをレイアウト エリアにドラッグしてください。



タブの順序の設定

GUI のタブの順序は、ユーザーがキーボード上の Tab キーを押す際に、GUI のコンポーネントがフォーカスされる順序です。フォーカスは、通常、境界または点線の境界で表されます。

同じ親をもつコンポーネントのタブ順序を、独立に設定できます。GUIDE の Figure と、Figure 内の各パネルとボタン グループは、それぞれタブ順序をもちます。たとえば、Figure を親としてもつコンポーネントのタブ順序を設定できます。パネルまたはボタン グループを親としてもつコンポーネントのタブ順序も設定できます。

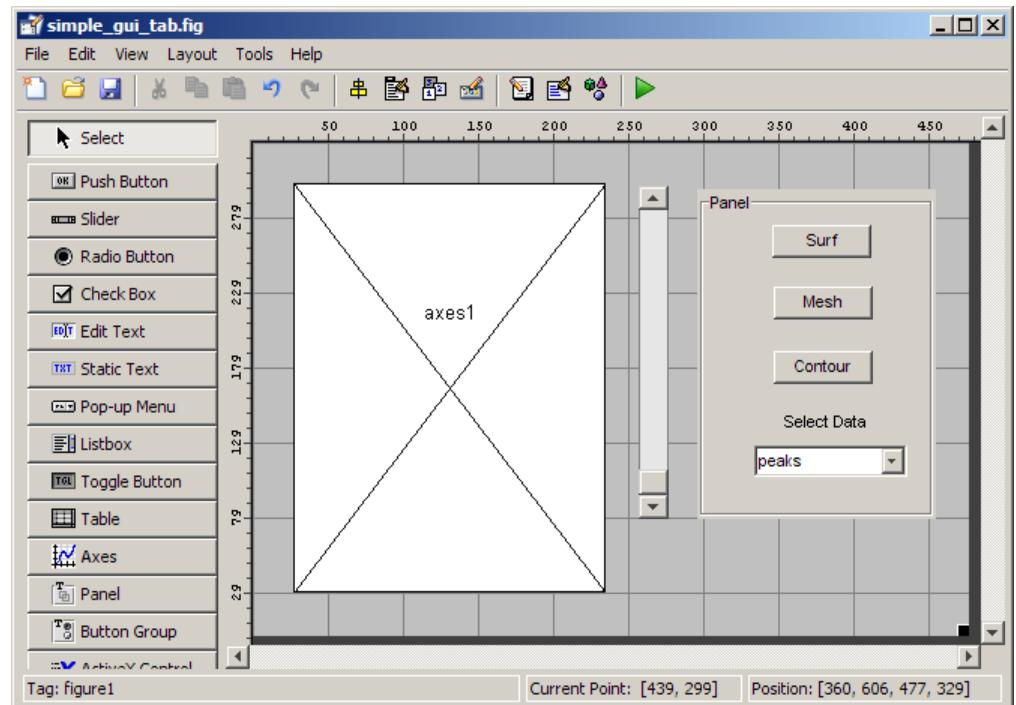
Figure レベルにあるコンポーネントにタブの順序を付ける場合、ユーザーはパネルまたはボタン グループに順序を付け、続いて、パネルまたはボタン グループに到達するレベルに戻る前に、パネルまたはボタン グループ内のコンポーネントに順序を付けます。

メモ 座標軸は、タブ順序が付けられません。GUIDE から、ActiveX コンポーネントをタブ順序に含めることはできません。

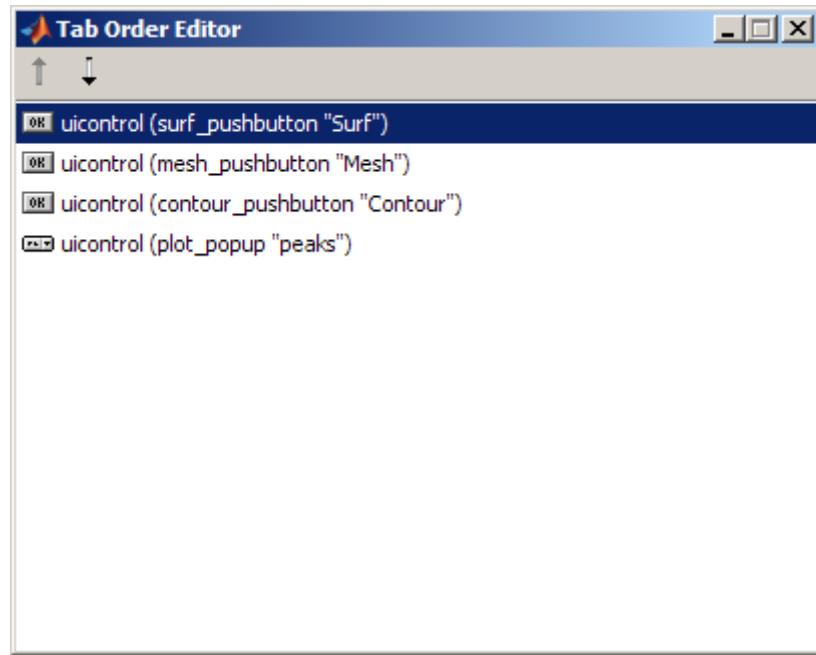
GUI を作成するとき、GUIDE は各レベルでタブの順序をユーザーがレイアウト エディターにコンポーネントを追加した順序に設定します。これは、多くの場合、ユーザーにとって最適な順序ではありません。

メモ タブの順序は、コンポーネントを積み重ねる順序にも影響します。コンポーネントが重なる場合、タブの順序が下位に現れるコンポーネントは、より上位に現れるコンポーネントの上に描かれます。詳細は、“前後位置”(p.6-80)を参照してください。

次の図の GUI は、座標軸、スライダー、パネル、スタティック テキスト、ポップアップメニューを含みます。これらの中で、Figure レベルにあるスライダー、パネル、ポップアップメニューのみタブが付けられます。パネルには、タブを付けることができる 3 つのプッシュ ボタンがあります。



パネル コンポーネントのタブ順序を調べて変更するには、パネルの背景をクリックして選択してから、レイアウト エディターの [ツール] メニューで [タブ順序編集] を選択します。



タブ順序編集は、現在のタブの順序でパネルのコンポーネントを表示します。タブの順序を変更するには、コンポーネントを選択し、上 または 下の矢印を押して、リスト内で、コンポーネントを上または下に移動します。例で最初の 3 つのコンポーネントのタブの順序を以下のように設定すると、

- 1 [Surf] プッシュ ボタン
- 2 [Contour] プッシュ ボタン
- 3 [Mesh] プッシュ ボタン

最初のタブを [Surf] プッシュ ボタンとし、次に [Contour] プッシュ ボタン、そして最後に [Mesh] プッシュ ボタンとします。続いて Figure レベルの残りのコンポーネントを選択して、タブの順序を付けます。

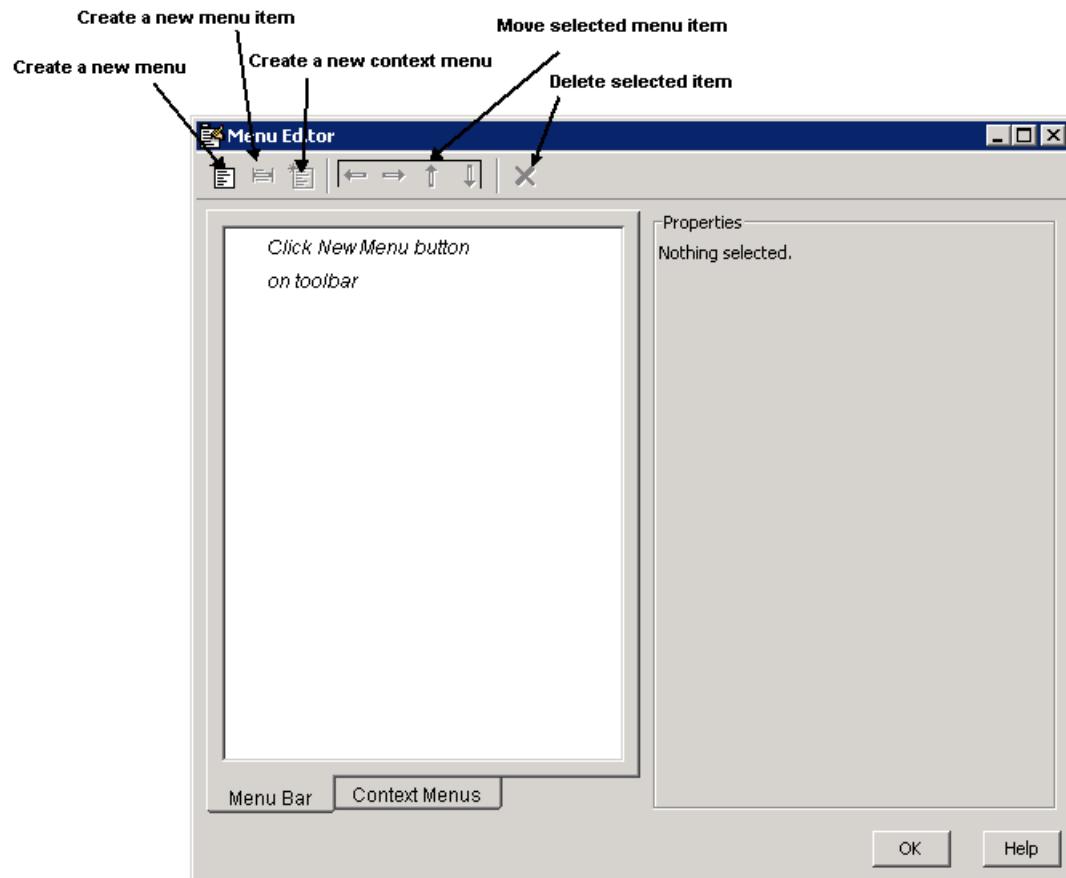
メニューの作成

このセクションの内容…

“メニュー バー に対してのメニュー” (p.6-101)

“コンテキスト メニュー” (p.6-112)

GUIDE を使用すると、コンポーネントに付加するコンテキスト メニューと同様にプルダウン メニューで GUI メニュー バーを使用できます。メニュー エディターを使用して、どちらのタイプのメニューも作成できます。[ツール] メニューからメニュー エディターにアクセスするか、[メニュー エディター] ボタンをクリックします。



メモ 一般に、章 8, “GUIDE GUI のプログラミング”で、コンポーネントに対して述べるプログラミングの規則は、メニュー項目にも適用されます。プログラミングと基本的な例についての詳細は、“メニュー項目”(p.8-59)と“メニュー項目のチェックの更新”(p.8-60)を参照してください。

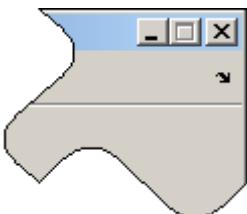
メニュー バー に対してのメニュー

- ・ “メニューが Figure のドックにどのように影響するか” (p.6-101)
- ・ “メニュー バーへの標準メニューの追加” (p.6-102)
- ・ “メニューの作成” (p.6-104)
- ・ “メニューに項目を追加する” (p.6-107)
- ・ “追加のドロップダウン メニュー” (p.6-109)
- ・ “カスケード メニュー” (p.6-110)

ドロップダウン メニューを作成する場合、GUIDE は GUI メニュー バーにタイトルを追加します。そのメニューに対してユーザーがメニュー項目を作成できます。各メニュー項目は、カスケードするメニュー（サブメニュー）をもつことができ、これらの項目も同様にカスケードするメニューをもつことができます。

メニューが Figure のドックにどのように影響するか

既定では、GUIDE を使用して GUI を作成する場合、GUIDE は GUI に対してメニュー バーを作成しません。GUI に対してメニューは必要ありませんが、ユーザーが GUI をドックしたりアンドックしたりできるようにするには、メニュー バーまたはツールバーを含む必要があります。これは、以下の図が示すように、ドックがドックのためのアイコンでコントロールされているためです。このアイコンは、メニュー バーまたはツールバーの右上隅付近の小さな曲がった矢印です。



標準メニュー バーをもつ Figure ウィンドウにも、[デスクトップ] メニューがあります。ユーザーはこのメニューから Figure ウィンドウをドックしたりアンドックすることができます。

ドックのための矢印と [デスクトップ] > [Figure をドック] メニュー項目を表示するには、プロパティインスペクターを使用して Figure のプロパティ DockControls を 'on' に設定します。ドックのコントロールを表示するには、MenuBar および/またはToolBar Figure プロパティを 'on' に設定する必要があります。

Figure プロパティ `WindowStyle` もドックの動作に影響します。既定値は '`normal`' ですが、'`docked`' に変更した場合、以下を適用します。

- ・ GUI は、それが実行しているときデスクトップにドックされた状態で開きます。
- ・ `DockControls` プロパティは '`on`' に設定されますが、`WindowStyle` が '`docked`' に設定されるまでは、オフにすることはできません。
- ・ `WindowStyle` '`docked`' で作成された GUI をアンドックすると、Figure がメニュー バーまたはツールバー（標準またはカスタマイズされた）表示しない限り、ドックのための矢印は表示されません。ドックのための矢印がないと、ユーザーはデスクトップからアンドックできますが、再びドックし直すことはできません。

ただし、GUIDE を使用して独自のメニュー バーまたはツールバーを与えると、GUI をドック可能としたい場合にドックのための矢印を表示できます。詳細は、以下の節と“ツールバーの作成”(p.6-120) を参照してください。

メモ モーダルなダイアログである GUI (`WindowStyle` を '`modal`' に設定した Figure) は、メニュー バー、ツールバー、ドックのコントロールをもつことができません。

詳細は、「Figure プロパティ」のリファレンス ページの `DockControls`、`MenuBar`、`ToolBar`、`WindowStyle` プロパティの説明を参照してください。あるいは、GUIDE の右クリックメニューのプロパティ インスペクターのこれらの名前において Figure の背景色を選択します。

メニュー バーへの標準メニューの追加

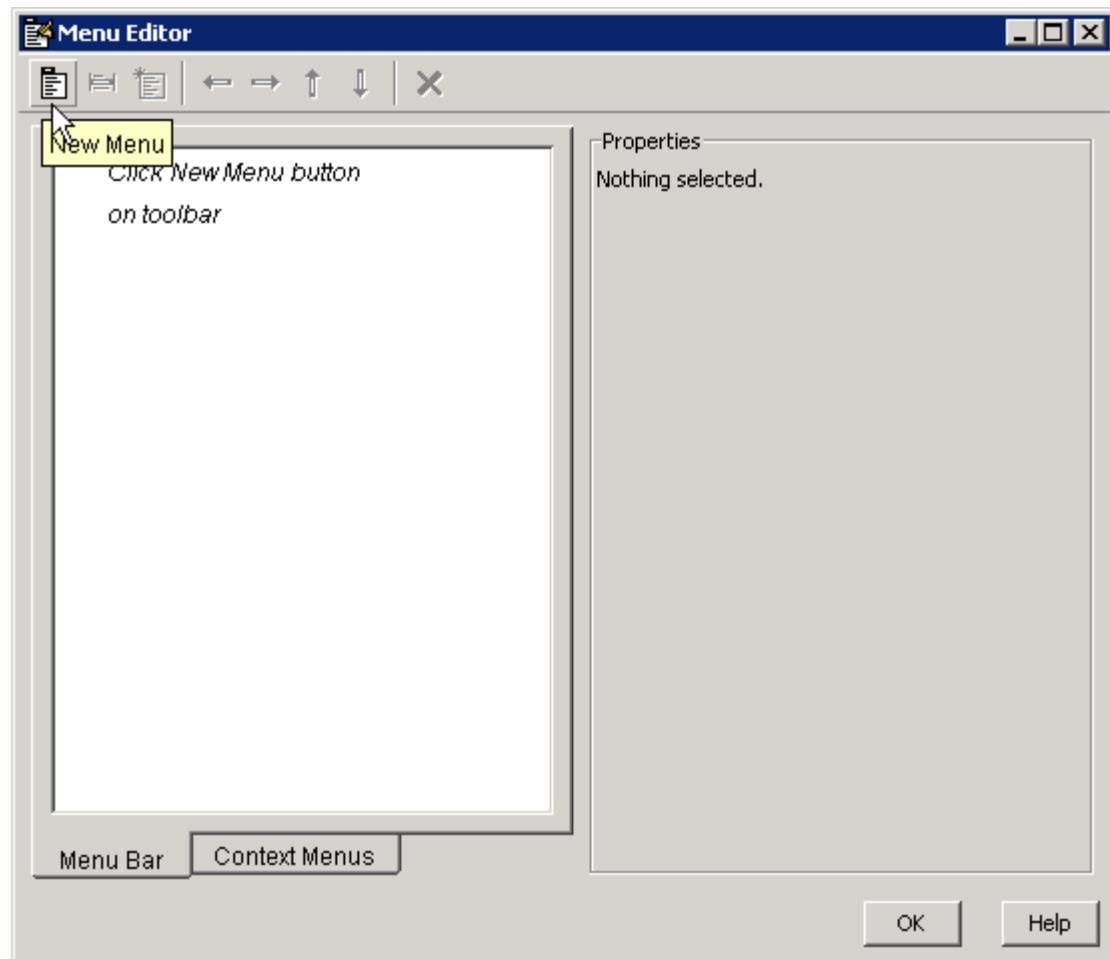
Figure の `MenuBar` プロパティは、ユーザー GUI が MATLAB の標準のメニューをメニュー バー上に表示するかどうかをコントロールします。GUIDE は `MenuBar` の値を `none` に初期設定します。GUI に MATLAB の標準メニューを表示させたい場合、プロパティ インスペクターを用いて `figure` に `MenuBar` を設定します。

- ・ `MenuBar` の値が `none` である場合、GUIDE は、作成するメニューのみを表示するメニュー バーを自動的に追加します。
- ・ `MenuBar` の値が `figure` である場合、GUI は MATLAB の標準メニューを表示し、GUIDE は作成するメニューをメニュー バーの右側に追加します。

いずれの場合においても、GUI のユーザーが Figure の DockControls プロパティを 'on' に設定して、GUI のドックのための矢印を使用して、GUI をドックしたり、アンドックすることができます。

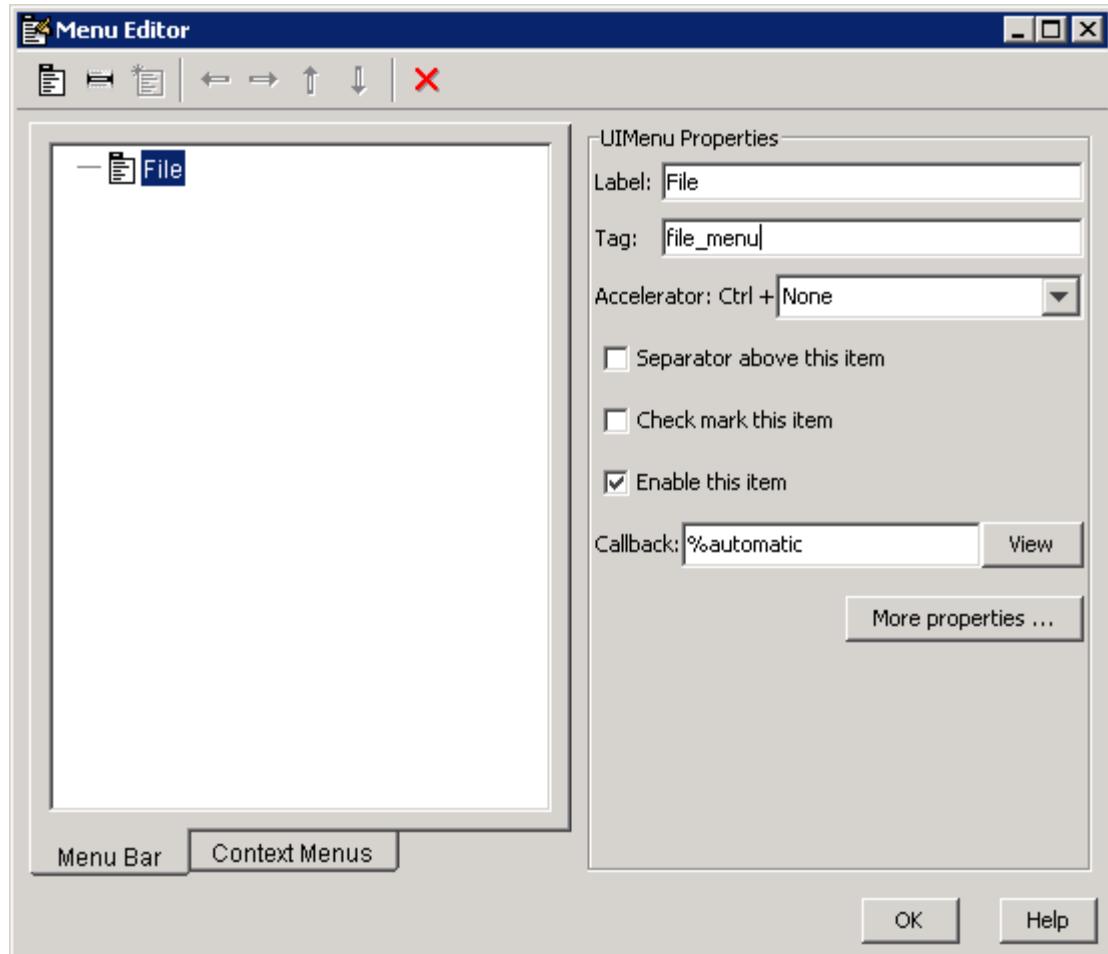
メニューの作成

- 新しいメニューは、ツールバーの新規メニュー ボタンをクリックすることから始めます。メニューのタイトル Untitled 1 がダイアログ ボックスの左ペインに表示されます。



メモ 既定では、GUIDE はメニュー エディターを開くと [メニュー バー] タブを選択しています。

2 メニュー タイトルをクリックして、メニュー プロパティの選択を右のペインに表示します。



3 新規メニュー項目に対する [ラベル] および [タグ] フィールドを編集してください。たとえば、[ラベル] を 'File' に [タグ] を 'file_menu' に設定します。変更が有効になるように、フィールドの外側をクリックします。

[ラベル] は、メニュー項目に対するテキストラベルを指定する文字列です。ラベルに "&" 文字を表示するには、文字列に 2 つの & 文字を使用します。remove、

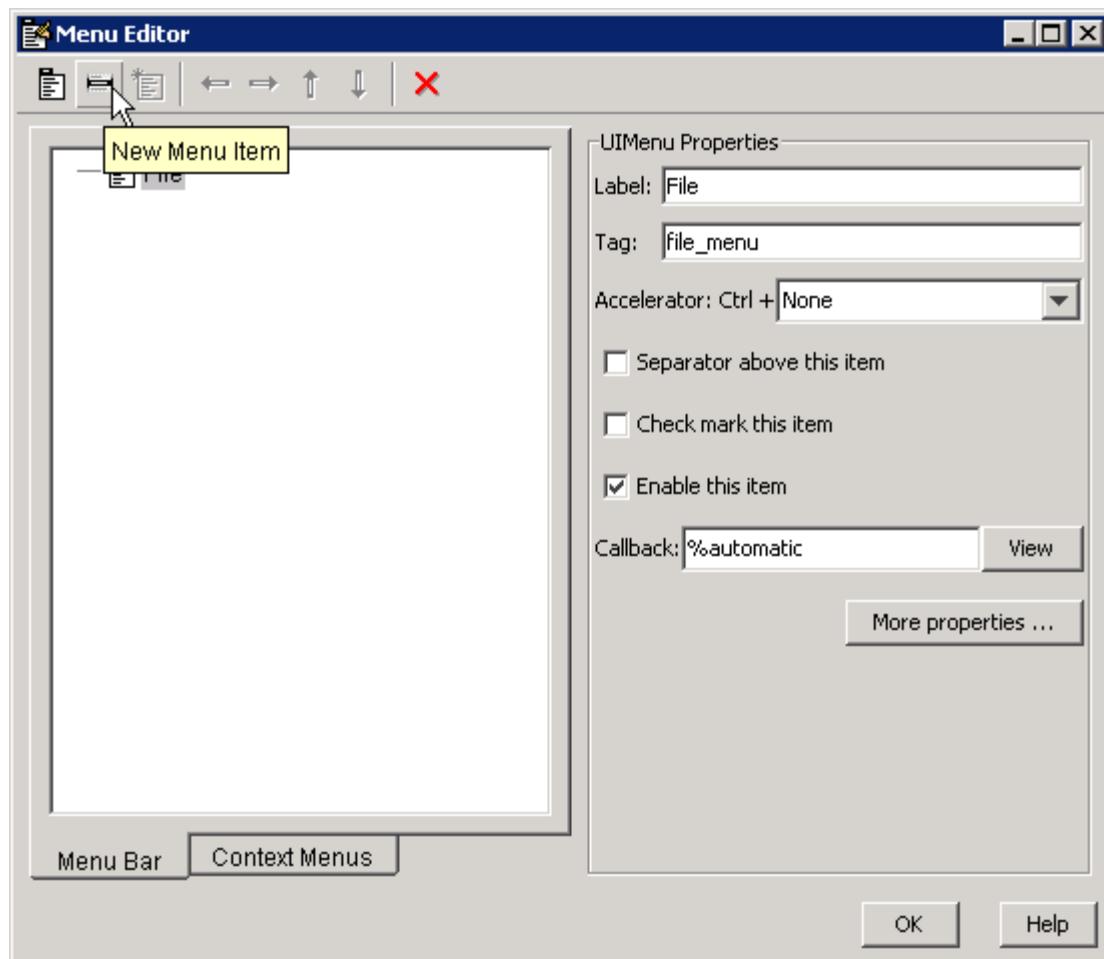
default、factory (大文字と小文字の区別あり) は予約語です。これらのいずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ (\) を付けます。たとえば、\\$remove は remove を与えます。

[タグ] はメニュー オブジェクトに対する識別子の文字列です。これは、メニュー項目を識別するためにコードで使用され、GUI で一意である必要があります。

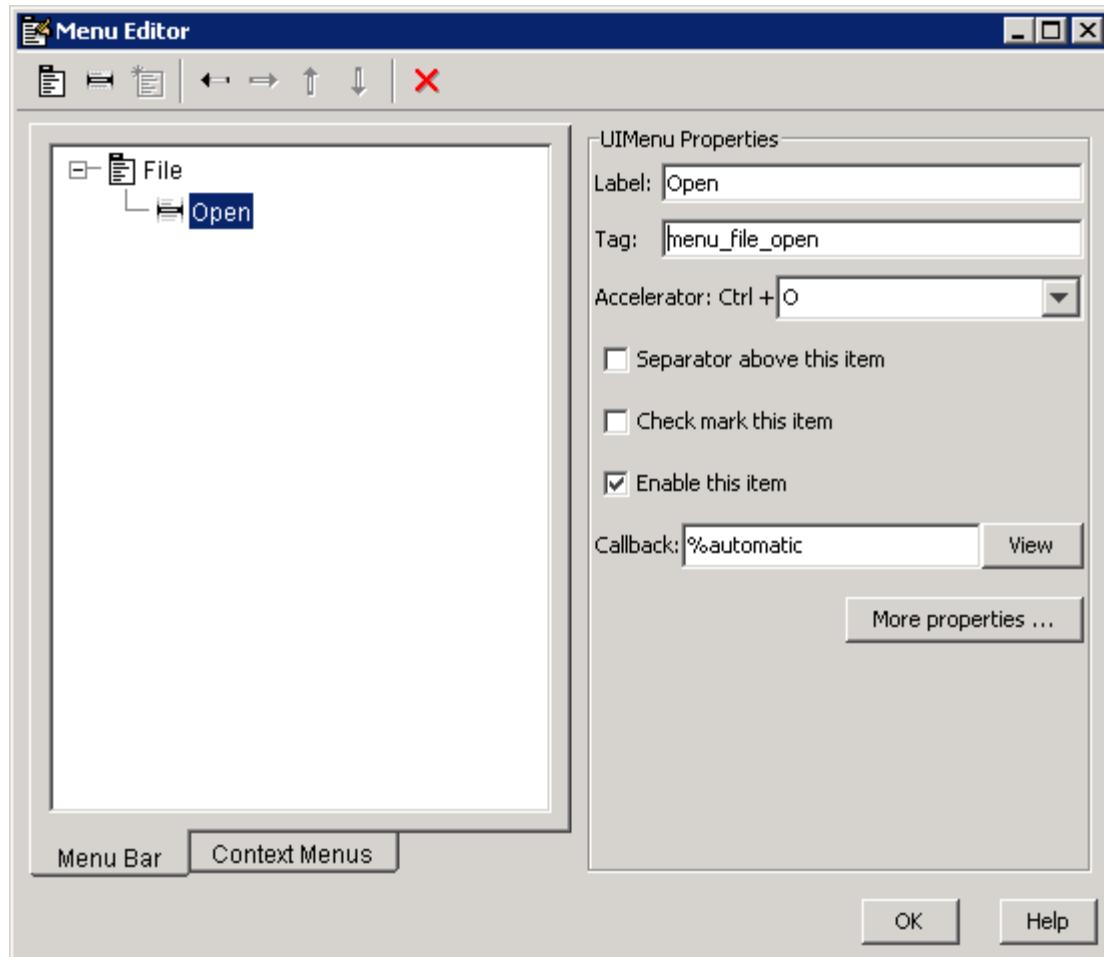
メニューに項目を追加する

ドロップダウンメニューにメニュー項目を作成するには、[新規メニュー項目] ツールを使用してください。

- 1 File を選択して、ツールバーの [新規メニュー項目] ボタンをクリックすることによって、File の下に [開く] メニュー項目を追加します。一時的な番号付けされたメニュー項目のラベル Untitled が表示されます。



- 2 新規メニュー項目に対する [ラベル] および [タグ] フィールドを編集してください。たとえば、[ラベル] を Open に [タグ] を menu_file_open に設定します。変更が有効になるように、フィールドの外側をクリックします。



以下を行うこともできます。

- ・ [Accelerator] ポップアップメニューを使用して、メニュー項目に alphabetic キーボード アクセラレータを選択します。Ctrl とあわせてキーボードをタイプすることは、子のメニューをもたないメニュー項目を選択したものと等価の動作をします。

アクセラレータはシステム上で他の目的で使用されるものもあり、結果として他のアクションが起こることもありますので、注意してください。

- ・ [この項目の上に区切りを入れる] をチェックすると、メニュー項目の上に区切りを表示します。
 - ・ [この項目にチェックマークを入れる] をチェックすると、メニューが最初に開かれた場合、メニュー項目の隣にチェックを表示します。チェックは、メニュー項目の現在の状態を示します。“コンテキストメニューに項目を追加”(p.6-114)の例を参照してください。
 - ・ [この項目を有効にする] をチェックすると、最初にメニューが開かれたときこの項目を有効にします。これによって、メニューが最初に開かれたときこの項目を選択できます。このオプションをクリアすると、メニューが最初に開くときにメニュー項目がグレーで表示されてユーザー側では選択できなくなります。
 - ・ メニュー項目に関連するアクションを実行するルーチン、つまり [コールバック] に対する文字列を指定します。GUI を保存していない場合は、既定値は %automatic です。GUI を保存し、このフィールドを変更していなかった場合、GUIDE は [タグ] フィールドと GUI ファイル名を組み合わせて値を自動的に設定します。このフィールドの指定とメニュー項目のプログラミングについての詳細は、“メニュー項目”(p.8-59)を参照してください。
- [表示] ボタンは、エディターにコールバック サブ関数を表示します。GUI を保存していない場合、GUIDE は保存するように指示します。
- ・ [オプション] ボタンをクリックすると、プロパティインスペクターを開き、すべてのメニュー プロパティを変更できます。プロパティについての詳細は、MATLAB ドキュメンテーションの「Uimenu プロパティ」を参照してください。

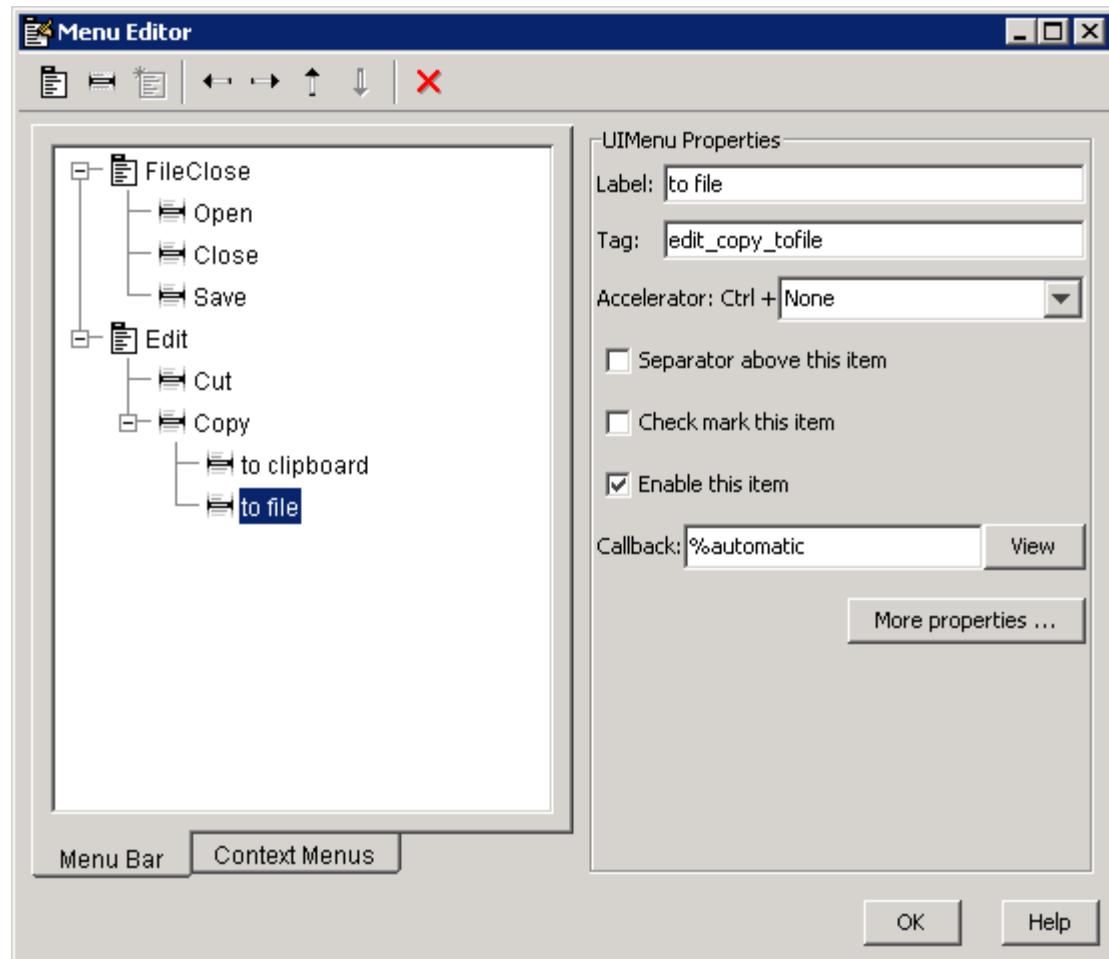
メモ 一般に、章 8, “GUIDE GUI のプログラミング”で、コンポーネントに対して述べるプログラミングの規則は、メニュー項目にも適用されます。プログラミングと基本的な例についての詳細は、“メニュー項目”(p.8-59)と“メニュー項目のチェックの更新”(p.8-60)を参照してください。

追加のドロップダウン メニュー

追加のドロップダウン メニューを作成するためには、File メニューを作成した場合と同じように、新規メニュー ボタンを使用します。たとえば、次の図は、Edit ドロップダウン メニューも示します。

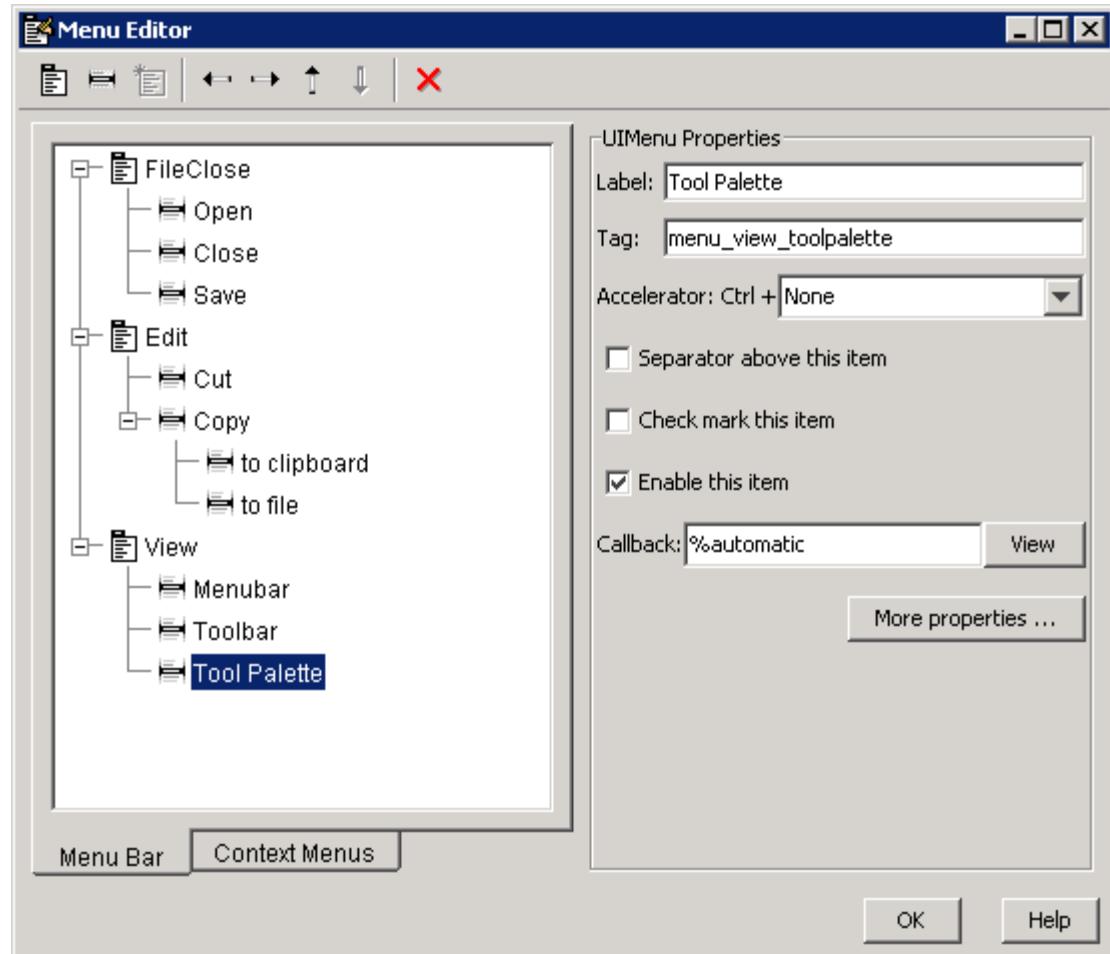
カスケード メニュー

カスケードするメニューを作成するためには、タイトルになるメニュー項目を選択し、[新規メニュー項目] ボタンをクリックします。下記の例において、Copy はカスケードメニューです。

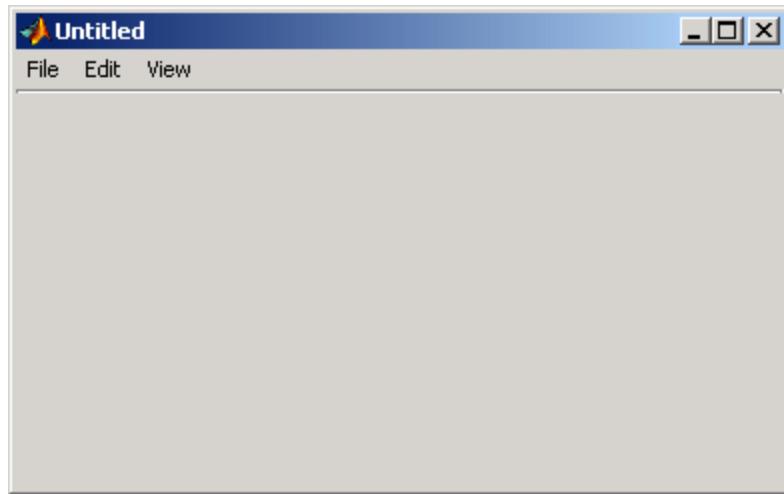


メモ メニュー項目のプログラミングについての詳細は、“メニュー項目”(p.8-59)を参照してください。

次のメニュー エディターの図は、Figure メニュー バーに対して定義される 3 つ のメニューを示します。



ユーザーが GUI を実行すると、メニュー バーにメニューが現れます。



コンテキストメニュー

コンテキストメニューは、メニューが定義されているオブジェクト上を右クリックすると表示されます。メニュー エディターは、コンテキストメニューの定義を可能とし、レイアウト内のオブジェクトと関連付けます。3つの手順があります。

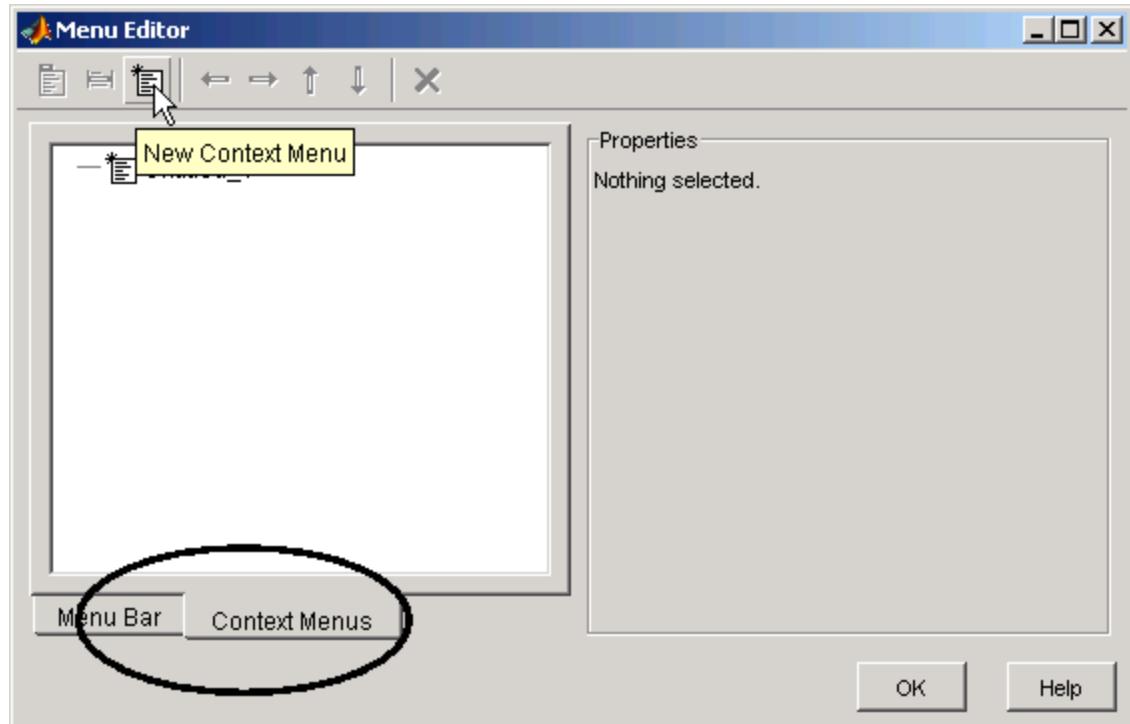
- 1 “親メニューの作成” (p.6-112)
- 2 “コンテキストメニューに項目を追加” (p.6-114)
- 3 “コンテキストメニューとオブジェクトの関連付け” (p.6-118)

メモ メニューの定義についての詳細は、“メニュー バー に対してのメニュー” (p.6-101)を参照してください。メニューに対するコールバック サブファンクションの定義についての詳細は、“メニュー項目” (p.8-59)を参照してください。

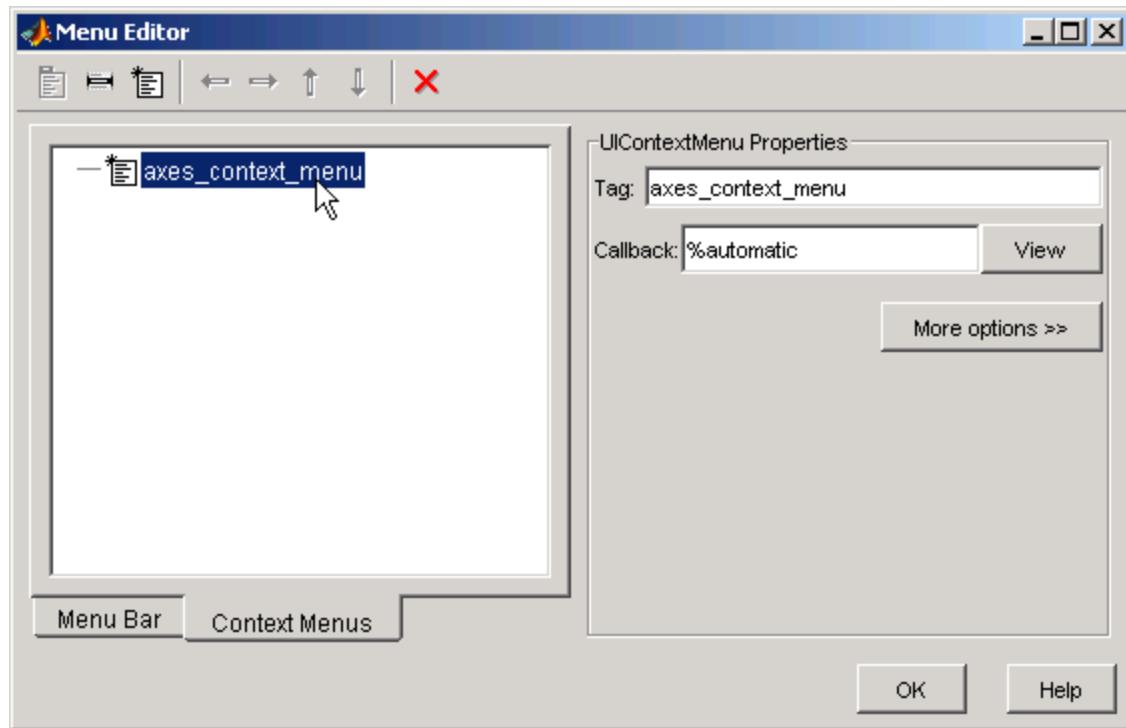
親メニューの作成

コンテキストメニュー内のすべての項目は、Figure メニュー バーに表示されないメニューの子です。親メニューを定義するには、

- 1 メニュー エディターの [コンテキストメニュー] タブを選択し、ツールバーから [新規コンテキストメニュー] を選択します。



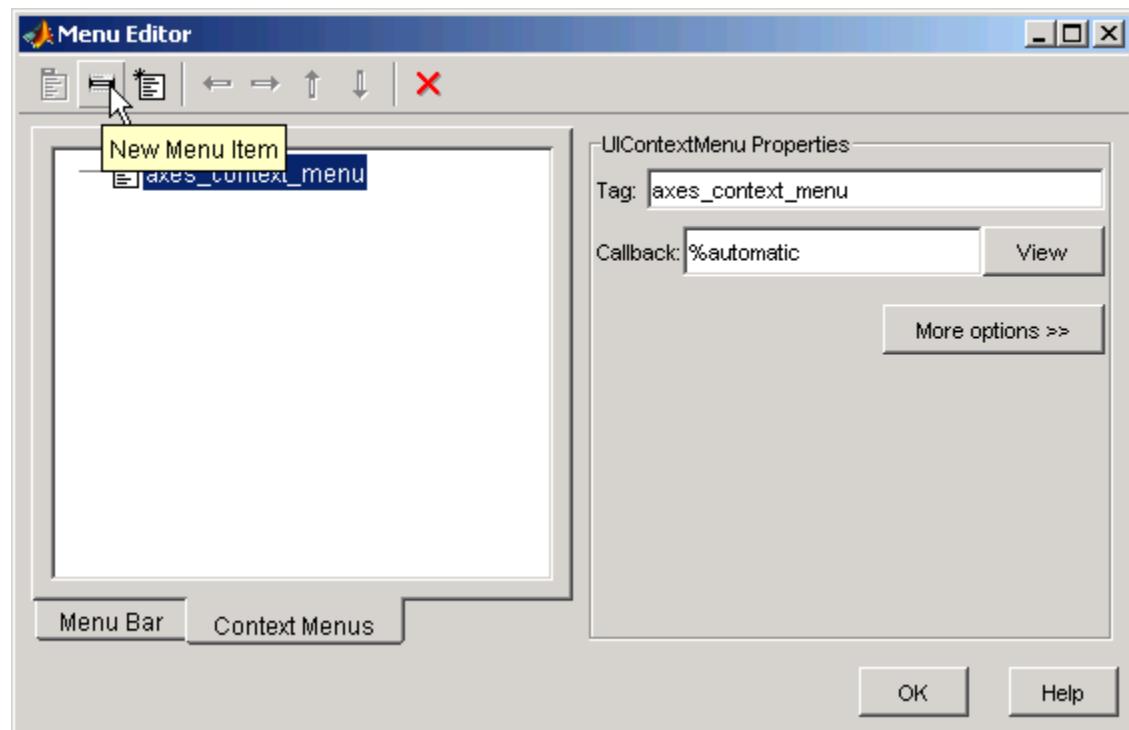
2 メニューを選択し、コンテキストメニューを識別するために、[Tag] フィールドを指定してください（この例では、axes_context_menu）。



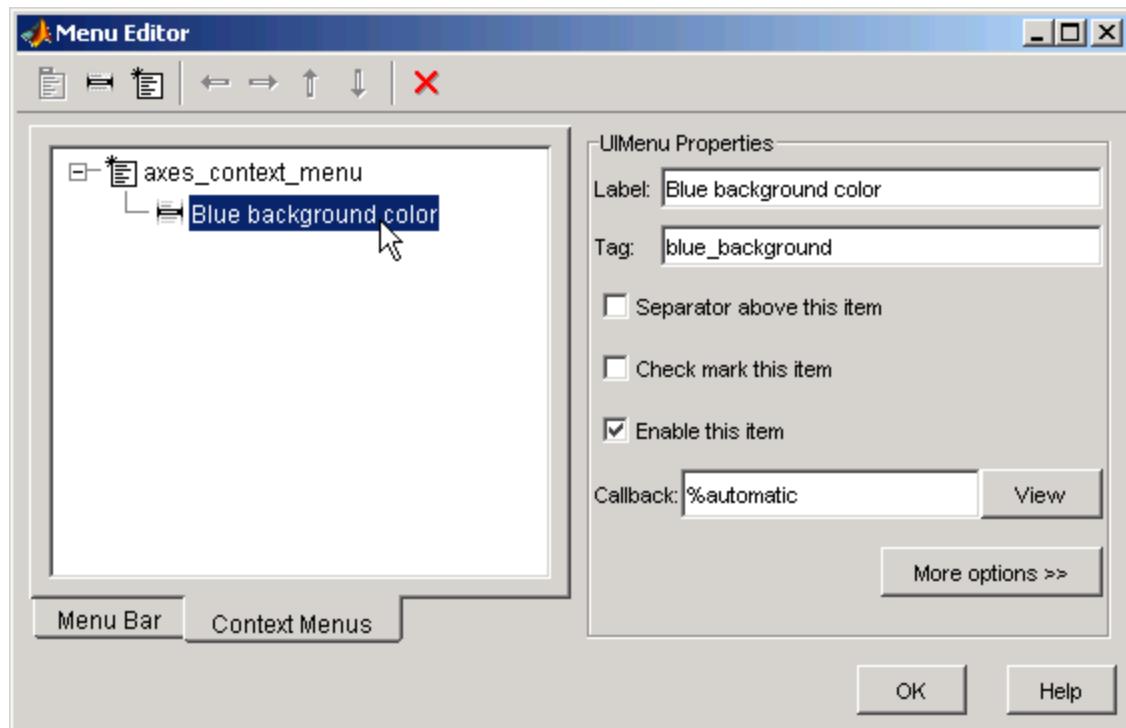
コンテキストメニューに項目を追加

コンテキストメニューにメニュー項目を作成するためには、[新規メニュー項目] ボタンを使用してください。

1 axes_context_menu を選択し、[新規メニュー項目] ツールをクリックすることによって、メニューに [Blue background color] メニュー項目を追加します。一時的な番号付けをされたメニュー項目のラベル Untitled が表示されます。



- 2 新規メニュー項目に対する [ラベル] および [タグ] フィールドを編集してください。たとえば、[ラベル] を Blue background color に [タグ] を blue_background に設定します。変更が有効になるように、フィールドの外側をクリックします。



以下を行うこともできます。

- [この項目の上に区切りを入れる]をチェックすると、メニュー項目の上に区切りを表示します。
- [この項目にチェックマークを入れる]をチェックすると、メニューが最初に開かれた場合、メニュー項目の隣にチェックを表示します。チェックは、メニュー項目の現在の状態を示します。“コンテキストメニューに項目を追加”(p.6-114)の例を参照してください。コードの例は、“メニュー項目のチェックの更新”(p.8-60)を参照してください。
- [この項目を有効にする]をチェックすると、最初にメニューが開かれたときこの項目を有効にします。これによって、メニューが最初に開かれたときこの項目を選択

できます。このオプションを選択解除すると、メニューが最初に開くときにメニュー項目がグレーで表示されてユーザー側では選択できなくなります。

- メニュー項目に関連する動作を実行するメニューで [Callback] を指定します。GUI を保存していない場合は、既定値は %automatic です。GUI を保存し、このフィールドを変更していなかった場合、[Tag] フィールドと GUI フィールド名を組み合わせて使用し、GUIDE は M ファイルにコールバックを自動的に作成します。コールバック名は、メニュー エディターの [Callback] フィールドに表示されませんが、メニュー項目を選択するとコールバックが実行します。

コールバックとして動作するために [Callback] フィールドに引用符で囲まれていない文字列を入力することもできます。その際、有効な MATLAB 表現またはコマンドを用いることができます。たとえば、以下の文字列

```
set(gca, 'Color', 'y')
```

は、現在の Axes の背景色を黄色に設定します。ただし、この動作を実行するためには推奨される方法は、GUI M ファイルでコールバックを配置することです。これにより、いくつかの Figure または Axes が存在する場合必ずしも確実であるとは限らない、gca は利用されません。このコールバックの M ファイル バージョンがあります。

```
function axesyellow_Callback(hObject, eventdata, handles)
% hObject    handle to axesyellow (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.axes1, 'Color', 'y')
```

このコードは、コンテキストメニューが付加されているオブジェクトが何であっても、Tag axes1 を用いて GUI Axes の背景色を設定します。

メニュー エディターでコールバック文字列を入力すると、M ファイル内の項目に対するコールバックが保存されている場合は、コールバックがオーバーライドされます。[Callback] フィールドに入力した文字列を削除すると、GUI が実行し、その項目が選択される場合に、その項目に対する M ファイルのコールバックが実行されます。

このフィールドの指定とメニュー項目のプログラミングについての詳細は、“メニュー項目”(p.8-59)を参照してください。GUIDE でのコンテキストメニューのプログラミングの他の例は、“テーブルのデータを対話的に調べる GUI”(p.10-32)を参照してください。

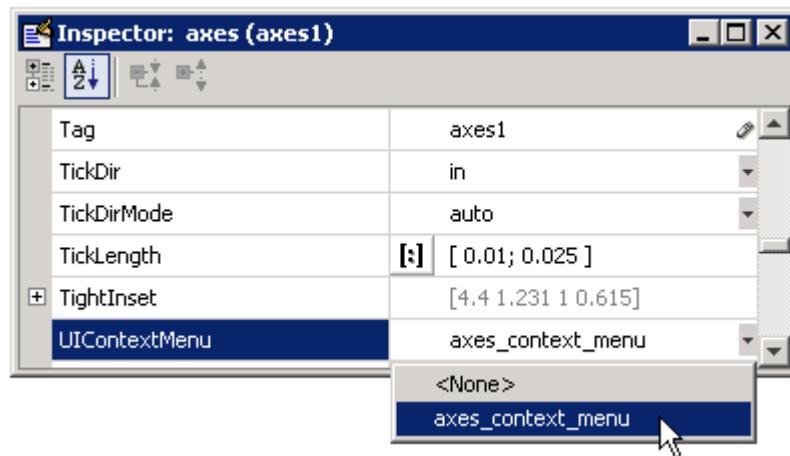
[表示] ボタンは、エディターにコールバック サブ関数を表示します。GUI を保存していない場合、GUIDE は保存するように指示します。

- プロパティ インスペクターを開きます。プロパティ インスペクターでは、[More options] ボタンをクリックすることで、コールバックを除くすべてのメニュー プロパティを変更できます。これらのプロパティについての詳細は、MATLAB ドキュメンテーションの「Uicontextmenu プロパティ」を参照してください。

コンテキスト メニューとオブジェクトの関連付け

- レイアウト エディターで、コンテキスト メニューを定義するオブジェクトを選択してください。
- プロパティ インスペクターを使用して、このオブジェクトのUIContextMenu プロパティを希望するコンテキスト メニュー名に設定してください。

次の図は、Tag プロパティを axes1 にした axes オブジェクトに対する UIContextMenu プロパティを示します。



GUI M ファイルで、コンテキスト メニューの各項目に対するコールバック サブファンクションを完成します。各コールバックは、ユーザーが、関連するコンテキスト メニュー項目を選択した場合に実行します。構文の定義についての詳細は、“メニュー項目”(p.8-59)を参照してください。

メモ 一般に、章 8, “GUIDE GUI のプログラミング”で、コンポーネントに対して述べるプログラミングの規則は、メニュー項目にも適用されます。プログラミングと基本的な例についての詳細は、“メニュー項目”(p.8-59)と“メニュー項目のチェックの更新”(p.8-60)を参照してください。

ツールバーの作成

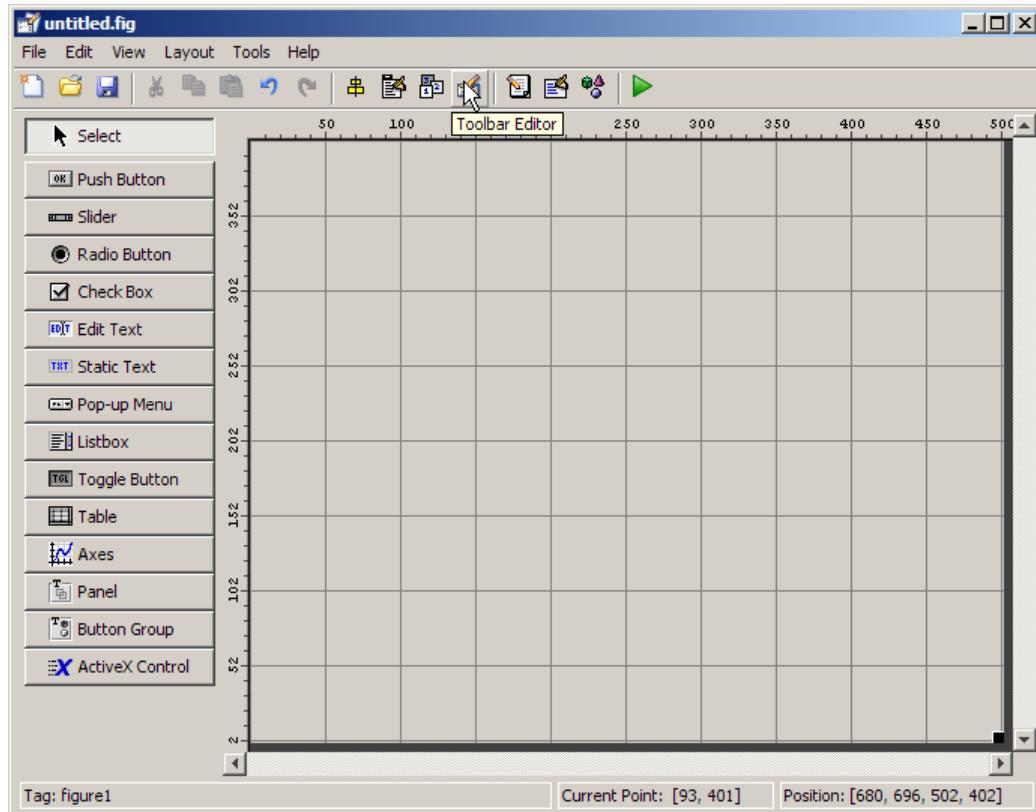
このセクションの内容…

“GUIDE を使用してツールバーを作成する” (p.6-120)

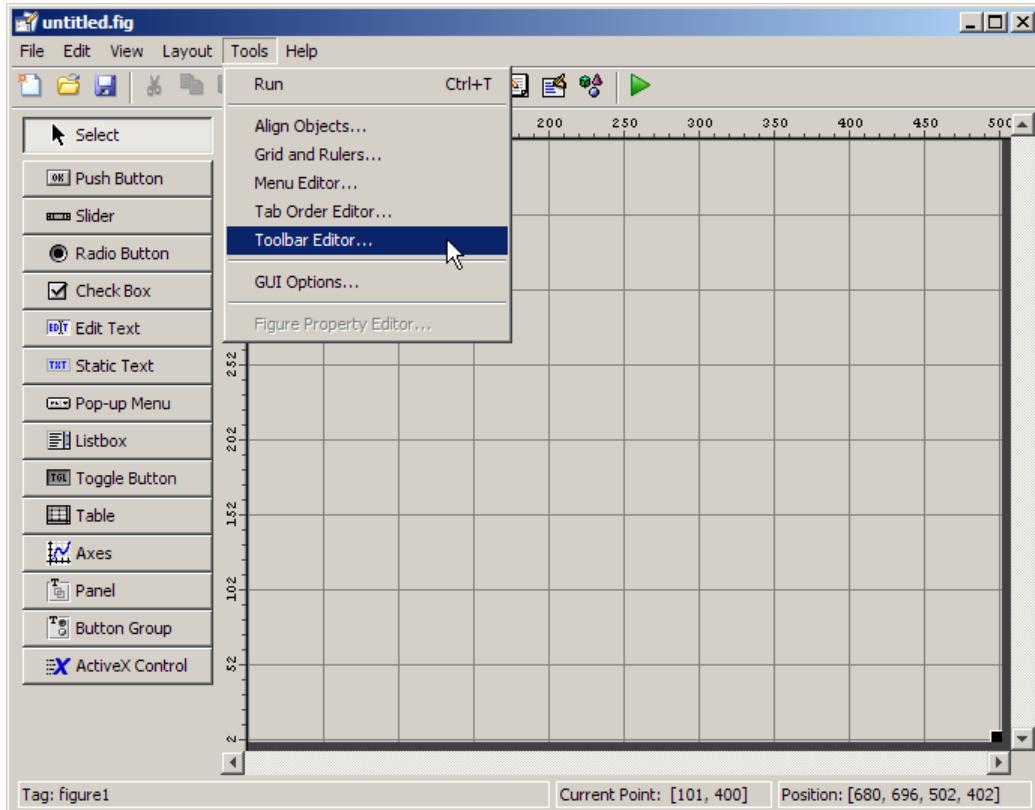
“ツール アイコンの編集” (p.6-129)

GUIDE を使用してツールバーを作成する

GUIDE でツールバー エディターを用いて、作成する GUI にツールバーを追加できます。ツールバー エディターは、GUIDE レイアウト エディターのツールバーから開きます。



ツールバー エディターは、[ツール] メニューから開くこともできます。



ツールバー エディターによって、関数 `uitoolbar`、`uipushtool`、`uitoggletool` のすべての機能に対話的にアクセスできます。ツールバー エディターは、GUIDE のコンテキスト内でのみ動作します。組み込まれた MATLAB ツールバーの修正には使用できません。ただし、ツールバー エディターを用いて、GUIDE で GUI のツールバーに追加、修正、削除を行うことができます。

GUIDE の GUI にツールバーを 1 つ追加できます。ユーザー GUI は、標準の MATLAB Figure ツールバーを含むこともできます。必要であれば、通常の Figure ツールバーのような外見をもつツールバーを作成できますが、(回転、ズーム、開くなど) ツールが固有の方法で動作するように、コールバックをカスタマイズできます。

メモ ユーザー GUI の Figure ツールバーを標準のものにする場合には、ツールバー エディターを利用する必要はありません。以下のように、これは Figure のToolBar プロパティを 'figure' に設定することで行えます。

- 1 GUIDE に GUI を開きます。
- 2 [表示] メニューから、[プロパティ インスペクター] を開きます。
- 3 ドロップダウン メニューを使用して、ToolBar プロパティを 'figure' に設定します。
- 4 Figure を保存します。

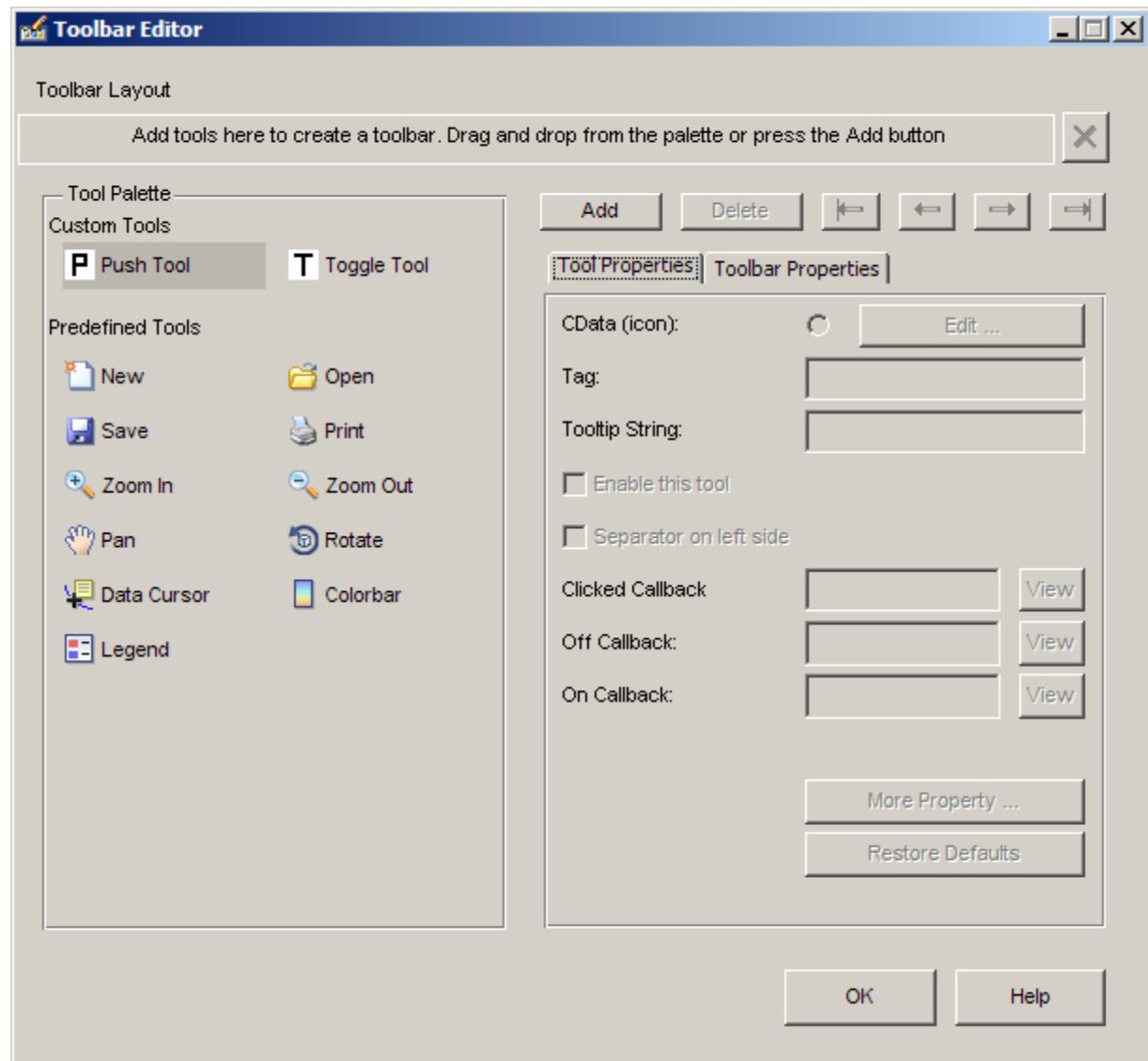
後で Figure ツールバーを削除する場合、ToolBar プロパティを 'auto' に設定し、GUI を再び保存します。これは、GUI にカスタム ツールバーがあると、カスタム ツールバーの削除や非表示は行いません。ツールバーを M コードを用いて作成する詳細は、“ツールバーの作成”(p.11-87)を参照してください。

MATLAB デスクトップでユーザーが GUI をドックしたり、アンドックしたりできるようにしたい場合には、GUI にツールバーまたはメニュー バーが必要になります。ツールバーやメニュー バーは、標準のもの、あるいは GUIDE で作成するものになります。さらに、Figure プロパティ DockControls はオンにします。詳細は、“メニューが Figure のドックにどのように影響するか”(p.6-101)を参照してください。

ツールバー エディターの利用

ツールバー エディターには、3 つの主な部分が含まれます。

- ・ 上部にある [ツールバー レイアウト] プレビュー エリア
- ・ 左にある [ツール パレット]
- ・ 右にある 2 つのタブをもつプロパティ ペイン



ツールを追加するには、[ツール パレット] から [ツールバー レイアウト] (はじめはテキストによる上記の指示を含む) にアイコンをドラッグして、[ツール プロパティ] ペインのツールのプロパティを編集します。

最初に GUI を作成する場合には、GUI にはツールバーはありません。ツールバー エディターを開いて最初のツールを配置するときにツールバーが作成され、直前に追加したツールのプレビューがウィンドウの上部に現れます。後で、ツールバーをもつ GUI を開くと、レイアウト エディターには表示されませんが、ツールバー エディターには既存のツールバーが表示されます。

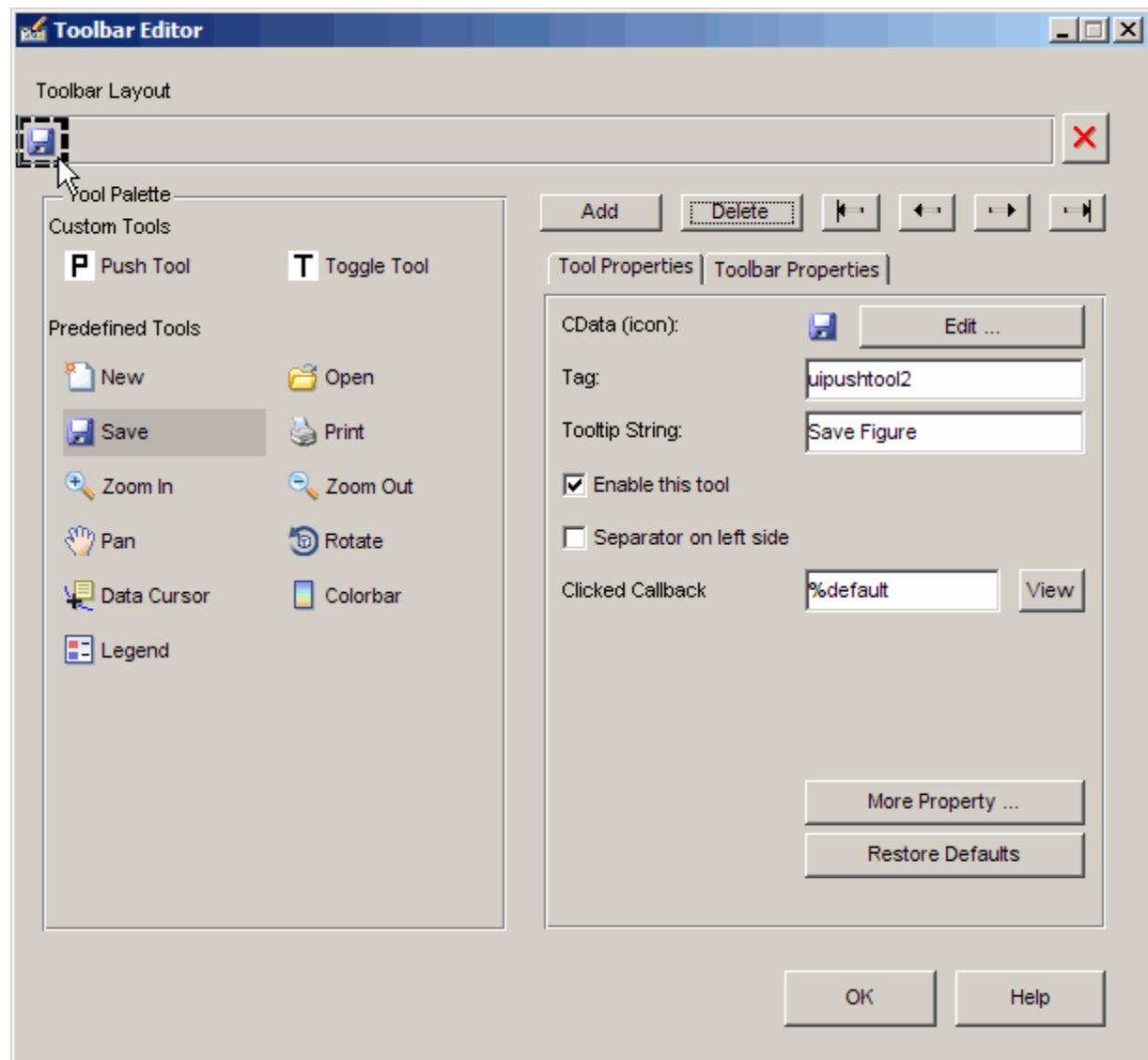
ツールの追加

以下の 3 とおりの方法でツールバーにツールを追加できます。

- ・ [ツール パレット] からツールをドラッグ アンド ドロップする。
- ・ パレットのツールを選択して、[追加] ボタンをクリックする。
- ・ パレットのツールをダブルクリックする。

ドラッグすると、ツールバーにツールを任意の順序で配置できます。その他の 2 つの方法では、[ツールバー レイアウト] の右端のツールの右にツールを配置します。新しいツールが選択され（周囲が点線で囲まれます）、そのプロパティは [ツール プロパティ] ペインに表示されます。ツールは一度に 1 つだけ選択できます。コンピューターのキーボード上のタブキーまたは矢印キーを用いて、[ツール パレット] を巡回できます。ツールバーに少なくとも 1 つのツールをすでに配置したことでしょう。

[ツール パレット] から [ツールバー レイアウト] エリアにツールを配置した後、以下の説明が示すように、現在選択したツールのプロパティがツールバー エディターに表示されます。



あらかじめ定義されたツールとカスタムツール
ツールエディターは、2つのタイプのツールを提供します。

- ・ 標準のアイコンと動作仕様をもつあらかじめ定義されたツール
- ・ 一般のアイコンをもち動作仕様がないカスタム ツール

定義済ツール. [ツール パレット] の下部のアイコンのセットは、標準の MATLAB Figure ツールを表します。これらの動作は組み込まれています。(回転やズームなど) Axes を必要とするあらかじめ定義されたツールは、Axes がない GUI では動作を示しません。あらかじめ定義されたツールの動作を定義するコールバックは、%default として示されています。これは、標準の Figure ツールバーとメニュー(ファイルのオープン、Figure の保存、モードの変更など)においてツールが呼び出す関数と同じ関数を呼び出します。他のいくつかのコールバックに対して %default を変更して、ツールをカスタマイズできます。GUIDE は、コールバック フィールドを変更するとき、あるいはその隣にある[表示] ボタンをクリックするときに、ツールの動作を修正するかどうかを警告し、続行するかどうかを尋ねます。

カスタム ツール. ツール パレットの上部の 2 つのアイコンは、プッシュ ツールとトグル ツールを作成します。これらには、オンやオフがクリックされたときの外見を管理する以外は、組み込まれた動作はありません。その結果、ツールバーにプッシュ ツールやトグル ツールを追加するときにユーザー独自のコールバックを提供する必要があります。カスタム ツールがクリックに応答するように、ユーザーが要求する仕様どおりの動作を作成するには、それらのコールバックを編集する必要があります。[ツール プロパティ] ペインのコールバックの隣にある[表示] ボタンをクリックして、エディター ウィンドウのコールバックを編集して、これを行います。

区切りの追加と削除

区切りは、ツールを動作させる垂直のバーで、ツールを視覚的にグループ化します。以下の 3 つの方法のいずれかで区切りの追加または削除をできます。

- ・ ツールのプレビューを右クリックして、ツールの区切りの on と off を切り替える [区切りを表示] を選択する。
- ・ ツールのプロパティ ペインで、左側のチェック ボックス [区切り記号] をチェック、またはクリアする。
- ・ プロパティ インスペクターからツールの Separator プロパティを変更する。

区切りを追加した後、[ツールバー レイアウト] のツールの左側に区切りが表示されます。区切りは、独立したオブジェクトまたはアイコンではありません。これは、ツールのプロパティです。

ツールの移動

2 通りの方法で、ツールバーのツールを並べ替えることができます。

- ・ ツールを新しい位置にドラッグする。
- ・ ツールバーでツールを選択し、ツールバーの右にある下の矢印ボタンの 1 つをクリックする。

ツールの左に区切りがある場合、区切りはツールと共に移動します。

ツールの削除

3 通りの方法で、ツールバーからツールを削除することができます。

- ・ ツールを選択して、Delete キーを押す。
- ・ ツールを選択して、GUI の [削除] ボタンをクリックする。
- ・ ツールを右クリックして、コンテキストメニューから [削除] を選択する。

これらのアクションは、元に戻すことができません。

ツールのプロパティの編集

[ツール プロパティ] ペインを用いて、現在選択しているツールの外見と動作を編集します。これは、最も一般的に使用されるツール プロパティを設定するためのコントロールを含みます。

- ・ CData – ツールのアイコン
- ・ タグ – ツールの内部的な名前
- ・ このツールを有効にする – ユーザーがツールをクリックできるかどうか
- ・ 左側に区切り – アイコンの左にある、ツールを区切りグループ化するためのバー
- ・ クリック時のコールバック – ユーザーがツールをクリックするときに呼び出される関数
- ・ オフ時のコールバック (uitoggletool のみ) – ツールが *off* の状態に置かれた場合に呼び出される関数
- ・ オン時のコールバック (uitoggletool のみ) – ツールが *on* の状態に置かれた場合に呼び出される関数

ツールのコールバックのプログラミングの詳細は、“コールバック:概要”(p.8-2)を参照してください。選択されたツールのこれらのプロパティや他のプロパティには、プロパティインスペクターを用いてアクセスすることもできます。プロパティインスペクターを開くには、[ツール プロパティ]ペインの[その他のプロパティ]ボタンをクリックします。

ツール アイコンの編集

選択されたツールバーのアイコンを編集するには、[ツール プロパティ]ペインの[CData(アイコン)]の隣の[編集]ボタンをクリックします。あるいは、[ツールバー レイアウト]を右クリックし、コンテキストメニューから[アイコン編集]を選択します。アイコンエディターは、ツールに読み込まれたツールの CData と共に開きます。アイコンの編集の詳細は、“アイコンエディターの利用”(p.6-131)を参照してください。

ツールバー プロパティの編集

ツールバーの空の部分、または[ツールバー プロパティ]タブをクリックすると、そのプロパティのうちの以下の 2 つを編集できます。

- Tag – ツールバーの内部的な名前
- Visible – ツールバーがユーザー GUI に表示されるかどうか

Tag プロパティは、はじめは uitoolbar1 に設定されます。Visible プロパティは、on に設定されます。on である場合、Visible プロパティは、Figure の Toolbar プロパティの設定によらず、GUI にツールバーを表示します。ユーザーが([表示]メニューから)ツールバーを組み込めるようにカスタムツールバーを切替える場合、ユーザーは Visible プロパティをコントロールするために、メニュー項目、チェックボックス、または他のコントロールを作成できます。

プロパティインスペクターのツールバーに対するほとんどのプロパティにアクセスするには、[その他のプロパティ]をクリックします。

ユーザーのツールバーの検証

ツールバーを試すには、レイアウトエディターの[実行]ボタンをクリックします。MATLAB は、.fig ファイルに変更を保存するかどうかをはじめに尋ねます。

ツールバーの削除

ツールバーは完全に削除できます。ツールバー エディターからツールバーを無効にし、GUI を(既定では表示されない、Figure ツールバー以外の)ツールバーがない状態にします。ツールバーを削除する方法は 2 とおりあります。

- ・ ツールバーの右端で、[削除]  ボタンをクリックする。
- ・ ツールバーの空のエリアで右クリックし、コンテキストメニューから [ツールバーを削除] を選択する。

ツールバーを削除せずに、“ツールの削除”(p.6-127)に示された方法で個々のツールすべてを削除する場合、ユーザー GUI には、空のツールバーが含まれます。

ツールバー エディターを閉じる

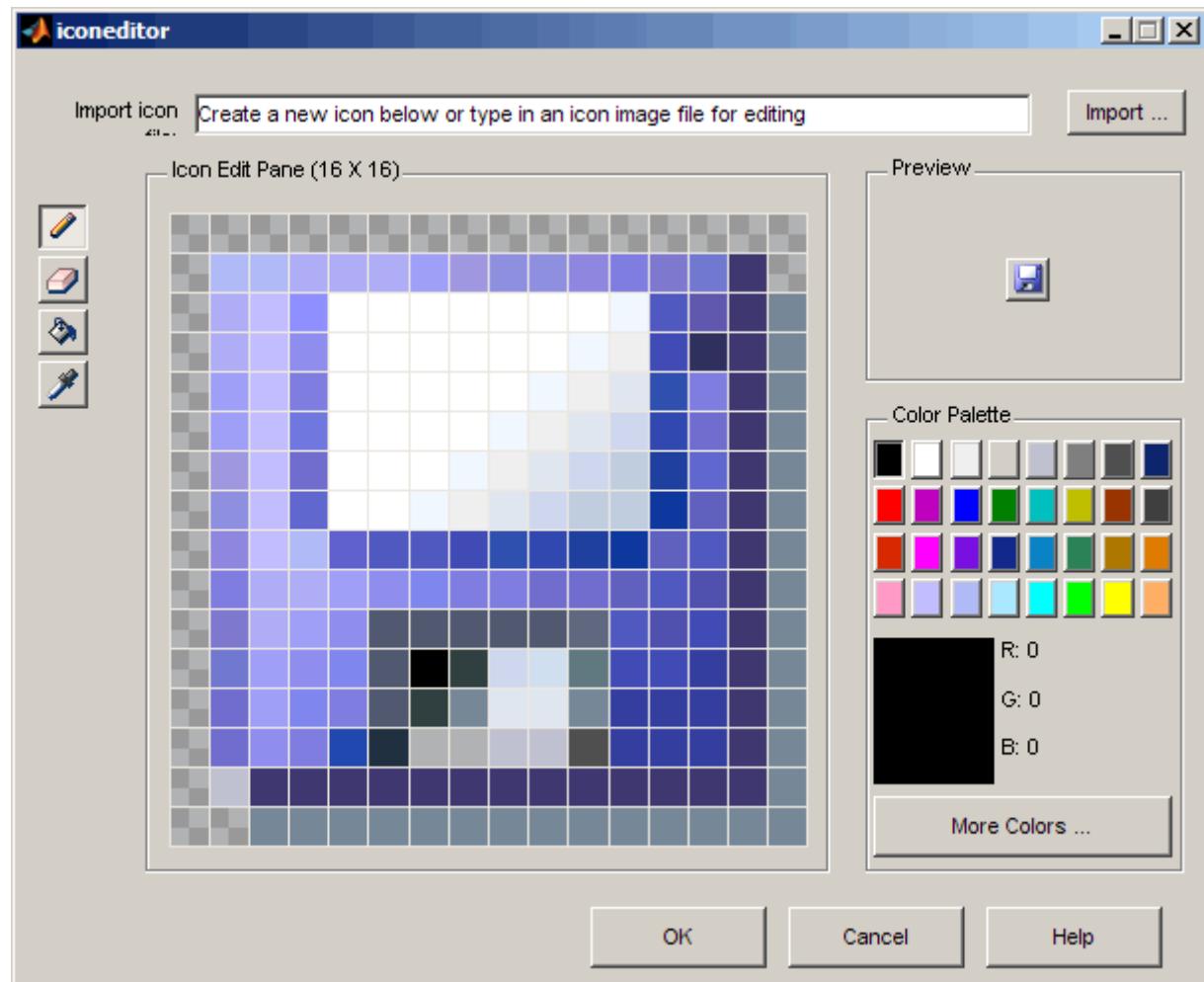
2 とおりの方法で、ツールバー エディターを閉じることができます。

- ・ [OK] ボタンを押す。
- ・ タイトルバーの [閉じる] ボックスをクリックする。

ツールバー エディターを閉じるとき、編集中の GUI と共に、ツールバーの現在の状態が保存されます。レイアウトエディターにはツールバーが表示されません。GUI を表示したり使用するには、実行する必要があります。

ツール アイコンの編集

GUIDE には、アイコン(ツールバー上のアイコンなど)を作成したり修正するための GUI である、独自のアイコン エディターを含みます。このエディターは、ツールバー エディターからのみアクセスできます。次の図は、標準の [保存] アイコンを用いて読み込まれたアイコン エディターを示します。



メモ ユーザー独自のアイコンエディターの作成方法を示す例があります。“アイコンエディター”(p.15-60)の例と、GUI Building ドキュメンテーションの「プログラミングによる GUI の作成」における、「複数の GUI でのデータの共有」の説明を参照してください。

アイコン エディターの利用

アイコン エディターの GUI は、以下のコンポーネントを含みます。

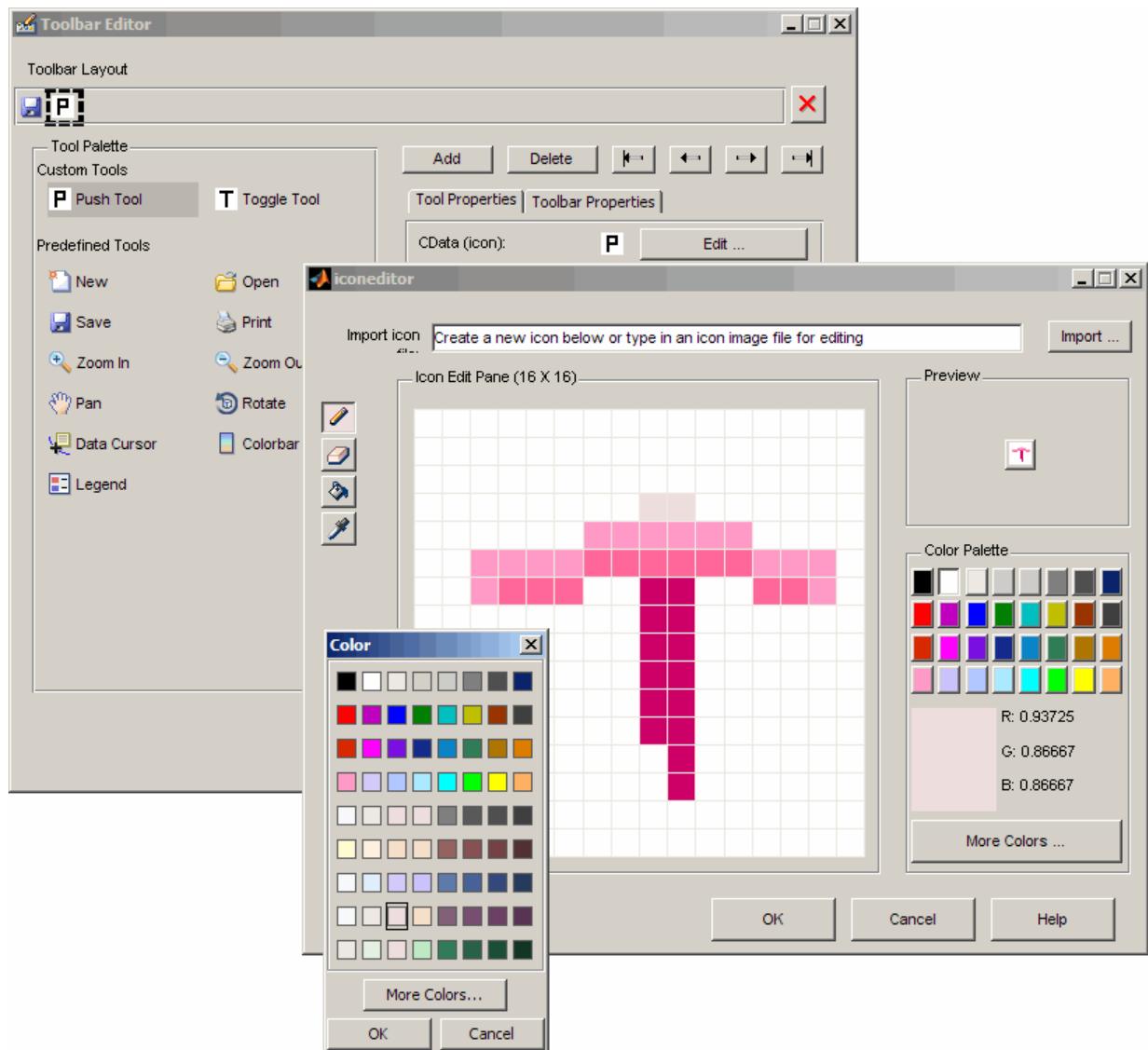
- ・ [アイコン ファイルの名前] – 編集のために読み込まれたアイコン イメージ ファイル
- ・ [インポート] ボタン – 編集のために既存のアイコン ファイルを選択するファイル ダイアログを開きます。
- ・ 描画ツール – 編集のアイコンのために左側にある 4 つのツールのグループ
 - 鉛筆ツール – クリックまたはドラッグによるカラー アイコン ピクセル
 - 色を消すツール – クリックまたはドラッグしてピクセルを消去して透明にします。
 - 色で塗りつぶすツール – 領域を現在のカラーと同じカラー ピクセルにします。
 - 色の選択ツール – 現在のカラーを定義するピクセルまたはカラー パレット見本をクリックします。
- ・ [アイコン編集] ペイン – アイコンの色を決める $n \times m$ グリッド
- ・ [プレビュー] ペイン – アイコンの現在の状態のプレビューをもつボタン
- ・ [カラー パレット] – 鉛筆とペイントツールが使用できる色の見本
- ・ [その他の色] ボタン – 色の選択と定義のための [色] ダイアログ ボックスを開きます。
- ・ [OK] ボタン – GUI を取り消し、その現在の状態にアイコンを返します。
- ・ [キャンセル] ボタン – アイコンを戻さずに GUI を閉じます。

アイコン エディターを扱うには、次のようにします。

- 1 選択されたツールのアイコンに対してアイコン エディターを開きます。
- 2 鉛筆ツールを用いて、グリッド内の四角を色付けします。
 - ・ パレットのカラー セルをクリックします。
 - ・ その色が [カラー パレット] プレビューの見本に表示されます。
 - ・ グリッドの特定の四角をクリックして、これらの四角に選択した色を移します。
 - ・ 左マウス ボタンを押したままマウスをグリッド上にドラッグして、触れている四角に選択した色を移します。

- ・ 他の色で上書きして色を変更します。
- 3 色を消すツールを用いて、いくつかの四角の色を消します。
- ・ パレットの消しゴムのボタンをクリックします。
 - ・ これらの四角を消すために特定の四角をクリックします。
 - ・ マウスをクリックしながらドラッグして、触れている四角を消します。
 - ・ 他の描画ツールをクリックして消しゴムを無効にします。
- 4 [OK] をクリックして GUI を閉じ、作成したアイコンを返します。あるいは、[キャンセル] をクリックして、選択されたツールのアイコンを修正せずに GUI を閉じます。

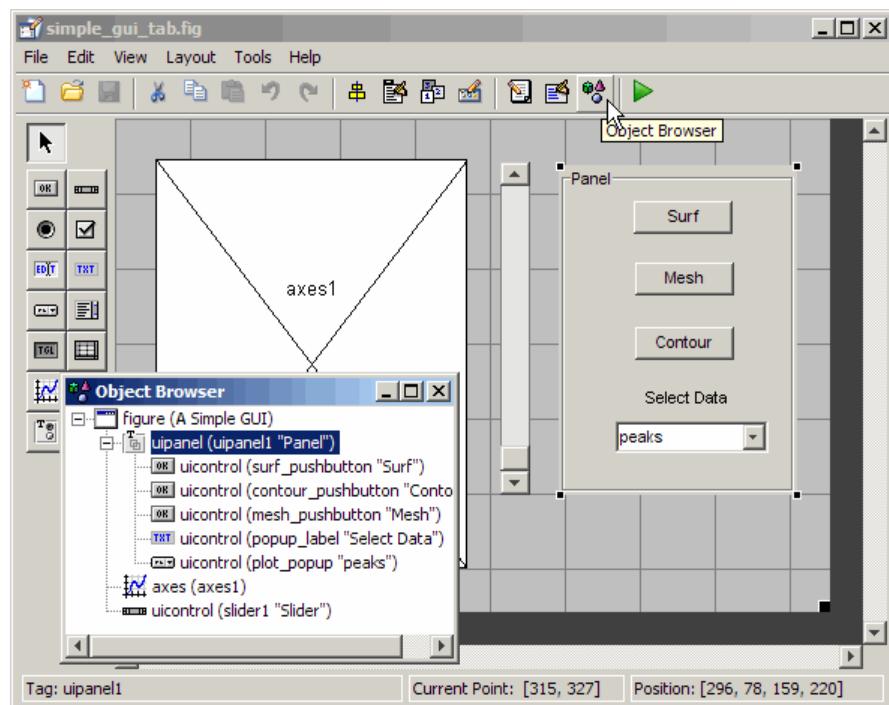
uipush tool アイコンを保存する前に、3 つの GUI が共に動作している様子が示されています。



オブジェクト階層の表示

オブジェクトブラウザは、Figure内のオブジェクトを階層的なリストとして表示します。特にメニュー、ペイン、あるいはボタングループを含むGUIをレイアウトするとき、オブジェクトの階層を定期的にチェックします。[表示] > [オブジェクトブラウザ]  から、あるいはGUIDEツールバーでオブジェクトブラウザアイコンをクリックしてオブジェクトブラウザを開きます。

次の図は、Figureオブジェクトとその子オブジェクトを示します。これは、uipanelの子のオブジェクトも表示します。



オブジェクトブラウザの中でコンポーネントの位置を確認するためには、レイアウトエディターでそのコンポーネントを選択してください。オブジェクトブラウザでも、自動的にそのコンポーネントが選択されます。同様に、オブジェクトブラウザでオブジェクトを選択すると、レイアウトエディターでも自動的にそのコンポーネントが選択されます。

クロスプラットフォーム互換性のための設計

このセクションの内容…

“既定のシステム フォント” (p.6-135)

“標準背景色” (p.6-136)

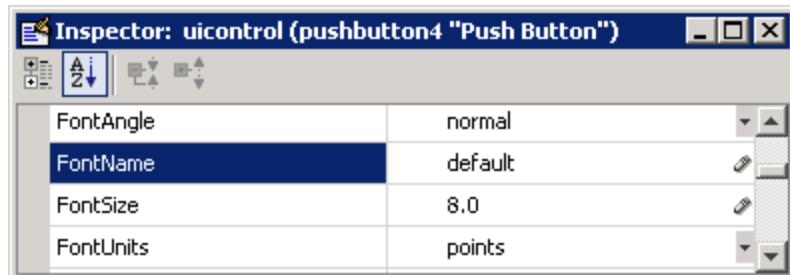
“クロスプラットフォーム互換の Units” (p.6-137)

既定のシステム フォント

uicontrol は既定で、使用しているプラットフォームの既定のフォントを使用します。たとえば、PC 上でユーザー GUI を表示するとき、uicontrol は MS San Serif を使用します。ユーザー GUI が、異なるプラットフォーム上で動作するとき、コンピューターの既定のフォントを使用します。このため、ユーザー GUI と他のアプリケーション GUI に関する外見を与えます。

FontName プロパティを名前の付いたフォントに設定した後、既定値に戻したい場合、プロパティを文字列 default に設定できます。そうすると、MATLAB は実行時にシステム既定値を使用します。

このプロパティを設定するために、プロパティ インスペクターを使用することができます。



あるいは、set コマンドを使用して、GUI M ファイルにプロパティを設定できます。たとえば、ユーザー GUI にプッシュ ボタンが存在する場合、そのハンドルは、handles 構造体の pushbutton1 フィールドに保存され、その後、次のステートメント、

```
set(handles.pushbutton1, 'FontName', 'default')
```

は、FontName プロパティにシステム既定値を使用するように設定します。

固定幅フォントの指定

uicontrol に固定幅フォントを使用したい場合、その FontName プロパティを文字列 fixedwidth に設定してください。この特別な識別子により、ターゲット プラットフォームに対して、ユーザー GUI が標準の固定幅を使用することを保証します。

ルートの FixedWidthFontName プロパティを調べることにより、与えられたプラットフォーム上使用される固定幅フォント名を検出することができます。

```
get(0,'FixedWidthFontName')
```

特定のフォント名の使用

FontName プロパティに対して、(Times あるいは Courier のような) 実際のフォント名を指定することができます。しかし、そのように指定すると、ユーザー GUI が異なるコンピューター上で動くとき、期待通りの外見にならない可能性があります。ターゲットコンピューターが指定のフォントをもたない場合、ユーザー GUI が良く見えなかったり、そのシステム上の GUI に使用される標準フォントではない、他のフォントに置き換えられることもあります。また、同じ名前でフォントの異なるバージョンのものは、与えられた文字の組に対して、サイズ要求が異なることもあります。

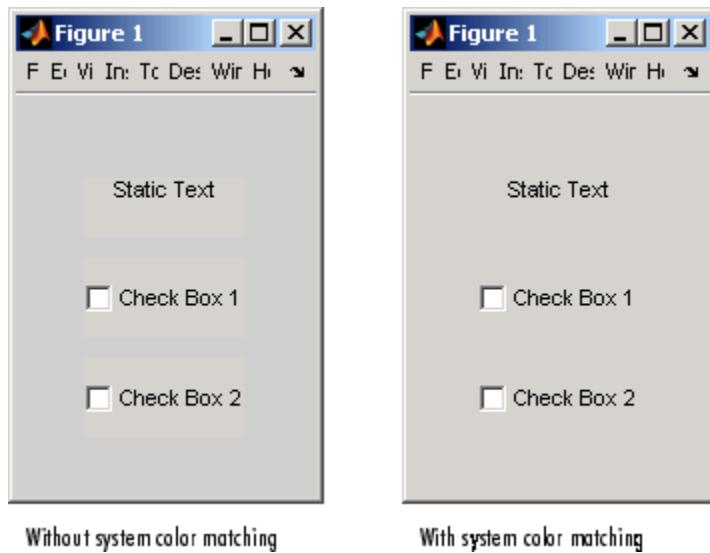
標準背景色

既定のコンポーネント背景色は GUI が実行しているシステムの標準のシステム背景色です。この色はコンピューター システムが異なると変わります。たとえば、PC の標準的なグレーの影は UNIX システムのものとは異なり、既定の GUI 背景色とは一致しないことがあります。

既定のコンポーネントの背景色を使用すると、GUI の背景色と同じ色を使用できます。このため、ユーザー GUI と他のアプリケーション GUI に関する外見を共有します。GUIDE でこれを行うには、レイアウトエディターの [ツール] メニューで [GUIDE オプション] > [背景にシステム配色を使用 (推奨)] をチェックします。

メモ このオプションは最初に [FIG-ファイルと M ファイルの作成] オプションを選択した場合に限り、利用できます。

次の図は、システム カラー マッチングがある場合とない場合の結果を示します。



クロスプラットフォーム互換の Units

クロスプラットフォーム互換の GUI は、スクリーン サイズと解像度が異なるコンピューター上で、正しく見える必要があります。ピクセルのサイズは異なるコンピューター上で変わることがあるので、Figure の Units の既定値である pixels を使用すると、すべてのプラットフォーム上で同じに見える GUI を生成することはできません。

このため、GUIDE は既定値で Figure に対する Units プロパティを characters にします。

システムに依存したユニット

Character unit は、既定値のシステム フォントからの文字によって定義されています。Character unit の幅は、システム フォントにおいて文字 x の幅に等しくなります。Character unit の高さは、テキストの 2 行のベースライン間の距離です。Character unit が四角でないことに注意してください。

Units とサイズ変更の動作

GUI のサイズ変更動作を GUI オプション ダイアログ ボックスから設定すると、GUIDE はプラットフォームが変わっても意図した外見や特徴を保つように GUI のコンポーネントに対する Units を自動的に設定します。サイズ変更動作のオプションを指定するには、[ツール] メニューから [GUI オプション] を選択してから、[リサイズ不可]、[比例 (プロポーショナル)] または [その他 (ResizeFcnを使用)] を選択して [リサイズ アクション] を指定します。

[リサイズ不可] オプションを選択すると、GUIDE は既定でコンポーネントの単位を characters にします。[比例 (プロポーショナル)] を選択すると、コンポーネントの単位を既定値の normalized にします。いずれの場合も、これらの設定によって、GUI は自動的に、GUI が異なるコンピューター上に表示されるときのサイズとコンポーネントの相対的な間隔を調整します。

[その他 (ResizeFcn を使用)] を選択すると、GUIDE は既定でコンポーネントの単位を characters にします。ただし、GUI のサイズ変更動作をカスタマイズするために、ResizeFcn コールバックを与える必要があります。

メモ Figure の Resize プロパティをプログラミングで、あるいはプロパティ インスペクターで修正する場合、GUIDE はコンポーネントの単位を自動的には調整しません。

GUI をレイアウトする場合、より一般的な単位、たとえば、インチまたはセンチメートルを使用すると便利な場合があります。しかし、異なるコンピューター上で GUI の外見を保つには、忘れずに Figure の Units プロパティを characters に変更し、GUI を保存する前に、コンポーネントの Units プロパティを characters (サイズ変更不可の GUI) あるいは normalized (サイズ変更可能な GUI) に変更します。

GUIDE GUI の保存と実行

- ・ “GUIDE GUI とそのファイルの名前付け” (p.7-2)
- ・ “GUIDE GUI の保存” (p.7-4)
- ・ “GUIDE GUI の実行” (p.7-10)

GUI とそのファイルの名前付け

このセクションの内容…

“GUI ファイル” (p.7-2)

“ファイルと GUI の名前” (p.7-3)

“GUI と GUI ファイルの名前変更” (p.7-3)

GUI ファイル

既定で、GUIDE は GUI を 2 つのファイルに格納します。これらのファイルは、GUI を保存、あるいは実行するときにはじめて作成されます。

- ・ 拡張子 .fig をもつ FIG-ファイルは、GUI レイアウトや、プッシュ ボタン、座標軸、パネル、メニューなどの GUI コンポーネントの完全な描写情報を含んでいます。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、修正できません。FIG-ファイルは MAT-ファイルのようなものであることに注意してください。
- ・ 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ます。

これら 2 つのファイルは、同じ名前を持ち、通常、同じフォルダーに存在します。これらは、GUI のレイアウトとプログラミングの作業に対応します。レイアウト エディターに GUI をレイアウトすると、ユーザーの作業は FIG-ファイルに保存されます。GUI をプログラミングする場合、ユーザーの作業は対応する M ファイルに格納されます。

メモ GUIDE で作成される GUI M ファイルは、ユーザーがそれを読み込み GUI を操作するときに、通常 FIG-ファイルが呼び出す関数を含みます。これらはスクリプト(実行可能であっても関数を定義しない MATLAB コマンドの列)にはなりません。

GUI が複数の ActiveX コンポーネントを含む場合、GUIDE は、各 ActiveX コンポーネントに対するファイルも生成することに注意してください。詳細は、“ActiveX コントロール” (p.8-48)を参照してください。

これらのファイルについての詳細は、“GUI ファイルの概要” (p.8-7)を参照してください。

ファイルと GUI の名前

GUI を定義する M ファイルと FIG-ファイルは、同じ名前をもたなければなりません。また、この名前は、GUI の名前でなければなりません。

たとえば、ファイルの名前が `mygui.fig` と `mygui.m` である場合、GUI の名前は `mygui` で、コマンド ラインで `mygui` とタイプすることによって GUI を実行できます。ただし、M ファイルと FIG-ファイルが同じフォルダーにあること、そのフォルダーがパス内にあることが前提になります。

ユーザーが最初に GUI を保存するときに名前が割り当てられます。GUI の保存についての詳細は、“GUI を保存する方法”(p.7-4)を参照してください。

GUI と GUI ファイルの名前変更

GUI の名前を変更するには、レイアウト エディターの [ファイル] メニューから [別名で保存] を用いて GUI FIG-ファイルの名前を変更します。これを行う場合、GUIDE は FIG-ファイルと GUI M ファイルの両方の名前を変更し、旧い名前を含むコード バック プロパティを新しい名前に更新し、M ファイル内のファイル名のすべてのインスタンスを更新します。GUIDE から GUI を保存する方法の詳細は、“GUI の保存”(p.7-4)を参照してください。

メモ GUI が適切に機能しなくなるので、GUIDE の外部では GUI ファイルの名前を変更しないでください。

GUI の保存

このセクションの内容…

“GUI を保存する方法”(p.7-4)

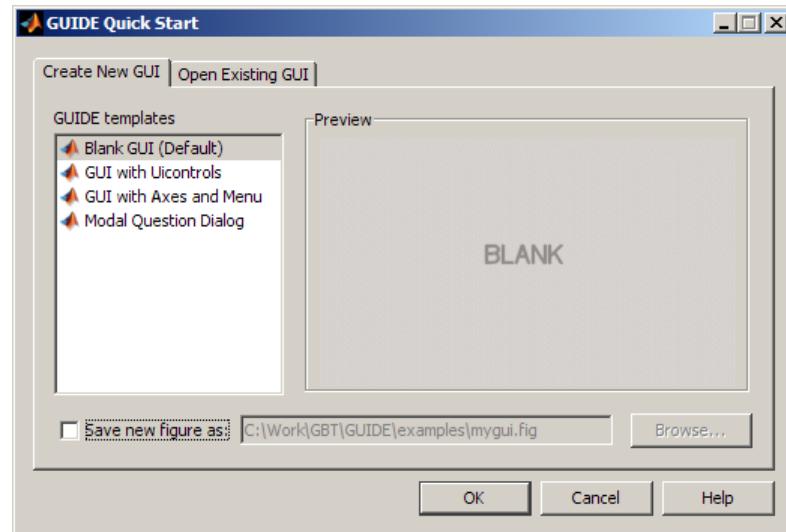
“新規の GUI の保存”(p.7-5)

“既存の GUI の保存”(p.7-8)

GUI を保存する方法

以下のいずれかの方法で、GUIDE で GUI を保存できます。

- GUIDE のクイック スタート ダイアログ ボックスから保存する方法。テンプレートを選択する前に、GUIDE によって GUI の名前を選択できます。[OK] をクリックすると、GUIDE はユーザーが指定する名前で GUI M ファイルと FIG ファイルを保存します。



- ファイルを最初に保存するときは、次のようにします。
 - レイアウト エディター ツールバー上で [保存] アイコン をクリック
 - [ファイル] メニューの [保存] または [別名で保存] オプションを選択

いずれの場合も、GUIDE は GUI を保存する前にユーザーに名前を入力するよう
に促し、たとえば、`mygui.fig` と `mygui.m` のような指定した名前で `.fig` ファイル
と `.m` ファイルを保存します。

- ・ GUI を最初に実行するとき
 - － レイアウトエディター ツールバー上で [実行] アイコン をクリック
 - － [ツール] メニューから [実行] を選択

いずれの場合も、GUIDE は GUI をアクティブにする前に GUI ファイルに名前を付
けて保存するようにプロンプトを表示します。

すべての場合に、GUIDE はテンプレート M ファイルを作成し、MATLAB エディター
に開きます。テンプレート M ファイルについての詳細は、“コールバック関数の名
前付け”(p.8-16)を参照してください。

メモ ほとんどの場合、現在のフォルダー、あるいはユーザー パスに GUI を保存する
必要があります。GUIDE が生成した GUI は、プライベート フォルダーからは正確
には実行できません。MATLAB 7.0 またはそれ以降のバージョンの MATLAB で
作成されたり、修正された GUI FIG-ファイルは、バージョン 6.5 およびそれ以前の
バージョンのものと自動的には互換性がありません。MAT-ファイルの一種である、
後方互換の FIG-ファイルを作成するには、ファイルを保存する前に、MATLAB [設
定] ダイアログ ボックスで、[一般] > [MAT-ファイル] > [MATLAB Version 5 以降
(-v6 を保存)] をチェックする必要があります。ボタン グループ、パネル、テーブル
は MATLAB 7 で導入されたので、以前のバージョンの MATLAB で実行することが
見込まれる GUI ではそれらを使用しないでください。

[-v6] オプションは廃止され、将来のバージョンの MATLAB では削除されることに
注意してください。

新規の GUI の保存

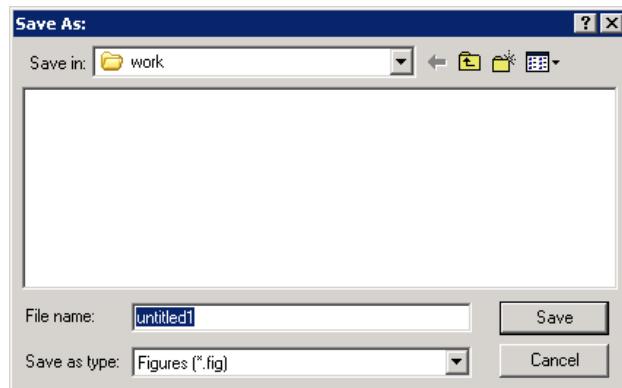
GUI を初めて保存する場合、あるいは [ファイル] メニューから [別名で保存] を使用
している場合には、以下の手順に従います。

メモ [ファイル] メニューから [別名で保存] を選択するか、あるいは、ツールバー上の [保存] ボタンをクリックすると、GUIDE は GUI をアクティブにせずに保存します。ただし、[ツール] メニューから [実行] を選択するか、あるいはツールバー上の [実行] アイコン▶をクリックすると、GUIDE は GUI をアクティブにする前に保存します。

- 1 GUI を変更したり、[ツール] メニューから [実行] を選択するか、あるいはツールバーの [実行] アイコン▶をクリックして GUI をアクティブにすると、GUIDE は以下のダイアログ ボックスを表示します。[Yes] をクリックして続行します。

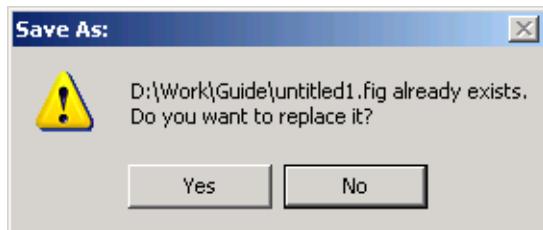


- 2 前の手順で [はい] をクリックしたり、GUI をアクティブにせずに保存したり、あるいはファイル メニューから [別名で保存] を選択した場合、GUIDE は [別名で保存] ダイアログ ボックスを開き、FIG-ファイル名を入力するように指示します。



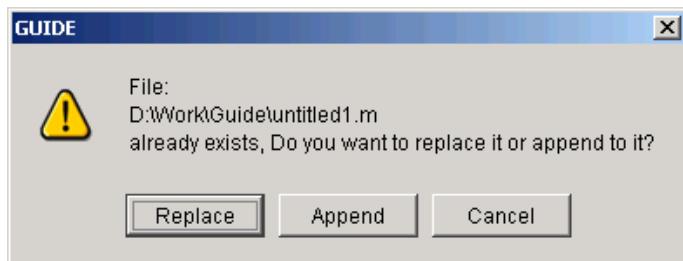
- 3 保存フォルダーを選択する場合、フォルダーを変更してから FIG-ファイルに使用したい名前を入力します。書き込み可能なフォルダーを選択するようにしてください。GUIDE はこの名前を用いて FIG-ファイルと M ファイルの両方を保存します。

- 4 既存のファイル名を選択すると、GUIDE は既存の FIG-ファイルを置き換えるかどうか尋ねるダイアログ ボックスを表示します。[はい] をクリックして続行します。

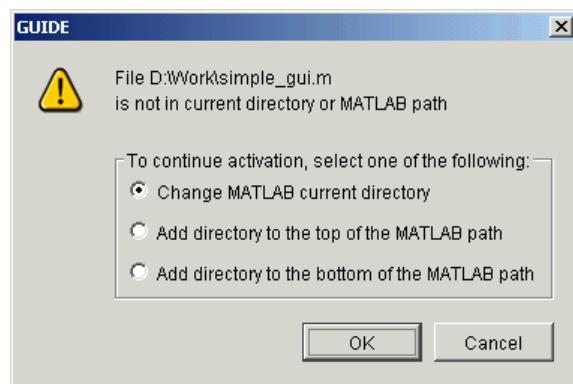


- 5 前の手順で [はい] を選択すると、GUIDE は 既存の M ファイルを置き換えるいか、あるいは追加するかを尋ねるダイアログ ボックスを表示します。一般には、[置換] を選択します。

[追加] を選択すると、GUIDE は、既存の M ファイルには存在せずに現在のレイアウトに存在するコンポーネントに対するコールバックを、既存の M ファイルに追加します。新しいコンポーネントを追加する前に、それらの Tag プロパティが既存の M ファイル内のコールバック関数に現れる Tag の値と重複しないことを確かめてください。Tag プロパティの指定についての詳細は、“各コンポーネントへの識別子の割り当て”(p.6-37)を参照してください。callback 関数の名前についての詳細は、“コールバック関数の名前付け”(p.8-16)を参照してください。



- 6 [ツール] メニューから [実行] を選択するか、またはツールバー上の [実行] ボタンをクリックして GUI をアクティブにするように選択する場合、GUI を保存するフォルダーが MATLAB パス上にないと、GUIDE は、現在の作業フォルダーを GUI ファイルを含むフォルダーに変更するオプションを与えたり、あるいは、そのフォルダーを MATLAB パスの最も上または下に追加して、ダイアログ ボックスを開きます。



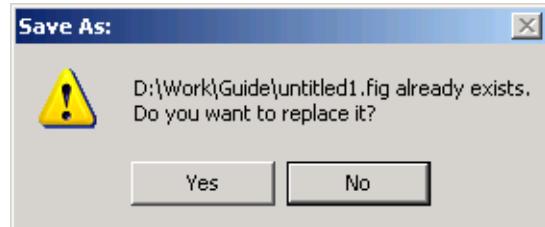
- 7 ファイルを保存後、GUIDE は MATLAB エディターに GUI M ファイルを開きます。GUI を実行しようとすると、GUI もアクティブにします。

既存の GUI の保存

現在の位置に既存の GUI を保存するときは、次の手順に従います。[ファイル] メニューから [別名で保存] を使用している場合、“新規の GUI の保存”(p.7-5)を参照してください。

GUI を変更したり、[ツール] メニューから [実行] を選択するか、あるいはツールバーの [実行] ボタンをクリックして GUI を保存してアクティブにすると、GUIDE は GUI を保存してからアクティブにします。これは、ユーザーが新しいコンポーネントを追加した場合でも M ファイルを自動的には開きません。

[ファイル] メニューから [保存] を選択するか、あるいはツールバー上の [保存] ボタン をクリックすると、GUIDE は GUI をアクティブにせずに保存します。



GUI の実行

このセクションの内容…

“M ファイルの実行” (p.7-10)

“GUIDE レイアウト エディターから” (p.7-10)

“コマンド ラインから” (p.7-11)

“M ファイルから” (p.7-11)

M ファイルの実行

一般に、ユーザーは、GUIDE が生成する M ファイルを実行することによって、ユーザー GUI を起動します。この M ファイルは、GUI を読み込むコマンドを含み、コンポーネント コールバックに対するフレームワークを提供します。M ファイルについての詳細は、“GUI ファイルの概要” (p.8-7) を参照してください。

M ファイルを実行すると、完全な機能を持つ GUI がスクリーン上に表示されます。以下の節に述べるように、3 つの方法で GUI を実行できます。

メモ GUI の Figure を表示するには、現在のフォルダーブラウザーでそのファイル名をダブルクリックします。openfig、open、hgload を実行して表示することもできます。これらの関数は FIG-ファイルを MATLAB ワークスペースに読み込み、Figure を開いて表示します。表示されている Figure が GUIDE GUI である場合、そのコンポーネントを操作できますが、何も起こりません。これは、GUI を初期化したり、コンポーネント コールバックを実行したりするために、対応する M ファイルが実行されていないためです。

GUIDE レイアウト エディターから

GUIDE レイアウト エディターから、以下の方法で GUI を実行できます。

- ・ レイアウト エディター ツールバー上で▶ボタンをクリック
- ・ [ツール] メニューから [実行] を選択

いずれの場合も、GUI が変更されてたり一度も保存されていない場合には、GUIDE は GUI をアクティブにする前に GUI ファイルを保存し、GUI M ファイルを MATLAB エ

ディターに開きます。この過程の詳細は、“GUI の保存”(p.7-4)を参照してください。GUI M ファイルについての詳細は、“GUI ファイルの概要”(p.8-7)を参照してください。

コマンド ラインから

GUI M ファイルを実行することによって、この M ファイルから GUI を実行します。たとえば、GUI M ファイルが mygui.m の場合、コマンド ラインで次のように入力します。

```
mygui
```

FIG ファイルと M ファイルは、ユーザー パスまたは現在のフォルダーにあります。開いたときに GUI を非表示にしたい場合は、次のように入力します。

```
mygui('Visible','off')
```

GUI が実行時に引数を受け入れる場合、それらは GUI の opening 関数に渡されます。詳細は、“Opening 関数”(p.8-25)を参照してください。

メモ 2つ以上の GUI のコピーを同時にアクティブにしたいかどうか考えてください。1つの GUI のみをアクティブにしたい場合、レイアウト エディターの [表示] メニューから [オプション] > [GUI は 1 つのインスタンスのみの実行を許可 (シングルトン)] を選択します。詳細は、“GUI オプション”(p.5-9)を参照してください。

M ファイルから

GUI M ファイルを実行することによって、M ファイルから GUI を実行します。たとえば、GUI M ファイルが mygui.m である場合、M ファイル スクリプトに次のステートメントを含めてください。

```
mygui
```

開いたときに GUI を非表示にしたい場合は、次のステートメントを使用します。

```
mygui('Visible','off')
```

M ファイルは、GUI が実行している MATLAB パスまたは現在の MATLAB フォルダになければなりません。

GUI が実行時に引数をアクセプトする場合、それらは GUI の opening 関数に渡されます。詳細は、“Opening 関数”(p.8-25)を参照してください。

メモ 2つ以上の GUI のコピーを同時にアクティブにしたいかどうか考えてください。1つの GUI のみをアクティブにしたい場合、レイアウトエディター [表示] メニューから [オプション] を選択し、[GUI は1つのインスタンスのみの実行を許可 (シングルトン)] を選択します。詳細は、“GUI オプション” (p.5-9)を参照してください。

GUIDE GUI のプログラミング

- ・ “コールバック:概要” (p.8-2)
- ・ “GUI ファイルの概要” (p.8-7)
- ・ “コールバックとコンポーネントとの関連付け” (p.8-11)
- ・ “コールバック構文と引数” (p.8-15)
- ・ “コールバックの初期化” (p.8-25)
- ・ “例:GUIDE GUI コンポーネントのプログラミング” (p.8-30)

コールバック:概要

このセクションの内容…

“GUIDE を利用して作成する GUI のプログラミング” (p.8-2)

“コールバックとは” (p.8-2)

“コールバックの種類” (p.8-2)

GUIDE を利用して作成する GUI のプログラミング

GUI をレイアウトした後、その動作をプログラムします。記述するコードは、GUI がイベントにどのように応答するかをコントロールします。イベントには、ボタン クリック、スライダー移動、メニュー項目の選択、コンポーネントの作成と削除などがあります。このプログラミングは、GUI の Figure 自身と各コンポーネントに対する、コールバックと呼ばれる関数のセットの形式をとります。

コールバックとは

コールバックは、ユーザーが 特定の GUI コンポーネントあるいは GUI の Figure と関連させる関数です。コールバックは、コンポーネントに対するイベントに応答して何らかのアクションを行うことで、そのコンポーネントの動作をコントロールします。このプログラミング方法は、イベントドリブン プログラミングと呼ばれることがあります。

コンポーネントに対してイベントが起こると、そのイベントがトリガーするコンポーネントのコールバックが MATLAB により呼び出されます。例として、あるデータのプロットをトリガーするボタンをもつ GUI を考えます。GUI ユーザーがボタンをクリックすると、MATLAB は、そのボタンをクリックすることに関するコールバックを呼び出します。すると、ユーザーがプログラムしたコールバックは、データを取得してプロットします。

コンポーネントは、プッシュ ボタン、リスト ボックス、スライダーなどのコントロール デバイスです。プログラミングでは、コンポーネントは、パネルやボタン グループなどのコンテナー、あるいはメニューにもなります。コンポーネントのリストや説明として、“利用可能なコンポーネント” (p.6-20)を参照してください。

コールバックの種類

GUI の Figure とコンポーネントの各タイプは、ある種のコールバックをトリガーできます。各コンポーネントに対して利用できるコールバックは、そのコンポーネントのプロパティです。たとえば、プッシュ ボタンは 5 つのコールバック プロパティ ButtonDownFcn、

Callback、CreateFcn、DeleteFcn、KeyPressFcn をもちます。パネルは 4 つのコールバック プロパティ ButtonDownFcn、CreateFcn、DeleteFcn、ResizeFcn をもちます。これらのプロパティのそれぞれに対してコールバック関数を作成できますが、作成は必須ではありません。これには、GUI の Figure 自身に対するコールバックも含まれます。

各コールバックには、トリガー メカニズム、すなわち、そのコールバックを実行させるイベントがあります。次の表は、GUIDE が利用するコールバック プロパティ、イベントのトリガー、それらが適用するコンポーネントをリストします。最初の列のリンクは、それぞれのタイプのコールバックのドキュメンテーションの検索結果にリンクします。これらのリンクは、MATLAB ヘルプ ブラウザーを使用しているときに限り機能します。

コールバック プロパティ	イベントトリガー	コンポーネント
ButtonDownFcn	ポインターがコンポーネント上にあったり、または Figure のコンポーネントから 5 ピクセル以内のときに、GUI ユーザーがマウス ボタンを押すと実行します。	座標軸、Figure、ボタン グループ、パネル、ユーザー インターフェイス コントロール
Callback	アクションのコントロール。たとえば、GUI ユーザーがプッシュ ボタンをクリックしたり、メニュー項目を選択すると、実行します。	コンテキスト メニュー、メニュー、ユーザー インターフェイス コントロール
CellEditCallback	編集可能なセルに対して、表の値の編集についてレポートします。イベントデータを使用します。	uitable
CellSelectionCallback	表にあるマウスの動作で選択された、セルのインデックスをレポートします。イベントデータを使用します。	uitable
ClickedCallback	アクションのコントロール。プッシュ ツールまたはトグル ツールがクリックされると、実行します。トグル ツールでは、コールバックの実行は状態に依存しません。	プッシュ ツール、トグル ツール

コールバック プロパティ	イベントトリガー	コンポーネント
CloseRequestFcn	Figure が閉じるとき実行します。	Figure
CreateFcn	関数によってコンポーネントが作成されると、コンポーネントを初期化します。Figure やコンポーネントが作成された後、表示されるまでに実行します。	座標軸、ボタン グループ、コンテキストメニュー、Figure、メニュー、パネル、プッシュ ツール、トグル ツール、ツールバー、ユーザーインターフェイス コントロール
DeleteFcn	コンポーネントまたは Figure が削除される直前にクリーンアップ作業を実行します。	座標軸、ボタン グループ、コンテキストメニュー、Figure、メニュー、パネル、プッシュ ツール、トグル ツール、ツールバー、ユーザーインターフェイス コントロール
KeyPressFcn	このコールバックをもつコンポーネントまたは Figure が選択されていて GUI ユーザーがキーボードのキーを押すと、実行します。	Figure、ユーザーインターフェイス コントロール
KeyReleaseFcn	GUI ユーザーがキーボードキーをはなし、Figure がフォーカスされると実行します。	Figure
OffCallback	アクションのコントロール。トグル ツールの State が off に変わると、実行します。	トグル ツール
OnCallback	アクションのコントロール。トグル ツールの State が on に変わると、実行します。	トグル ツール

コールバック プロパティ	イベントトリガー	コンポーネント
ResizeFcn	GUI ユーザーがパネル、ボタン グループ、あるいは Resize プロパティが On になっている Figure をサイズ変更するときに実行します。	Figure、ボタン グループ、パネル
SelectionChangeFcn	GUI ユーザーがボタン グループ コンポーネントで、異なるラジオ ボタンやトグル ボタンを選択すると、実行します。	ボタン グループ
WindowButtonDownFcn	ポインターが Figure ウィンドウ内にある場合にユーザーがマウス ボタンをクリックすると、実行します。	Figure
WindowButtonMotionFcn	ポインターが Figure ウィンドウ上を動くと実行します。	Figure
WindowButtonUpFcn	マウス ボタンをはなすと、実行します。	Figure
WindowKeyPressFcn	Figure またはその子オブジェクトのいずれかが選択されているときに、キーを押すと、実行します。	Figure
WindowKeyReleaseFcn	Figure またはその子オブジェクトのいずれかが選択されているときに、キーをはなすと、実行します。	Figure
WindowScrollWheelFcn	GUI ユーザーがマウス ホイールをスクロールし、Figure がフォーカスされると、実行します。	Figure

メモ ユーザー インターフェイス コントロールは、プッシュ ボタン、スライダー、ラジオ ボタン、チェック ボックス、エディット テキスト ボックスやスタティック テキスト ボックス、リスト ボックス、トグル ボタンを含みます。これらは、*uicontrol* オブジェクトとして参照することができます。

特定のコールバックの詳細を表示するには、前述の表のリンクをたどり、プロパティ インスペクターでプロパティ名を右クリックし、[これはなに?] ポップアップ メニューを選択するか、またはコンポーネントのプロパティ リファレンス ページを調べてください。コンポーネントのプロパティとしては、たとえば、Figure プロパティ、Uicontrol プロパティ、Uibuttongroup プロパティ、または Uitable プロパティがあります。

コールバックの動作と、それらの形式についてさらに議論するには、“コールバックとは”(p.12-7)と、このドキュメンテーションの「プログラミングによる GUI の作成」の以下の節を参照してください。

GUI ファイルの概要

このセクションの内容…

“M ファイルと FIG-ファイル” (p.8-7)

“GUI M ファイルの構造” (p.8-8)

“既存の GUI M ファイルへのコールバック テンプレートの追加” (p.8-9)

“GUIDE が生成するコールバックについて” (p.8-9)

M ファイルと FIG-ファイル

既定では、ユーザーが最初に GUI を保存したり実行する場合、GUIDE は GUI を 2つのファイルに保存します。

- 拡張子 .fig をもつ FIG-ファイルは、GUI レイアウトや、プッシュ ボタン、座標軸、パネル、メニューなどの各 GUI コンポーネントの完全な描写情報を含んでいます。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、修正できません。FIG-ファイルは MAT-ファイルを特殊化したものです。詳細は、を参照してください。
- GUI の動作をコントロールするコールバックの初期化コードとテンプレートを含む、拡張子 .m をもつ M ファイルです。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します。コールバックは関数であるので、GUI M ファイルは MATLAB スクリプトにはなりません。

GUI を最初に保存するときはいつでも、GUIDE は MATLAB エディターに自動的に M ファイルを開きます。

FIG-ファイルと M ファイルは、同じ名前をもつ必要があります。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ます。レイアウト エディターに GUI をレイアウトすると、ユーザーのコンポーネントとレイアウトは FIG-ファイルに保存されます。GUI をプログラムすると、ユーザーのコードは対応する M ファイルに保存されます。

GUI が複数の ActiveX コンポーネントを含む場合、GUIDE は、各 ActiveX コンポーネントに対するファイルも生成します。詳細は、“ActiveX コントロール” (p.8-48)を参照してください。

GUI の名前付けと保存についての詳細は、章 7, “GUIDE GUI の保存と実行”を参照してください。GUI とそのファイルの名前を変更したい場合、“GUI と GUI ファイルの名前変更”(p.7-3)を参照してください。

GUI M ファイルの構造

GUIDE が作成する GUI M ファイルは、関数ファイルです。メイン関数の名前は、M ファイルの名前と同じです。たとえば、M ファイルの名前が `mygui.m` である場合、メイン関数の名前は `mygui` です。ファイル内の各コールバックは、メイン関数のサブ関数です。

GUIDE が M ファイルを生成するとき、GUIDE は各コンポーネントに対して、最も一般的に使用されるコールバックに対するテンプレートを自動的に加えます。また、`opening` 関数コールバックと `output` 関数コールバックと同様に、何らかの初期化コードも含みます。行うべきことは、GUI が希望通りに動作するように、コンポーネントのコールバックにコードを追加することです。`opening` 関数コールバックと `output` 関数コールバックにコードを追加することもできます。GUI M ファイルは、関数を次の表に示す順に並べます。

セクション	説明
コメント	<code>help</code> コマンドに応じて、コマンド ラインに表示されます。ユーザー GUI の必要に応じて、コメントを編集します。
初期化	GUIDE の初期化タスク。このコードは編集しないでください。
Opening 関数	GUI ユーザーが GUI にアクセスする前に、初期化タスクを実行します。
Output 関数	<code>opening</code> 関数がコントロールを返し、コントロールがコマンド ラインに返る前に、MATLAB コマンド ラインに出力を返します。
コンポーネントと Figure コールバック	GUI の <code>Figure</code> と個々のコンポーネントの動作をコントロールします。MATLAB は、コンポーネントや <code>Figure</code> 自身に対する特定のイベントに応じてコールバックを呼び出します。
ユーティリティ/補助関数	<code>Figure</code> またはコンポーネントに対するイベントに直接的には関連しない様々な関数を実行します。

既存の GUI M ファイルへのコールバック テンプレートの追加

GUIDE を保存するときに、GUIDE は、M ファイルに対するいくつかのコールバックに対するテンプレートを自動的に追加します。M ファイルに他のコールバックを追加したい場合は、簡単に追加できます。

GUIDE 内では、以下のいずれかの方法でコールバック サブ関数のテンプレートを GUI M ファイルに追加できます。コールバックを追加したいコンポーネントを選択し、次のようにします。

- ・ 右マウス ボタンをクリックし、レイアウト エディターのコンテキスト メニューを表示します。[コールバックの表示] サブメニューから希望するコールバックを選択します。
- ・ [表示] メニューにおいて、[コールバックの表示] サブメニューから希望するコールバックを選択します。
- ・ コンポーネントをダブルクリックして、プロパティ インスペクターにそのプロパティを表示します。プロパティ インスペクターで、M ファイルにインストールしようとするコールバック名の隣にある“鉛筆と紙”が描かれたアイコン をクリックします。
- ・ ツールバー ボタンでは、ツールバー エディターにおいて、[クリック時のコールバック] (プッシュ ツール ボタンの場合) の隣にある、あるいは [オン時のコールバック]、または [オフ時のコールバック] (トグル ツールの場合) の隣にある、[表示] ボタンをクリックします。

これらの動作のいずれかを行うと、GUIDE は GUI M ファイルにコールバック テンプレートを追加し、直前にユーザーが追加したコールバックで編集するために M ファイルを開きます。GUI M ファイルに現存するコールバックを選択すると、GUIDE はコールバックは追加しませんが、ユーザーが選択したコールバックで編集するために M ファイルを開きます。

詳細は、“コールバックとコンポーネントとの関連付け” (p.8-11)を参照してください。

GUIDE が生成するコールバックについて

GUI コンポーネントに対して GUIDE で生成されたコールバックは、プログラミングで作成されたコールバックに似ていますが、いくつか違いがあります。

- ・ GUIDE は、GUI M ファイル内で関数テンプレートとしてコールバックを作成します。これは、GUI コンポーネントが関数ハンドルにより呼び出します。

GUIDE は、コールバックのタイプとコンポーネントの Tag プロパティに基づいてコールバックに名前を付けます。たとえば、togglebutton1_Callback はそのような既定のコールバックの名前です。コンポーネントの Tag を変更すると、GUIDE は M ファイルのそのコールバックすべての名前を変更して新しいタグを含みます。プロパティインスペクターを利用すると、コールバックの名前を変更したり、コールバックを他の関数で置き換えたり、あるいは完全に削除したりできます。

- GUIDE は、コールバックに 3 つの引数を与えます。これらは常に同じ名前をもちます。
- GUIDE が生成するコールバックに引数を追加できますが、GUIDE がそこに配置する引数の変更や削除は行わないでください。
- GUIDE が生成するコールバックは、名前を編集したり、コンポーネントの Tag を変更することで、名前を変更できます。
- プロパティ インスペクターからコンポーネントを消去することで、コンポーネントからコールバックを削除できます。このアクションにより、M ファイルのコードは削除されません。
- 複数のコンポーネントでコードの共有を可能にするために、複数のコンポーネントに対して同じコールバック関数を指定することができます。

GUIDE でコンポーネントを削除した後、それがもっていたコールバックはすべて M ファイルに残ります。他のコンポーネントがコールバックを使用していないことを確認してから、コールバックのコードを手動で削除できます。詳細については、“GUI M ファイルからのコールバックの削除”(p.8-14)を参照してください。コンポーネントを削除せずにコールバックを削除する方法については、を参照してください。

コールバックとコンポーネントとの関連付け

このセクションの内容…

“GUI コンポーネント” (p.8-11)

“コールバック プロパティを自動的に設定” (p.8-11)

“GUI M ファイルからのコールバックの削除” (p.8-14)

GUI コンポーネント

GUI は多数のコンポーネントをもつことができます。GUIDE は、特定のコンポーネントに対する特定のイベントに応答して、どのコールバックが実行するかを指定する方法を提供します。GUI ユーザーが [はい] ボタンをクリックする際に実行するコールバックは、[いいえ] ボタンに対して実行するものとは通常異なります。同様に、各メニュー項目は、通常、異なる機能を行います。コールバック プロパティのリストとそれぞれが適用されるコンポーネントについては、“コールバックの種類” (p.8-2) を参照してください。

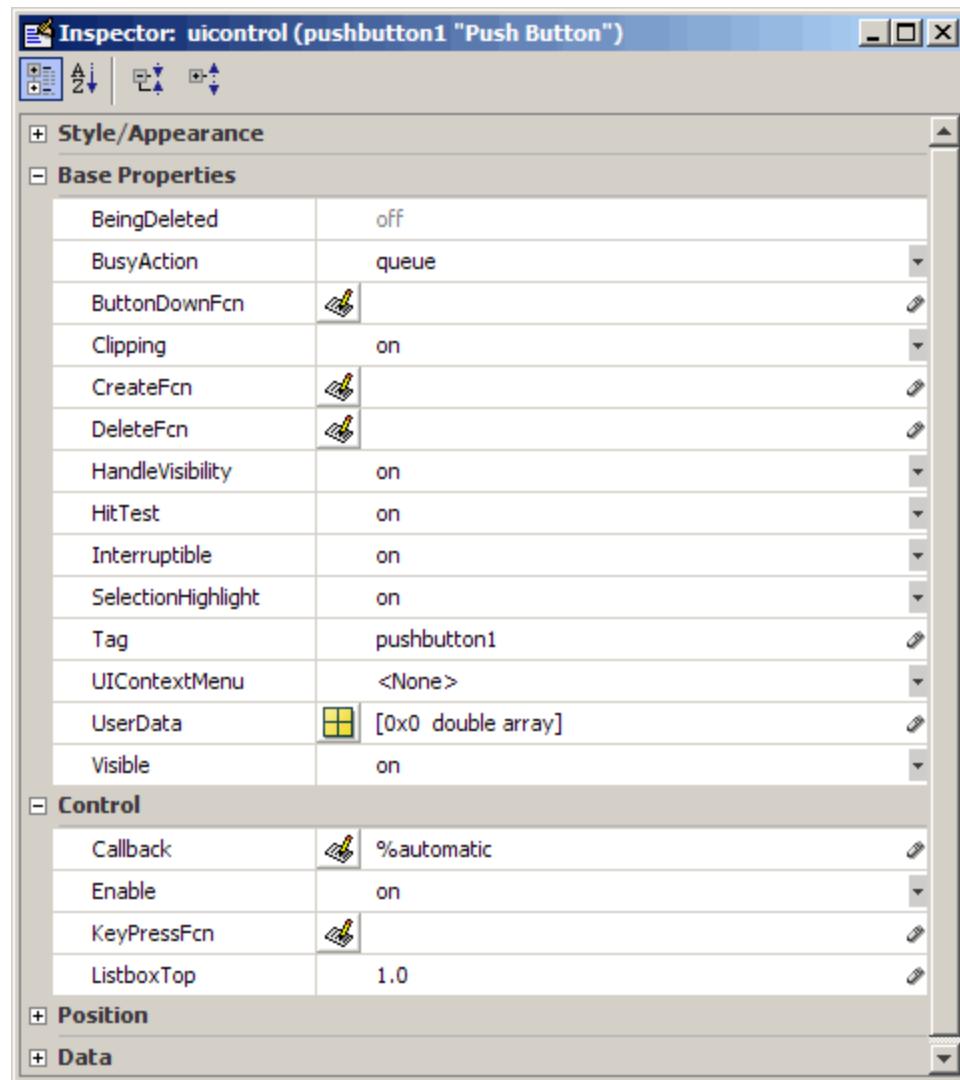
コールバック プロパティを自動的に設定

GUIDE を使用している場合、GUIDE は、各コンポーネントに対して最も一般的に使用されるコールバック プロパティを最初に %automatic に設定します。たとえば、プッシュボタンは 5 つのコールバック プロパティ、ButtonDownFcn、Callback、CreateFcn、DeleteFcn、KeyPressFcn をもちます。GUIDE は、最も一般的に使用されるコールバックである Callback プロパティのみを %automatic に設定します。プロパティインスペクターを使用して、他のコールバック プロパティを %automatic に設定することもできます。それを行うには、コールバック名の隣にある “鉛筆と紙” が描かれたア

イコン  をクリックします。GUIDE は直ちに %automatic を、コールバックに対して GUI が呼び出す列である MATLAB 表現で置き換えます。コーリング シークエンスで、コールバック名、たとえば、サブ関数名が、コンポーネントの Tag プロパティとコールバック プロパティの名前から作成されます。

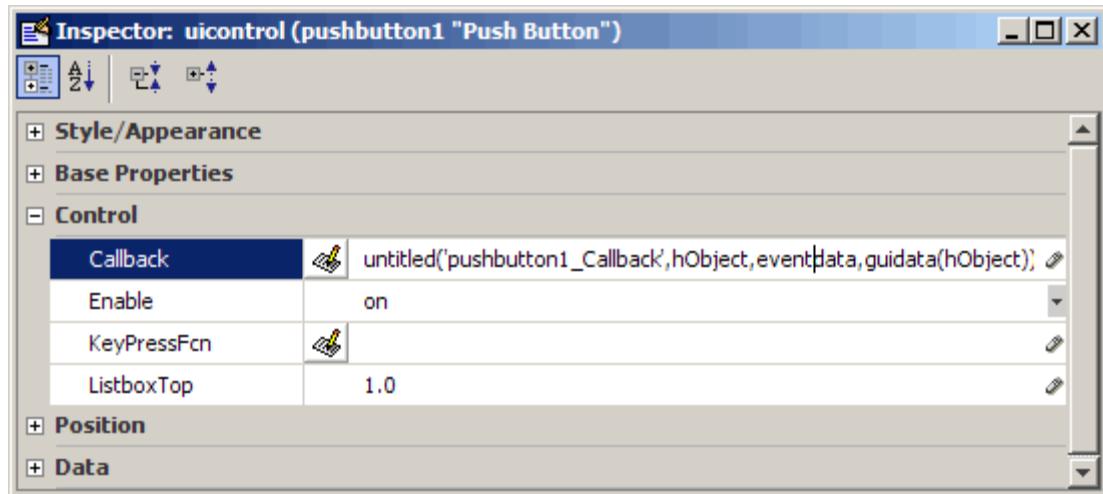
以下の図は、GUI を保存する前の、GUIDE プロパティインスペクターでのプッシュボタンのプロパティを示します。GUIDE は Tag プロパティを pushbutton1 に設定します。GUI を保存する前に、Callback プロパティは %automatic と表示されます。GUI を保存するときに、GUIDE が名前を生成することを示します。

メモ GUI を保存する前に文字列 %automatic を変更すると、GUIDE は、コンポーネントまたはメニュー項目に対するコールバックを自動的に追加しません。ユーザー自身でコールバックを提供するかどうかは、ユーザーが決めることができます。このコールバックは、ユーザーが入力した文字列と同じである必要があります。



この GUI を保存する場合、GUIDE は、コンポーネントの Tag プロパティの値にアンダーライン (_) とコールバック プロパティの名前を付けて、コールバックの名前を作成します。たとえば、GUI のプッシュ ボタンの Callback プロパティに対する MATLAB 表現は、Tag プロパティの pushbutton1 が untitled である場合、以下のようにになります。

```
untitled('pushbutton1_Callback', hObject, eventdata, guidata(hObject))
```



この場合、untitled は、その GUI のメイン関数の名前であるのと同様に GUI M ファイルの名前です。残りの引数は、pushbutton1_Callback に対する入力引数です。特に、

- hObject はコールバック オブジェクト（この場合、pushbutton1）のハンドルです。
- eventdata は、イベント データを含む MATLAB 構造体を渡します。オブジェクトがイベント データを生成しない場合、eventdata は空の行列を含みます。 eventdata 構造体には、構造体が提供するオブジェクトの各タイプに固有のコンテンツ（フィールド名）があります。
- guidata(hObject) は、この GUI に対するハンドル構造体を取得し、コールバックに渡します。

コールバックの引数とそれらのカスタマイズ方法の詳細は、“入力引数”(p.8-21) と “コールバック関数のシグネチャ”(p.8-17) を参照してください。

GUI を保存するとき、GUIDE はエディターに GUI M ファイルも開きます。この M ファイルは、Tag が pushbutton1 であるコンポーネントに対する、Callback コールバックのテ

ンプレートを含みます。GUI をアクティブにすると、プッシュ ボタンをクリックすることで、そのコンポーネントに対する Callback コールバックが実行させます。

GUIDE がコールバックの名前を付けた後の、コールバックの名前変更についての詳細は、“GUIDE が割り当てるコールバックの変更”(p.8-20)を参照してください。GUI M ファイルへのコールバック テンプレートの追加についての詳細は、“既存の GUI M ファイルへのコールバック テンプレートの追加”(p.8-9)を参照してください。

次のトピック“コールバック構文と引数”(p.8-15)は、コールバック テンプレートについての詳細を提供します。

GUI M ファイルからのコールバックの削除

GUI M ファイルからコールバックを削除したい場合があります。コールバックが手動で作成されたか、自動的に作成されたかに関わらず、コールバックを削除できます。コールバックを削除する理由として一般的なものに、以下があります。

- ・ コールバックの応答先であったコンポーネントを削除する場合。
- ・ プロパティ インスペクターにおいて適切なコールバック プロパティで識別した別のコールバック関数をコンポーネントに実行させる場合。説明とガイドラインは、“GUIDE が割り当てるコールバックの変更”(p.8-20)を参照してください。

コールバックの削除は、そのコールバックが使用されていないことを確認した場合にのみ行います。このコールバックが GUI の他のところで使用されていないことを確認するには、

- ・ GUI M ファイルでコールバックの名前がある場所を探します。
- ・ GUIDE で GUI を開き、プロパティ インスペクターを使用して、削除したいコールバックを使用しているコンポーネントがないかチェックします。

いずれの場合も、コールバックに対するリファレンスを見つけると、リファレンスを削除するか、あるいはコールバックを GUI M ファイルに保持します。一旦、GUI でコードが必要ないことを確かめたら、M ファイルからコールバック関数全体を手動で削除します。

コールバック構文と引数

このセクションの内容…

- “コールバック テンプレート” (p.8-15)
- “コールバック関数の名前付け” (p.8-16)
- “GUIDE が割り当てるコールバックの変更” (p.8-20)
- “入力引数” (p.8-21)

コールバック テンプレート

GUIDE は、コールバック構文と引数についての規則を定義し、これらの規則を M ファイルに追加するコールバックのテンプレートで実現します。各テンプレートは、プッシュボタンに対する Callback サブ関数に対するものに似ています。

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

...
```

最初のコメント行は、コールバックの実行を引き起こすイベントについて記述します。この下に関数定義行が続きます。残りのコメントは入力引数を記述します。最後のコメントの後に、ユーザー コードを挿入してください。

Figure と GUI コンポーネントのコールバックには、eventdata 引数にイベントに固有のデータを与えるものがあります。たとえば、以下は、プッシュボタンの KeyPressFcn コールバックのテンプレートです。

```
% --- Executes on key press with focus on pushbutton1
function pushbutton1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
% Key: name of the key that was pressed, in lower case
% Character: character interpretation of the key(s) that was pressed
% Modifier: name(s) of the modifier key(s) (i.e., control, shift)
% pressed
```

```
% handles structure with handles and user data (see GUIDATA)
```

次の表には、イベントデータを与えるコールバックや、コールバックが適用されるコンポーネントが挙げられています。詳細は、適切なプロパティのリファレンスページを参照してください。

GUI コンポーネント	イベントデータをもつコールバック	プロパティのリファレンスページ
Figure	KeyPressFcn、 KeyReleaseFcn、 WindowKeyPressFcn、 WindowKeyReleaseFcn、 WindowScrollWheel	Figure プロパティ
ユーザー インターフェイス コントロール (uicontrol)	KeyPressFcn	Uicontrol プロパティ
ボタン グループ (uibuttongroup)	SelectionChangeFcn	uibuttongroup プロパティ
テーブル (uitable)	CellEditCallback、 CellSelectionCallback	Uitable プロパティ

メモ 新規のコールバックに対して、コールバックのコメント行を自動生成しないことも可能です。[設定] ダイアログ ボックスで、[GUIDE] を選択して [新規に生成されたコールバック関数にコメントを追加] のチェックを外します。

コールバック関数の名前付け

前のコールバックの例は、以下の関数の定義を含みます。

```
function pushbutton1_Callback(hObject, eventdata, handles)
```

GUIDE がテンプレートを作成するとき、GUIDE はコンポーネントの Tag プロパティにアンダーライン (_) とコールバック プロパティの名前を付けてコールバックの名前を作成します。上記の例では、pushbutton1 はプッシュ ボタンの Tag プロパティであり、Callback はプッシュ ボタンのコールバック プロパティの 1 つです。Tag プロパティは、GUI 内のコンポーネントを一意的に識別します。

コンポーネントを追加後最初に GUI を保存するとき、GUIDE はそのコンポーネントに対するコールバックを M ファイルに追加し、Tag プロパティの現在の値を用いてコールバックの名前を生成します。コンポーネントに対する既定の Tag を変更した場合、GUI を保存する前に、他のコンポーネントの Tag の値が複製されていないことを確認してください。GUIDE は、存在する Tag の複製が見つかると警告します。

詳細は、“GUIDE が割り当てるコールバックの変更”(p.8-20)および“コールバックとコンポーネントとの関連付け”(p.8-11)を参照してください。

コールバック関数のシグネチャ

関数シグネチャは関数名、関数のパラメーターの数、順序、タイプ、関数に適用する限定子を挙げます。すくなくとも一度保存した GUI のコンポーネントをプロパティインスペクターに表示すると、その Callback プロパティがすでに設定されていることがわかります。GUIDE が GUI を保存するとき、GUI は以下のことを行います。

- ・ コールバック シグネチャを生成して、それを Callback プロパティの値として割り当てます。
- ・ シグネチャがポイントする関数に対して、GUI M ファイルにテンプレートを追加します。

コンポーネントには、他のコールバック、たとえば、CreateFcn または DeleteFcn があります。GUIDE はこれらのコールバックを同じ方法で与えます。コールバックが何かを行うように、テンプレートにコードを追加するかどうかは、ユーザーが決めることです。

たとえば、プロパティインスペクターでプッシュボタンの Callback プロパティに対して、ユーザーが “鉛筆と紙” が描かれたアイコン  をクリックすると、GUIDE は MATLAB エディターに GUI M ファイルを表示し、コールバックの最初の行にカーソルを置きます。GUIDE が、以下のように、M ファイルに関数を定義すると、

```
function pushbutton1_Callback(hObject, eventdata, handles)
```

プロパティインスペクターに表示された、Callback プロパティに対する関数シグネチャは、次のようにになります。

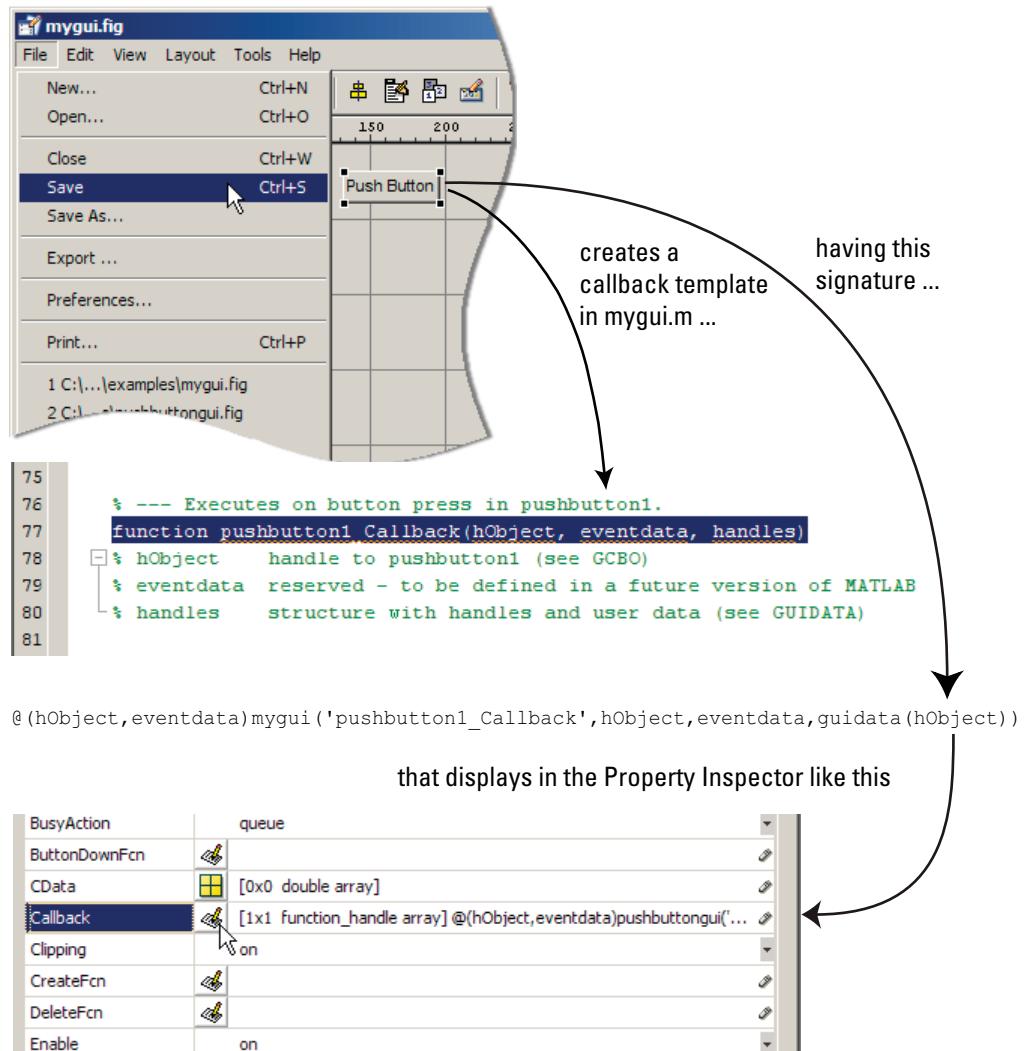
```
@(hObject, eventdata) mygui('pushbutton1_Callback', hObject, eventdata, guidata(hObject))
```

構文 @(hObject, eventdata) は、これが無名関数であることを示します。以下の情報を与えて、ユーザーがこのプッシュボタンをクリックすると、シグネチャにより、MATLAB は適切なコールバックを実行できます。

- ・ コールバック関数が存在する M ファイルの名前 (' mygui')
- ・ M ファイル内のコールバック関数名 (' pushbutton1_Callback')
- ・ コールバック関数に渡す引数リスト
 - 1 hObject – コールバックを与えるコンポーネント(この場合、プッシュボタン)のハンドル
 - 2 eventdata – コンポーネントにより生成されるイベントデータを含む構造体。プッシュボタンやイベントデータを生成しない他のコンポーネントの場合、この引数は空行列になります。
 - 3 guidata(hObject) – GUI に対する “handles 構造体” (p.8-23)。コールバック間のコンポーネントハンドルのやりとりに使われます。

以下の図は、これらの要素がどのように互いに関連するかを説明しています。

Saving a GUI with a push button in GUIDE...



GUIDE が生成するコールバックの詳細は、“入力引数”(p.8-21)を参照してください。

GUIDE が割り当てるコールバックの変更

“コールバック関数の名前付け”(p.8-16)に述べるように、コンポーネントの Tag プロパティ (checkbox1) とそのコールバック タイプを連結して、GUIDE はコールバック名を生成します。コールバックのタイプを変更することはできませんが、その Tag を変更できます。Tag は、ユーザーが次に GUI を保存するときにコールバックの名前を変更します。

コールバックに、より意味のある名前を与えるには、コンポーネントの Tag プロパティを変更します。たとえば、Tag プロパティを checkbox1 から warnbeforesave に変更します。ユーザーが付ける名前をもつコールバック テンプレートを GUIDE が自動的に作成するように、可能であれば、GUI を保存する前に Tag プロパティを変更してください。ただし、GUI を保存した後に Tag プロパティを変更しようとした場合、すべてのコンポーネントのタグが異なるならば、新規の Tag に従い GUIDE は以下の項目を更新します。

- ・ M ファイルにおけるコンポーネントのコールバック関数
- ・ コンポーネントのコールバック プロパティの値。これは、プロパティ インスペクターで見ることができます。
- ・ コンポーネントのハンドル番号を含む handles 構造体のフィールドに対する M ファイル のリファレンス。handles 構造体についての詳細は、“handles 構造体”(p.8-23)を参照してください。

Tag プロパティを変更せずに特定のコールバック関数の名前を変更するには、

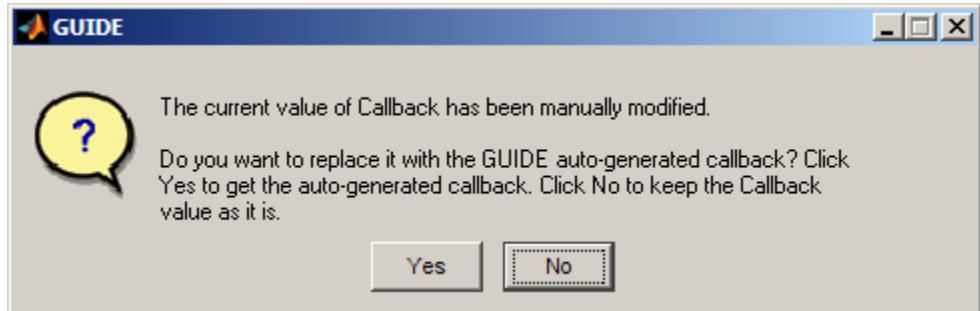
- ・ プロパティ インスペクターにおいて、コールバック プロパティ内の名前の文字列を新規の名前で置き換えます。たとえば、mygui のプッシュ ボタンに対するコールバック プロパティの値が

```
mygui('pushbutton1_Callback', hObject, eventdata, guidata(hObject))
```

の場合、文字列 pushbutton1_Callback はコールバック関数の名前です。適切な名前、たとえば closethegui に変更します。

- ・ 必要に応じて、M ファイルのコールバック関数名のインスタンスを更新します。たとえば、そのコールバック関数の定義において function closethegui に変更します。

コールバック シグネチャを変更後、GUI M ファイル内の関数定義に進むために “鉛筆と紙” が描かれたアイコン をクリックすると、GUIDE がダイアログ ボックスを表示するので変更を確認できます。



GUIDE が自動生成したコールバックに戻るには、[はい] をクリックします。変更されたコールバックを保つには、[いいえ] をクリックします。

メモ M ファイルに存在する他の関数に対するコールバックをポイントしている場合を除き、プロパティ インスペクターでコールバック関数のシグネチャを変更したら、GUI M ファイルのコールバック関数定義を忘れずに変更してください。たとえば、いくつかのトグル ボタンまたはメニュー項目で同じコールバックを使用したい場合があります。

入力引数

GUIDE が生成した GUI M ファイルのすべてのコールバックは、以下の標準的な入力引数をもちます。

- `hObject` – コールバックがトリガーされるオブジェクト、たとえば GUI コンポーネントのハンドル。ボタン グループ `SelectionChangeFcn` コールバックの場合、`hObject` は選択されたラジオ ボタンまたはトグル ボタンのハンドルです。
- `eventdata` – ユーザーのアクションでトリガーされるイベントの列。ユーザーのアクションとして、MATLAB 構造体（あるいは、イベント データを生成しないコンポーネントに対する空行列）の形でコンポーネントがエミットしたテーブルの選択などがあります。
- `handles` – GUI のすべてのオブジェクトのハンドルを含む MATLAB 構造体。さらに、この構造体にアプリケーションが定義するデータが含まれることがあります。この構造体の詳細は、“`handles` 構造体” (p.8-23)を参照してください。

オブジェクトのハンドル

1番目の引数は、コールバックを与えるコンポーネントのハンドルです。このハンドルを使用してコールバック コードが使う関連するプロパティを取得し、プロパティは必要に応じて変更します。たとえば、

```
theText = get(hObject,'String');
```

は String プロパティをローカル変数 theText に置いてください。このプロパティは、スタティック テキストの内容またはボタンの名前になります。プロパティは設定することで変更できます。たとえば、以下のようにします。

```
set(hObject,'String',date)
```

この特定のコードは、現在のデータを表示するためにオブジェクトのテキストを変更します。

イベント データ

イベント データは、キー プレス、ホイール スクロール、マウス ドラッグなど、ユーザーの動作を記述するデータの流れです。GUIDE GUI の自動生成されたコールバックは、Handle Graphics® と uicontrol に対するイベント データと uitable オブジェクト コールバックに対するイベント データにアクセスできます。以下のものは、トリガーされるときにイベント データを受け取ります。

- uitable の CellEditCallback
- uitable の CellSelectionCallback
- uicontrol の KeyPressFcn と Figure
- Figure の KeyReleaseFcn
- uibuttongroup の SelectionChangeFcn
- Figure の WindowKeyPressFcn、またはその子オブジェクトのいずれか
- Figure の WindowKeyReleaseFcn、または子オブジェクトのいずれか
- Figure の WindowScrollWheelFcn

イベント データは、3 つの標準的な引数の 2 番目として GUIDE が生成するコールバックに渡されます。イベント データを与えないコンポーネントでは、引数は空です。イベント データを与えるコンポーネントでは、引数は構造体を含みます。この構造体は、構造体を生成したコンポーネントとイベントのタイプに従い、構成が変化します。

たとえば、キープレスに対するイベントデータは、現在押されているキー（複数のキーの場合もあり）の情報を提供します。以下は、GUIDE が生成した KeyPressFcn コールバックのテンプレートです。

```
% --- Executes on key press with focus on checkbox1 and none of its controls.
function checkbox1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata   structure with the following fields (see UICONTROL)
%           Key: name of the key that was pressed, in lower case
%           Character: character interpretation of the key(s) that was pressed
%           Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
```

渡された eventdata 構造体には 3 つのフィールドがあります。これらのフィールドでは、(' =' など) 押されている Character、(' control' など) キーの Modifier、(' equals' など、説明された) Key の名前を識別します。

イベントデータを与えるコンポーネントは、データを渡すために、イベントに固有のフィールド名をもつ異なる構造体を使用します。通常、イベントデータをもつコールバックは、イベントが続く限り、あるいは時にはイベントが始まる時点で、さらにその後はその値が変更するときに限り、繰り返し実行されます。

コールバックがどのようにイベントデータを使用するかについては、GUIDE の uitable の例 “テーブルのデータを対話的に調べる GUI” (p.10-32) とプログラミングによる uitable の例 “テーブルのデータを表示してグラフを作成する GUI” (p.15-17) を参照してください。

handles 構造体

GUIDE は Figure 内のすべてのオブジェクトのハンドルを含む handles 構造体を作成します。エディット テキスト、パネル、popupmenu メニュー、プッシュ ボタンを含む GUI に対して、handles 構造体は元々これと同じように見えます。GUIDE は、コンポーネントの Tag プロパティを使用して、そのハンドルに対する構造体の要素に名前を付けます。

```
handles =
    figure1: 160.0011
    edit1: 9.0020
    uipanel1: 8.0017
    popupmenu1: 7.0018
    pushbutton1: 161.0011
```

```
output: 160.0011
```

GUIDE は、handles 構造体を作成して GUI データを保持します。これは、すべてのコールバックに対して入力引数として渡され、GUI のコールバックがプロパティの値とアプリケーション データを共有できるようにします。

GUI データの詳細は、“データ管理の仕組み”(p.9-2) と guidata リファレンス ページを参照してください。

handles 構造体にフィールドを追加する詳細、および 構造体を適切に保存するための説明は、章 13, “アプリケーション定義のデータの管理”を参照してください。

コールバックの初期化

このセクションの内容…

“Opening 関数” (p.8-25)

“出力関数” (p.8-28)

Opening 関数

opening 関数は、各 GUI M ファイル内の最初のコールバックです。これは GUI が表示される直前、ただし、すべてのコンポーネントが作成された後、すなわち コンポーネントの CreateFcn コールバックが存在する場合には、それが実行した後に実行されます。

ユーザーが GUI ヘアクセスする前に、opening 関数を使用して、初期化タスクを実行することができます。これにより、たとえば、データの作成や外部ソースからのデータの読み込みを行うことができます。GUI コマンド ライン引数は、opening 関数に渡されます。

- ・ “名前付けとテンプレート” (p.8-25)
- ・ “入力引数” (p.8-26)
- ・ “初期のテンプレートコード” (p.8-28)

名前付けとテンプレート

GUIDE は、M ファイルの名前に _OpeningFcn を付け加えることにより、opening 関数に名前を付けます。以下は、mygui M ファイルに生成される opening 関数テンプレートの例です。

```
% --- Executes just before mygui is made visible.
function mygui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mygui (see VARARGIN)

% Choose default command line output for mygui
handles.output = hObject;
```

```
% Update handles structure  
guidata(hObject, handles);  
  
% UIWAIT makes mygui wait for user response (see UIRESUME)  
% uiwait(handles.mygui);
```

入力引数

opening 関数は、4つの入力引数 hObject、 eventdata、 handles、 varargin をもちます。最初の 3 つは、“入力引数”(p.8-21)に述べたものと同じです。最後の引数 varargin によって、コマンド ラインから opening 関数に引数を渡すことができます。opening 関数は、それら(たとえば、プロパティ値の設定)にアクションをとり、handles 構造体にそれらを追加することによりコールバックに対して引数を利用可能とすることもできます。

varargin の利用の詳細は、varargin のリファレンス ページと MATLAB「プログラミングの基礎」ドキュメンテーションの「数が可変の引数の引き渡し」を参照してください。

Opening 関数にオブジェクトのプロパティを渡す。コンポーネントに連続する 2 つのコマンド ライン引数として、プロパティの名前/値の組を渡し、opening 関数内にその値を設定できます。Figure のプロパティをユーザーが設定している場合は、GUIDE はプロパティを自動的に取り扱います。たとえば、Position は(既定の文字単位で) 有効な Figure プロパティなので、my_gui('Position', [71.8 44.9 74.8 19.7]) は指定した位置に GUI を開きます。

プロパティまたはプロパティの組に対して新規の名前を定義できます。たとえば、ユーザーが便利なように、GUI が Figure プロパティに対するエイリアスを受け取るようにすることができます。たとえば、Name を呼び出すのではなく Title 引数をもつ GUI が開くようにしたい場合があります。Name は、GUI のタイトル バー上の名前を指定するプロパティです。これを行うには、Figure の Name プロパティを設定するために、その OpeningFcn にコードを提供する必要があります。以下の例は、これを行う方法を説明します。

有効な Figure プロパティでない入力引数を渡すと、ユーザー コードはその名前を認識して、適切なオブジェクトに適当なプロパティを設定するために名前/値の組を使用する必要があります。そうでない場合、引数は無視されます。以下の例は、[GUIDE の クイック スタート] ダイアログ ボックスから利用できる、モーダル クエスチョン ダイアログ GUI テンプレートに対する opening 関数からのものです。追加されたコードは、メッセージをもつモーダル ダイアログを開きます。このメッセージは、コマンド ラインから、またはこのダイアログを呼び出す他の GUI で指定されたものです。たとえば、

```
mygui('String', 'Do you want to exit?')
```

は、GUI にテキスト 'Do you want to exit?' を表示します。これを行うには、'String' は有効な Figure プロパティではないので、opening 関数をカスタマイズする必要があります。'String' は、スタティックなテキスト プロパティです。モーダル クエスチョン ダイアログのテンプレート ファイルは、以下のコードを含みます。

- ・ 関数 nargin を使用して、ユーザーが指定した引数の数 (hObject, eventdata, handles は含みません) を判定します。
- ・ varargin を解析してプロパティ名/値の組を取得して、各名前の文字列を小文字に変換します。
- ・ 引数 'title' が Figure の Name プロパティのエイリアスとして使用されるケースを扱います。
- ・ 'string' を取り扱います。次の値を String プロパティとして適切なスタティック テキスト オブジェクトに割り当てます。

```
function modalgui_OpeningFcn(hObject, eventdata, handles, varargin)

%
% Insert custom Title and Text if specified by the user
% Hint: when choosing keywords, be sure they are not easily confused
% with existing figure properties. See the output of set(figure) for
% a list of figure properties.
if(nargin > 3)
    for index = 1:2:(nargin-3),
        if nargin-3==index, break, end
        switch lower(varargin{index})
            case 'title'
                set(hObject, 'Name', varargin{index+1});
            case 'string'
                set(handles.text1, 'String', varargin{index+1});
        end
    end
end
%
```

`if` ブロックは、プロパティ名またはエイリアスをチェックして、`varargin` の奇数個の要素をループします。`case` ブロックは、以下の（偶数個の）`varargin` 要素を、Figure またはそのコンポーネントの 1 つの適切なプロパティに対する値として割り当てます。`opening` 関数に実行させる追加のプロパティを割り当てるために、さらにケースを追加できます。

初期のテンプレートコード

最初に、入力関数のテンプレートには、以下のコード行が含まれます。

- `handles.output = hObject` は新しい要素 `output` を `handles` 構造体に追加し、入力引数 `hObject` の値を割り当てます。`hObject` は Figure のハンドル、すなわち GUI のハンドルです。このハンドルは、後で `output` 関数によって使用されます。出力関数についての詳細は、“出力関数”(p.8-28)を参照してください。
- `guidata(hObject, handles)` は、`handles` 構造体を保存します。`handles` 構造体への変更を保存するには、`guidata` を使用する必要があります。`handles` 構造体のフィールドの値を設定するだけでは不十分です。詳細は、“`handles` 構造体”(p.8-23)および“GUI データ”(p.9-7)を参照してください。
- `uiwait(handles.mygui)` は、最初はコメントアウトされ、`uiresume` が呼び出されるか、または GUI が削除されるまで、GUI の実行をブロックします。ユーザーは、`uiwait` を使って他の MATLAB ウィンドウにアクセスし、情報を得ることができるごとに注意してください。GUI が開いたときに、GUI がブロックされている状態にしたい場合は、このステートメントのコメント記号を削除してください。

出力関数

`output` 関数は、実行中に生成された出力をコマンド ラインに出力します。これは `opening` 関数がコントロールを返し、コントロールがコマンド ラインに返る前に実行されます。これは、ユーザーが `opening` 関数で出力を生成する必要があるか、または他のコールバックが出力を生成する間に、`opening` 関数でその実行を停止させるために、`uiwait` を呼び出す必要があることを意味します。

- “名前付けとテンプレート”(p.8-29)
- “入力引数”(p.8-29)
- “出力引数”(p.8-29)

名前付けとテンプレート

GUIDE は、M ファイルの名前に _OutputFcn を付け加えることにより、output 関数に名前を付けます。次のものは、mygui M ファイルに生成される output 関数テンプレートの例です。

```
% --- Outputs from this function are returned to the command line.
function varargout = mygui_OutputFcn(hObject, eventdata, ...
    handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

入力引数

output 関数は 3 つの入力引数 hObject、eventdata、および handles をもちます。これらは、“入力引数”(p.8-21)に述べたものと同じです。

出力引数

output 関数は、コマンド ラインに返す 1 つの出力引数 varargout をもちます。既定では、output 関数は、handles.output を varargout に割り当てます。従って、既定の出力は GUI に対するハンドルです。このハンドルは、opening 関数で、handles.output に割り当てられます。

以下の方法により出力を変更することができます。

- handles.output の値の変更。この値は、構造体またはセル配列を含む任意の有効な MATLAB 値とすることができます。
- varargout への出力引数の追加

varargout はセル配列です。これは、可変数の出力引数を含むことができます。既定では、GUIDE は、出力引数 handles.output のみ作成します。追加の出力引数を作成するには、handles 構造体に新しいフィールドを作成し、次のようなコマンドを使用してフィールドを varargout に追加します。

```
varargout{2} = handles.second_output;
```

例:GUIDE GUI コンポーネントのプログラミング

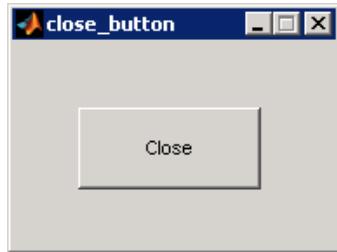
このセクションの内容…

- “プッシュ ボタン” (p.8-30)
- “トグル ボタン” (p.8-32)
- “ラジオ ボタン” (p.8-32)
- “チェック ボックス” (p.8-33)
- “エディット テキスト” (p.8-34)
- “テーブル” (p.8-35)
- “スライダー” (p.8-36)
- “リスト ボックス” (p.8-36)
- “ポップアップ メニュー” (p.8-38)
- “パネル” (p.8-39)
- “ボタン グループ” (p.8-42)
- “座標軸” (p.8-44)
- “ActiveX コントロール” (p.8-48)
- “メニュー項目” (p.8-59)

以下の節で述べる大部分のコントロールを組み込んだ完成した GUI の例として、GUIDE ドキュメンテーションの “多くのコンポーネントをもつ GUI の取り扱い” (p.6-24) を参照してください。

プッシュ ボタン

この例は、プッシュ ボタンのみを含みます。ボタンをクリックすると、GUI が閉じます。



次はプッシュ ボタンの Callback プロパティを用いたコールバックです。これは、コマンド ラインに文字列 Goodbye を表示してから GUI を閉じます。

```
function pushbutton1_Callback(hObject, eventdata, handles)
display Goodbye
close(handles.figure1);
```

プッシュ ボタンあるいはトグル ボタンへのイメージの追加

プッシュ ボタンあるいはトグル ボタンにイメージを追加するには、ボタンの CData プロパティに、トゥルーカラー イメージを定義する RGB 値の $m \times n \times 3$ 配列を割り当てます。たとえば、配列 a は (rand で生成される) 0 から 1 までの乱数の値を使用して、 16×64 トゥルーカラー イメージを定義します。

```
a(:,:,1) = rand(16, 64);
a(:,:,2) = rand(16, 64);
a(:,:,3) = rand(16, 64);
set(hObject, 'CData', a)
```



ボタンが作成されたときにイメージを追加するには、ボタンの CreateFcn コールバックにそのコードを追加します。この場合、通常ラベルとして使用される、ボタンの String プロパティの値を削除して差し支えありません。

行列 X の変換と対応するカラーマップ、すなわち、(X, MAP) イメージから RGB (トゥルーカラー) 形式への変換の詳細は、関数 ind2rgb を参照してください。

トグル ボタン

トグル ボタンのコールバックは、トグル ボタンがどんな状態にあるかを確認する必要があります。トグル ボタンが押されたとき、Value プロパティは Max プロパティに設定され、トグル ボタンが押されていないとき、Min プロパティに設定されます。

次のコードは、GUI M ファイルにおいてコールバックをプログラムする方法を説明します。

```
function togglebutton1_Callback(hObject, eventdata, handles)
button_state = get(hObject,'Value');
if button_state == get(hObject,'Max')
    % Toggle button is pressed, take appropriate action
    ...
elseif button_state == get(hObject,'Min')
    % Toggle button is not pressed, take appropriate action
    ...
end
```

トグル ボタンの Value プロパティに、トグル ボタンの Max または Min プロパティの値を設定することによって、選択されたボタンをもつラジオ ボタンを作成します。この例は、そのような割り当てに対して可能な構文を説明します。

```
set(handles.togglebutton1,'Value','Max')
```

は、トグル ボタンの Tag プロパティが togglebutton1 のトグル ボタンを押された状態にします。

メモ ボタン グループを用いて、トグル ボタンの排他的な選択を管理できます。詳細は、“ボタン グループ”(p.8-42)を参照してください。

ラジオ ボタン

Value プロパティの状態を確認することにより Callback コールバック内から、ラジオ ボタンの現在の状態を決定することができます。ラジオ ボタンが選択されると、Value プロパティが Max プロパティに等しくなります。ラジオ ボタンが選択されないと、Value プロパティは Min プロパティに等しくなります。これについて、次の例で説明します。

```
function radiobutton1_Callback(hObject, eventdata, handles)
if (get(hObject,'Value') == get(hObject,'Max'))
```

```
% Radio button is selected, take appropriate action
else
    % Radio button is not selected, take appropriate action
end
```

ラジオ ボタンの Value プロパティに、ラジオ ボタンの Max または Min プロパティの値を設定することによって、ラジオ ボタンの状態をプログラムによっても変更できます。この例は、そのような割り当てに対して可能な構文を説明します。

```
set(handles radiobutton1, 'Value', 'Max')
```

は、Tag プロパティが radiobutton1 であるラジオ ボタンを選択し、以前に選択されたラジオ ボタンを非選択とします。

メモ ボタン グループを用いて、ラジオ ボタンの排他的な選択を管理できます。詳細は、“ボタン グループ”(p.8-42)を参照してください。

チェック ボックス

Value プロパティの状態を確認することにより、コールバック内から、チェック ボックスの現在の状態を決定することができます。チェック ボックスが、チェックされると Value プロパティは Max プロパティに等しくなり、チェックされないと Min プロパティに等しくなります。これについて、次の例で説明します。

```
function checkbox1_Callback(hObject, eventdata, handles)
if (get(hObject, 'Value') == get(hObject, 'Max'))
    % Checkbox is checked-take appropriate action
else
    % Checkbox is not checked-take appropriate action
end
```

チェック ボックスの Value プロパティに、チェック ボックスの Max または Min プロパティの値を設定することによって、チェック ボックスの状態をプログラムによっても変更できます。この例は、そのような割り当てに対して可能な構文を説明します。

```
maxVal = get(handles.checkbox1, 'Max');
set(handles.checkbox1, 'Value', maxVal);
```

は、Tag プロパティが checkbox1 であるチェック ボックスを押された状態にします。

エディット テキスト

エディット ボックスでユーザーが入力した文字列を得るには、Callback コールバックで String プロパティを取得します。

```
function edittext1_Callback(hObject, eventdata, handles)
user_string = get(hObject,'String');
% Proceed with callback
```

エディット テキストの Max プロパティと Min プロパティの値を $\text{Max} - \text{Min} > 1$ であるように設定すると、ユーザーは複数行を入力できます。たとえば、Min を既定値 0 として Max を 2 に設定すると、ユーザーは複数行の入力が可能です。

エディット テキスト コンポーネントからの数値データの取得

MATLAB は、文字列として、エディット テキストの String プロパティの値を返します。数値の入力を希望する場合は、文字を数字に変換する必要があります。これは、コマンド str2double を使用して行うことができ、文字列を倍精度値に変換します。数字でない文字列を入力した場合、str2double は NaN を返します。

エディット テキスト コールバックにおいて、次のコードを使用できます。これは String プロパティの値を取得し、倍精度値に変換します。変換された値が、数字でない文字の入力を示す NaN であるかどうか (isnan) をチェックし、エラー ダイアログ (errordlg) を表示します。

```
function edittext1_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'String'));
if isnan(user_entry)
    errordlg(' You must enter a numeric value' , 'Bad Input' , 'modal')
    uicontrol(hObject)
    return
end
% Proceed with callback...
```

エディット テキスト コントロールは、(Return を入力するか、あるいはクリックすることで) ユーザーが送信して変更するときに、フォーカスがなくなります。行 uicontrol(hObject) は、エディット テキスト ボックスへのフォーカスを元の状態に戻します。これを行うことは、そのコールバックが動作するために必要ありませんが、ユーザー入力が検証できないイベントにおいては役立ちます。このコマンドは、エディット テキスト ボックスでテキストすべてを選択する効果があります。

コールバック実行のトリガー

エディット テキスト コンポーネントの内容が変更された場合、GUI 内でエディット テキストの外部をクリックすると、エディット テキスト コールバックが実行されます。ユーザーは、1 行のみのエディット テキストに対して Enter を押すか、あるいは複数行のエディット テキストに対して Ctrl+Enter を使用することもできます。

利用可能なキーボード アクセラレータ

GUI のユーザーは以下のキーボード アクセラレータを用いて、エディット テキストのコンテンツを修正できます。これらのアクセラレータは、変更可能ではありません。

- Ctrl+X – 切り取り
- Ctrl+C – コピー
- Ctrl+V – 貼り付け
- Ctrl+H – 最後の文字を削除する
- Ctrl+A – すべてを選択する

テーブル

テーブルには、数、文字データ、現在の選択（ドロップダウン メニュー）を含むことができます。各列には、同じタイプのデータが含まれなければなりません。テーブルまたはテーブル内の列は、エンド ユーザーにより編集可能であるようにすることができます。列の形式を指定したり、行と列に連続する番号を与えていたり、あるいはそれらを個々にラベルすることができます。行と列の数は、テーブルが表示するデータ行列のサイズを反映するように自動的に適用します。大部分のコンポーネント（ButtonDownFcn, DeleteFcn, KeypressFcn）に共通するコールバックをもつ他、テーブルは以下の特別なコールバックをもちます。

- CellEditCallback
- CellSelectionCallback

これらのコールバックは、テーブルに固有であり、以下で説明します。いずれもイベント データを与えます。

テーブルの CellEditCallbacks

テーブルはユーザーにより編集可能であれば（1 つ以上の列がその ColumnEditable プロパティを true に設定しているので）、CellEditCallback は、ユーザーがテーブ

ル セルの値を変更するたびに実行します。コールバックは、渡されたイベントデータを用いて、どのセルが変更されたか、セルの以前の値は何であったか、新しい値は何であるかを特定できます。たとえば、新しい値が有効であっても無効であっても(たとえば、個人の身長と体重を表す数は正でなければならない)コールバックはアクセスできます。このコールバックは、エラーを警告してから、無効な値を前の値で置き換えることができます。

テーブルの CellSelectionCallback

ユーザーがテーブルのセルを選択する度に、テーブルの CellSelectionCallback が実行します。これは、テーブルのセルが編集可能であっても、そうでなくとも起こります。セルが編集可能でない場合、セルの範囲全体をドラッグして、すべてのセルを選択できます。セルが編集可能である場合、ユーザーは、ドラッグではなく、Shift キーを押しながらクリック、または Ctrl キーを押しながらクリックして、一度に 2 つ以上のセルを選択できます。現在選択されたセルすべてのインデックスは、CellSelectionCallback eventdata 構造体に返されます。コールバックは選択が変更されるたびに実行し、新しいイベントデータが渡されます。

スライダー

次の例に示されるように、Value プロパティの状態を確認することによりコールバック内から、スライダーの現在の状態を決定できます。

```
function slider1_Callback(hObject, eventdata, handles)
    slider_value = get(hObject, 'Value');
    % Proceed with callback...
```

Max プロパティと Min プロパティは、スライダーの最大値と最小値を指定します。スライダーの範囲は、Max - Min です。

リストボックス

リストボックスの Callback コールバックがトリガーされると、リストボックスの Value プロパティは選択された項目のインデックスを含みます。ここで 1 はリストの最初の項目に対応します。String プロパティは、文字列のセル配列としてリストを含みます。

次の例では、選択された文字列を取得します。この例では、listbox1 は Tag プロパティの値であると仮定します。String プロパティから返された値をセル配列から文字列に変換する必要があることに注意してください。

```
function listbox1_Callback(hObject, eventdata, handles)
```

```

index_selected = get(hObject,'Value');
list = get(hObject,'String');
item_selected = list{index_selected}; % Convert from cell array
% to string

```

リスト ボックスの Value プロパティに表示したいリストを設定することによって、表示される項目のリストをプログラムでも指定できます。たとえば、

```
set(handles.listbox1,'Value',2)
```

は、Tag プロパティ listbox1 をもつリスト ボックスの 2 番目の項目を選択します。

コールバック実行のトリガー

MATLAB は、マウス ボタンがはなされたり、あるキーが押された後、リスト ボックスの Callback プロパティを用いたコールバックを実行します。

- 矢印キーは、Value プロパティを変更してコールバックの実行をトリガーし、Figure の SelectionType プロパティを normal に設定します。
- Enter キーとスペース キーは、Value プロパティを変更しませんが、コールバックの実行をトリガーし、Figure の SelectionType プロパティを open に設定します。

ユーザーがダブルクリックすると、コールバックは各クリックの後に実行します。

MATLAB は、Figure の SelectionType プロパティを最初のクリックで normal に設定し、2 回目のクリックで open に設定します。コールバックは、クリックが 1 回であるか 2 回であるかを Figure の SelectionType プロパティに確認し判定します。

リスト ボックスの例

リスト ボックスを利用した詳細については、以下の例を参照してください。

- “リスト ボックス ディレクトリ リーダー” (p.10-54) — リスト ボックスにディレクトリの内容を表示する GUI を生成する方法を示します。これにより、ファイル名をダブルクリックして、様々なファイル タイプを開くことができるようになります。
- “リスト ボックスからワークスペース変数にアクセス” (p.10-61) — リスト ボックスの GUI から、MATLAB ベース ワークスペース内の変数にアクセスする方法を示します。

ポップアップ メニュー

ポップアップ メニュー の Callback コールバックがトリガーされると、ポップアップ メニュー の Value プロパティは、選択された項目のインデックスを含みます。ここで 1 はメニューの最初の項目に対応します。String プロパティは、文字列のセル配列としてメニュー項目を含みます。

メモ ポップアップ メニューは、ドロップダウン メニューまたはコンボ ボックスと呼ばれることがあります。

選択されたメニュー項目のインデックスのみの利用

この例では、選択された項目のインデックスのみを取得します。これは、switch ステートメントを使用して値に基づいたアクションをとります。ポップアップ メニューのコンテンツが固定している場合には、この方法を使用できます。そうでない場合、選択された項目に対する実際の文字列を取得するためにインデックスを使用できます。

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
switch val
case 1
% User selected the first item
case 2
% User selected the second item
% Proceed with callback...
```

ポップアップ メニューの Value プロパティに表示したいリストのインデックスを設定することによって、表示される項目のリストをプログラムでも指定できます。たとえば、

```
set(handles.popupmenu1,'Value',2)
```

は、Tag プロパティ popupmenu1 をもつポップアップ メニューの 2 番目の項目を選択します。

選択された文字列を判別するインデックスの使用

この例は、ポップアップ メニューで選択された実際の文字列を取得します。これは、ポップアップ メニューの Value プロパティを使用して文字列のリストにインデックスを付けます。この方法は、プログラムがユーザーのアクションに基づきポップアップ メニューの内容を動的に読み込み、選択した文字列を得る必要がある場合に有

効です。セル配列から String プロパティにより返される値を文字列に変換するこ
とが必要ですので、注意してください。

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
string_list = get(hObject,'String');
selected_string = string_list{val}; % Convert from cell array
% to string
% Proceed with callback...
```

パネル

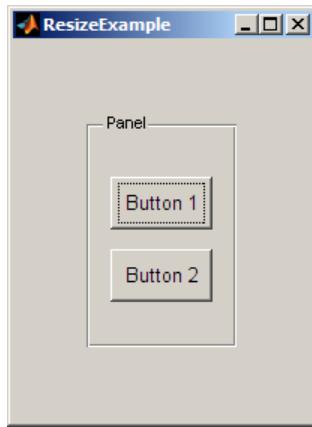
パネルは GUI コンポーネントをグループ化し、関連するコントロールを視覚的にグ
ループ化することにより、ユーザー インターフェイスの理解を容易にします。パネル
は、プッシュ ボタン、スライダー、ポップアップ メニューなどのユーザー インターフェイ
ス コントロールや座標軸と同様に、パネルとボタン グループを含むことができます。
パネル内での各コンポーネントの位置は、パネルの左下隅を基準に求められます。

一般に、GUI がサイズ変更されると、パネルとそのコンポーネントもサイズ変更され
ます。ただし、パネルとそのコンポーネントのサイズと位置をコントロールできます。
GUI の [サイズ変更動作] を [その他 (ResizeFcnを使用)] に設定し、パネルに対して
ResizeFcn コールバックを与えると、これを行うことができます。

メモ Figure に対する [サイズ変更動作] を その他 (ResizeFcnを使用) に設定する場
合、レイアウト エディターの [ツール] メニューから [GUI オプション] を選択します。サ
イズ変更の動作への単位の影響について詳細は、“クロスプラットフォーム互換の
Units” (p.6-137) を参照してください。

Figure の [サイズ変更動作] が [その他 (ResizeFcn 使用)] であっても、コンポーネ
ントが正規化された Units を追加すると、ResizeFcn がその動作をオーバーライドしな
い限り、コンポーネントは自動的に比例的にサイズ変更します。次の例では、それ以
上のことをするために ResizeFcn をどのように使用できるか示します。GUI はコン
ポーネントを自動的に再配置します。そのパネルの ResizeFcn は、ボタン ラベル
の fontSize を比例的に調整します。

- 1 GUIDE で、パネル内に 2 つのプッシュ ボタンがあるパネルを含む GUI を作成
します。プロパティ インスペクターにおいて、ボタン [Button 1] と [Button 2] に
名前を付けます。Figure の Units を pixels に、その Position を [420 520 150
190] に設定します。GUI は、次のようになります。



2 2つのプッシュボタンに対してコールバックを作成して、それぞれに以下のコード行を置きます。

```
set(gcbf, 'Position', [420 520 150 190])
```

これは、GUI を初期のサイズにリセットするので、手動でサイズ変更を試みることができます。

3 プロパティインスペクターにおいて、パネルと 2 つのボタンの Units を normalized に設定します。さらに、両方のボタンの fontSize を 10 に設定します。両方のボタンの fontUnits プロパティは、points に設定されることを確かめてください。

4 プロパティ インスペクターで `ResizeFcn` のための鉛筆アイコンをクリックすることで、パネルに対する `ResizeFcn` コールバックを作成し、プロパティ インスペクターに以下のコードを挿入します。

```

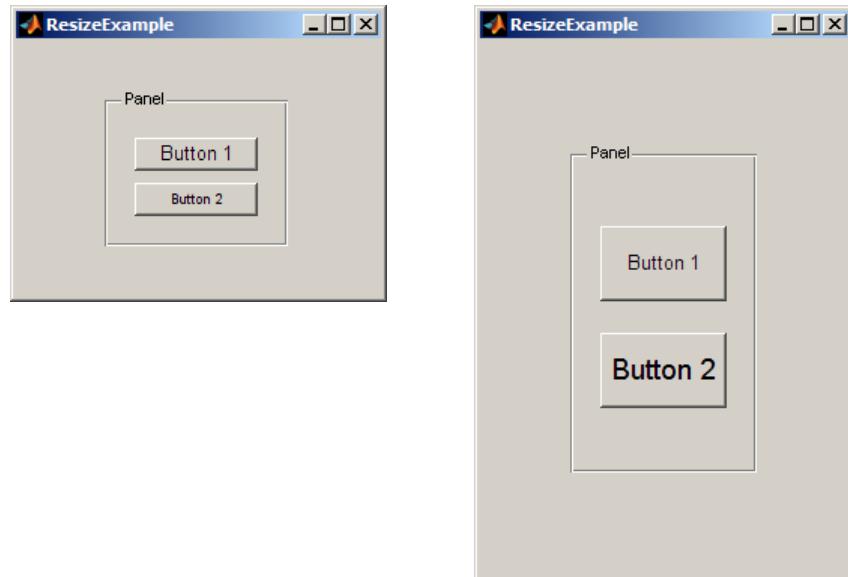
function uipanel1_ResizeFcn(hObject, eventdata, handles)
%
%
% set(hObject,'Units','Points') % Was normalized
panelSizePts = get(hObject,'Position'); % Now in points
panelHeight = panelSizePts(4);
set(hObject,'Units','normalized'); % Now normalized again
% Keep fontsize in constant ratio to height of panel
newFontSize = 10 * panelHeight / 115; % Calculated in points

```

```
buttons = get(hObject,'Children');
set(buttons(1),'FontSize',newFontSize); % Resize the first button
% Do not resize the other button for comparison
```

このコードは、Figure がサイズ変更したとき、ボタンラベルの 1 つ（この例では、下にあるボタン）のサイズを調整します。コードでは、パネルの現在のサイズと元のサイズ（ポイントで表される）の比に、元のボタンの fontSize、10 ポイントを乗算して newFontSize を計算します。この場合、ボタンの fontSize の 1 つがその値に設定されます。

5 GUI を実行すると、GUI は前の図のようになります。その大きさを小さくしたり大きくしたりサイズ変更すると、次の図に示すように一方のボタンのテキストは縮小または拡大されます。



いずれかのボタンをクリックすると、GUI とその 2 つのボタンは元の大きさに戻ります。すべての Units は normalized なので、比例的にサイズ変更するために他のコードは必要ありません。

ヒント `ResizeFcn` を必要とせずに、コンポーネントの `fontUnits` を `normalized` に設定することで自動的にサイズ変更するため、テキストをコントロールできます。この例は、コールバックコードと同じ結果を達成する 1 つの方法を説明します。

入れ子されたパネルは、内側から外側へ（子から親への順に）サイズ変更します。パネルのサイズ変更について詳細は、「`uipanel` プロパティ」のリファレンスページを参照してください。

ボタン グループ

ボタン グループはパネルに似ていますが、ボタン グループはラジオ ボタンやトグル ボタンに対する排他的な選択を管理するために使用されます。ボタン グループが、ラジオ ボタン、トグル ボタン、またはその両方のセットを含む場合、ボタン グループはこれらのうちの 1 つのみが選択されることを許可します。ユーザーがボタンをクリックすると、そのボタンが選択され、他のすべてのボタンは非選択になります。

ボタン グループをプログラミングするとき、個々のボタンに対してコールバックをコーディングしません。代わりに、その `SelectionChangeFcn` コールバックを使用して、選択に対して応答を管理します。次の例、“ボタン グループのプログラミング”(p.8-43) は、これを行うために `uibuttongroup` のイベントデータを使用する方法を説明します。

次の図は、2 つのラジオ ボタンと 2 つのトグル ボタンをもつボタン グループを示します。[Radio Button 1] が選択されます。



ユーザーがもう一方のラジオ ボタンまたはトグル ボタンの 1 つをクリックすると、こちらが選択された状態になり、Radio Button 1 が非選択になります。次の図は、[ToggleButton 2] をクリックした結果を示します。



ボタン グループの SelectionChangeFcn コールバックは、選択が行なわれるときに呼び出されます。この hObject 入力引数は、選択されたラジオ ボタンまたはトグル ボタンのハンドルを含みます。

ラジオ ボタンとトグル ボタンのセットを含むボタン グループをもつ場合、以下を行いたいとします。

- ラジオ ボタンまたはトグル ボタンが選択されたときに直ちに動作が起こるよう に、個々のトグル ボタンの Callback 関数においてではなく、ボタン グループの SelectionChangeFcn コールバック関数において、ラジオ ボタンをコントロールする コードを含む必要があります。
- 選択に基づきアクションをするプッシュ ボタンのような他のコンポーネントでは、そ のコンポーネントの Callback コールバックが、ボタン グループの SelectedObject プロパティから、選択されたラジオ ボタンまたはトグル ボタンのハンドルを取 得できます。

ボタン グループのプログラミング

SelectionChangeFcn コールバックのこの例は、選択されたオブジェクトの Tag プロパティを用いて、実行する適切なコードを選択します。各コンポーネントの Tag プロパティは、そのコンポーネントを識別する文字列で、GUI で一意でなければなりません。

```
function uibuttongroup1_SelectionChangeFcn(hObject, eventdata)
switch get(eventdata.NewValue, 'Tag') % Get Tag of selected object.
    case 'radiobutton1'
```

```
% Code for when radiobutton1 is selected.  
case 'radiobutton2'  
    % Code for when radiobutton2 is selected.  
case 'togglebutton1'  
    % Code for when togglebutton1 is selected.  
case 'togglebutton2'  
    % Code for when togglebutton2 is selected.  
% Continue with more cases as necessary.  
otherwise  
    % Code for when there is no match.  
end
```

コールバック プロパティの値が関数ハンドルとして指定される場合に限り、`hObject` と `eventdata` 引数はコールバックに対して利用可能です。`eventdata` の詳細は、`Uibuttongroup` プロパティのリファレンス ページの「`SelectionChangeFcn`」プロパティを参照してください。他の例は、`uibuttongroup` のリファレンス ページと“カラー パレット”(p.15-49)を参照してください。

座標軸

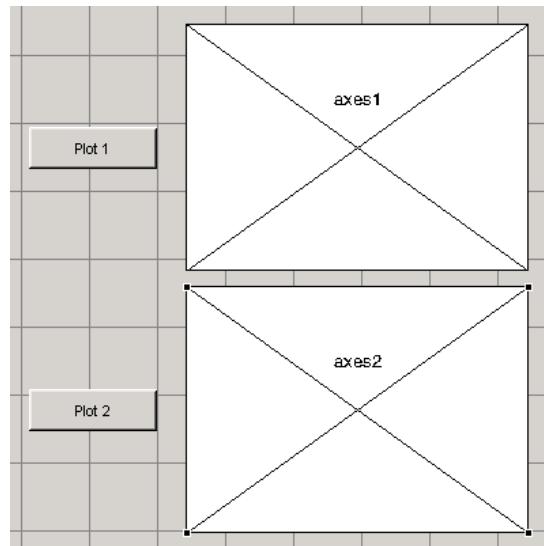
座標軸により、ユーザー GUI が、グラフィックス(たとえば、グラフやイメージ)を表示することが可能になります。以下のトピックは、ユーザー GUI に座標軸コンポーネントをプロットする方法を簡単に説明します。

- ・ “座標軸へのプロット”(p.8-44)
- ・ “サブプロットの作成”(p.8-47)

座標軸へのプロット

多くの場合、GUI の座標軸以外のコンポーネントに属するコールバックから、座標軸にプロットを作成します。たとえば、ボタンを押すと座標軸にグラフがプロットされます。この場合、ボタンの `Callback` コールバックはプロットを生成するコードを含みます。

次の例は、2つの座標軸と2つのボタンを含みます。一方のボタンをクリックすると、一方の座標軸にプロットを作成し、他方のボタンをクリックすると、他方の座標軸にプロットを作成します。次の図は、レイアウトエディターに表示されたこれらのコンポーネントを示します。



1 [Plot 1] プッシュボタンの Callback コールバックに次のコードを追加します。関数 `surf` は 3 次元の影付き表面プロットを作成します。関数 `peaks` は、ガウス分布の平行移動とスケーリングにより得られる正方形行列を返します。

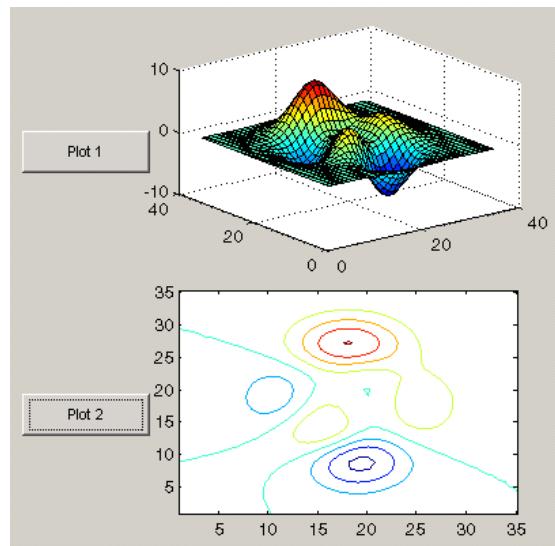
```
surf(handles.axes1, peaks(35));
```

2 [Plot 2] プッシュボタンの Callback コールバックに次のコードを追加します。関数 `contour` は、行列の等高線図を表示します。この場合、`peaks` の出力を表示します。

```
contour(handles.axes2, peaks(35));
```

3 [ツール] メニューから [実行] を選択して GUI を実行します。

4 [Plot 1] ボタンをクリックして、第 1 軸に表面プロットを表示します。[Plot 2] ボタンをクリックして、第 2 軸に等高線図を表示します。



2つの座標軸を使用するより複雑な例については、“複数の座標軸をもつ GUI”(p.10-2)を参照してください。

メモ 座標軸の動作と外見の多くの特性をコントロールするために、設定できるプロパティについての詳細は、MATLAB「グラフィックス」ドキュメンテーションの“Axes Properties”を参照してください。一般的のプロットに関する詳細は、MATLAB「グラフィックス」ドキュメンテーションの“Plots and Plotting Tools”を参照してください。

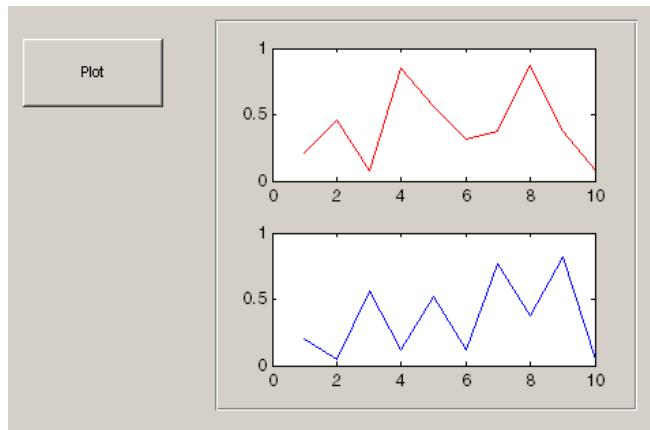
座標軸を含む GUI では、[GUI オプション] ダイアログにおける [コマンド ラインからのアクセス] オプションが、[Callback] (既定値) に設定されることを確かめる必要があります。レイアウトエディターから、[ツール] > [GUI オプション] > [コマンド ラインへのアクセス:コールバック] を選択します。このオプションの動作に関する詳細は、“コマンド ラインからのアクセス”(p.5-10)を参照してください。

サブプロットの作成

関数 `subplot` を使用して、タイル模様で座標軸を作成します。GUIDE が作成した GUI がサブプロット以外のコンポーネントを含む場合、このサブプロットはパネルに含まれなければなりません。

例として、次のコードは、関数 `subplot` を使用して、Tag プロパティ `uipanel1` をもつパネルに 2 つのサブプロットをもつ座標軸を作成します。このコードは、[Plot] プッシュボタンの Callback コールバックの一部です。[Plot] ボタンを押す度に、コードは各サブプロット内に線を描きます。`a1` と `a2` は、サブプロットのハンドルです。

```
a1=subplot(2,1,1,'Parent',handles.uipanel1);
plot(a1,rand(1,10),'r');
a2=subplot(2,1,2,'Parent',handles.uipanel1);
plot(a2,rand(1,10),'b');
```



ヒント 複数の座標軸を取り扱っている場合、データをプロットしようとしている座標軸を、次のようなコマンドで“上げる”ことはしない方が良いでしょう。

```
axes(a1)
```

これにより、座標軸 a1 が現在の座標軸になりますが、これは Figure をリストックしたり、すべての未解決のイベントを書き出します。これは、コンピューターの資源を消費しますが、コールバックの実行に必要になることはほとんどありません。以下のように、ユーザーが呼び出しているプロット関数の最初の引数として単に座標軸のハンドルを与えると、より効率的です。

```
plot(a1, ...)
```

これは、Figure をリストックしたり、あるいはキューされたイベントを書き出さずに、座標軸 a1 にグラフィックスを出力します。関数 line のように Axes のハンドルを引数としないプロット関数に対して座標軸を指定する場合、以下のように a1 を現在の座標軸にすることができます。

```
set.figure_handle,'CurrentAxes',a1)  
line(x,y,z,...)
```

詳細は、「Figure プロパティ」のリファレンス ページの CurrentAxes の説明を参照してください。

サブプロットの詳細は、subplot のリファレンス ページを参照してください。ユーザー GUI へのパネルの追加についての詳細は、“GUIDE レイアウト エリアへのコンポーネントの追加”(p.6-31)を参照してください。

ActiveX コントロール

この例は、サンプルの ActiveX コントロール Mwsamp Control をプログラムします。この例では、最初に円をクリックして円の半径を変更できます。次に、同じことをするために GUI のスライダーをプログラムします。

- ・ “ActiveX コントロールのプログラミング”(p.8-49)
- ・ “ActiveX コントロールを更新するユーザー インターフェイス コントロールのプログラミング”(p.8-54)

このトピックスは、以下についても説明します。

- ・ “ActiveX コントロールに対するメソッドの表示” (p.8-56)
- ・ “ActiveX コントロール を含む GUI の保存” (p.8-57)
- ・ “ActiveX コントロール を含む GUI のコンパイル” (p.8-58)

ActiveX コントロールの詳細は、MATLAB「外部インターフェイス」ドキュメンテーションの“Creating COM Objects”を参照してください。

メモ GUIDE により Figure がサイズ変更可能である場合、ActiveX コントロールは自動的にサイズ変更できます。GUIDE の外部で ActiveX コントロールをもつ GUI を作成していると、MATLAB「外部インターフェイス」ドキュメンテーションの“Example – Using Internet Explorer® Program in a MATLAB Figure”に述べるサイズ変更の手法を使用できます。

ActiveX コントロールのプログラミング

サンプルの ActiveX コントロール Mwsamp Control は、四角の中央に円を含みます。この例は、ユーザーが円をクリックするときに円の半径を変更し、新しい半径を表示するラベルを更新するコントロールをプログラムします。

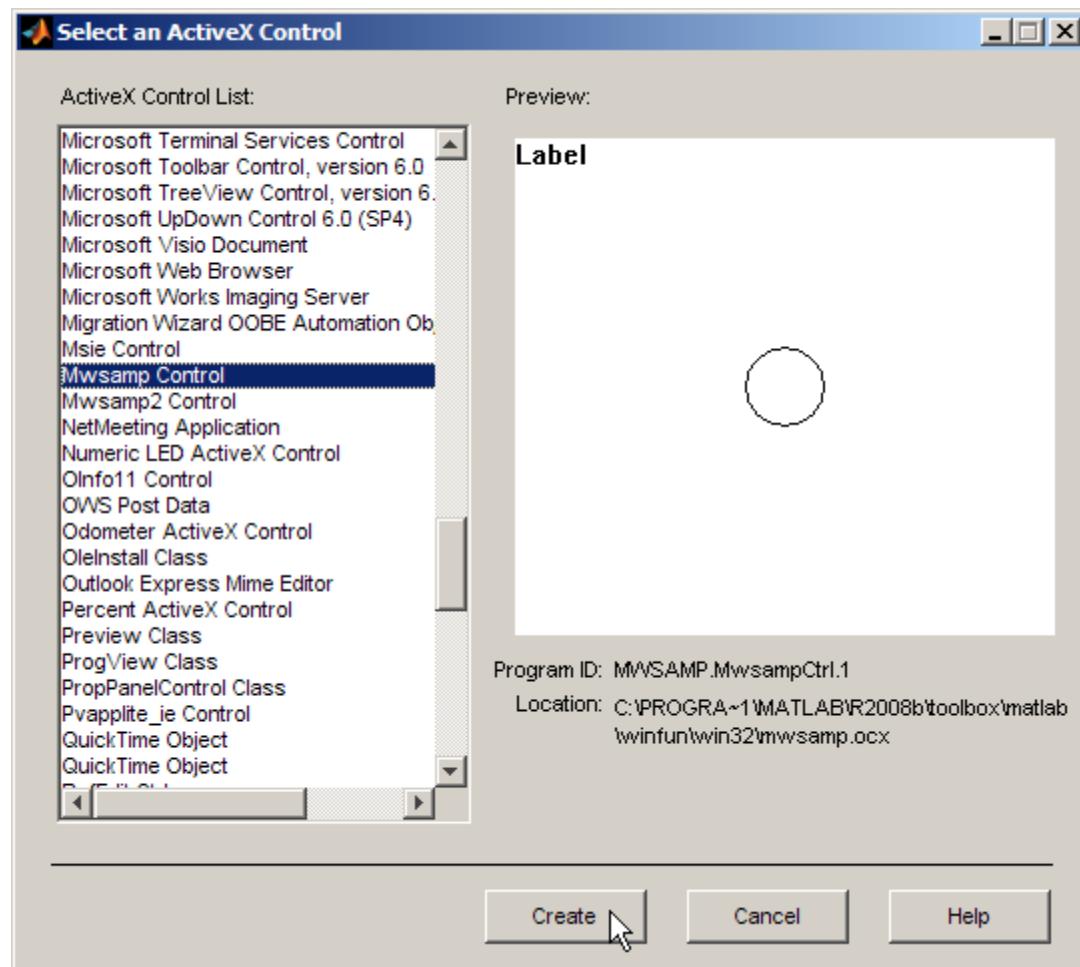
MATLAB ヘルプ ブラウザーでこれを読み頂いている場合、以下のリンクをクリックして以下の例の完成したバージョンを GUIDE レイアウトエディターと MATLAB エディターに表示できます。

メモ 以下のリンクは、MATLAB コマンドを実行し、MATLAB ヘルプ ブラウザー内で動作するように設計されています。オンラインあるいは PDF でこれを読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

- ・ レイアウトエディターに Mwsamp の GUI を表示するには、ここをクリックしてください。.
- ・ MATLAB エディターに Mwsamp の GUI M ファイルを表示するには、ここをクリックしてください。

例の GUI またはその M ファイルを修正し、ユーザーによる変更を保持したい場合、レイアウトエディターで [ファイル] > [別名で保存] を使用し、書き込み権限のあるフォルダーにファイルを保存します。

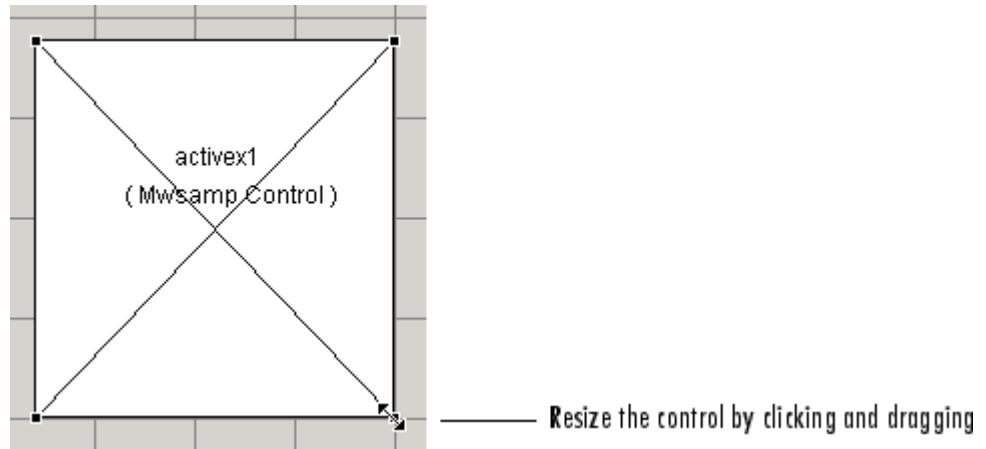
- 1 サンプルの ActiveX コントロールを追加するには、GUIDE レイアウトエディターの ActiveX ツール  をクリックして、それを含めるためのエリアをドラッグ アウトします。ダイアログ ボックスが開きます。
- 2 ダイアログ ボックスの左側のリスト ボックスから、[Mwsamp Control] を選択します。そのプレビューが右側に現れます。



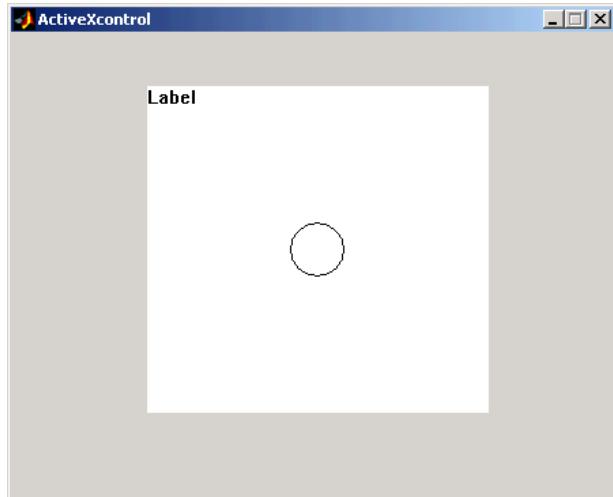
メモ [作成] をクリックすると、ユーザーの作業フォルダーにファイル Mwsamp_activedex1 のコピーが配置されます。GUI ファイルを他のフォルダーに移動させると、GUI ファイルが使用している ActiveX コントロールを移動する必要があります。

- 3 [作成] をクリックして、サンプルの ActiveX コントロール Mwsamp を GUI に追加し、プレビュー ペインに示す四角に近いサイズまでサイズ変更します。次の図は、レイアウト エディターに表示されるときの ActiveX コントロールを示します。

コンポーネントの追加についてのヘルプが必要な場合、“ActiveX コントロールの追加”(p.6-75)を参照してください。

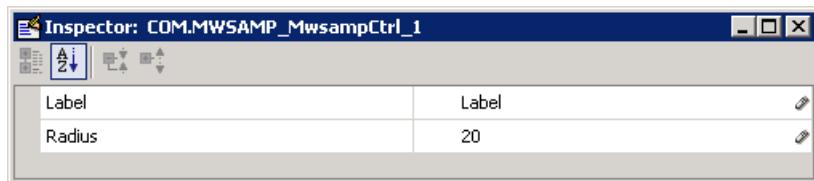


- 4 ツールバーの■ボタンをクリックして GUI をアクティブにし、指示に従い GUI を保存します。GUIDE は、次の図に示す GUI を表示し、GUI M ファイルを開きます。



- 5 プロパティインスペクターで ActiveX プロパティを表示します。レイアウトエディターでコントロールを選択し、[表示] メニューから、またはツールバーの [プロパティインスペクター] ボタン をクリックして [プロパティインスペクター] を選択します。

次の図は、プロパティインスペクターに表示される mwsamp ActiveX コントロールのプロパティを示します。ユーザーシステムのプロパティは、異なることがあります。



この ActiveX コントロール mwsamp は、2 つのプロパティをもちます。

- ・ Label、コントロールの一番上に表示されるテキストを含みます。
- ・ Radius、円の既定値の半径、既定値は 20 です。

- 6 Click カールバックを、GUI M ファイルに置きます。[表示] メニューから [カールバックの表示] を選択してから、[Click] を選択します。

GUIDE は、GUI M ファイルの最後に、新規のカールバック テンプレート activex1_Click を追加します。

- 7 以下のコードを mswamp コントロールの activex1_Click コールバックに追加します。このコードは、ActiveX コントロールをプログラムして、ユーザーが円をクリックするときに円の半径を変更し、新しい半径を表示するためにラベルを更新します。

```
hObject.radius = floor(.9*hObject.radius);
hObject.label = [ ' Radius = ' num2str(hObject. radius)];
refresh(handles.figure1);
```

- 8 以下のコマンドを、opening 関数 Mwsamp_OpeningFcn の最後に追加します。このコードは、ユーザーが最初に GUI を開くときにラベルを初期化します。

```
handles.activex1.label = ...
[ ' Radius = ' num2str(handles.activex1.radius)];
```

M ファイルを保存します。ここで、ユーザーが GUI を開いて ActiveX コントロールをクリックする度に、円の半径が 10 パーセント減少し、新しい値の半径が表示されます。次の図は、円を 6 回クリックした後の GUI を示します。



GUI を十分な回数クリックすると、円は消えます。

ActiveX コントロールを更新するユーザー インターフェイス コントロールのプログラミング

このトピックは、GUI にスライダーを追加し、円の半径を変更するようにスライダーをプログラミングすることによって前の例を続けます。この例は、ユーザーが円をクリックするとスライダーも更新する必要があります。

- 1 レイアウトにスライダーを追加してから、slider1_Callback コールバックに次のコードを追加します。

```
handles.activex1.radius = ...
get(hObject,'Value')*handles.default_radius;
handles.activex1.label = ...
['Radius = ' num2str(handles.activex1.radius)];
refresh(handles.figure1);
```

最初のコマンドは、以下のことを行います。

- スライダーの Value を取得します。この例では、スライダーの Min と Max プロパティの既定値、0 と 1 の間の値です。
- handles.activex1.radius を既定値の Value 倍に設定します。

- 2 opening 関数において、handles 構造体に既定値の半径を追加します。

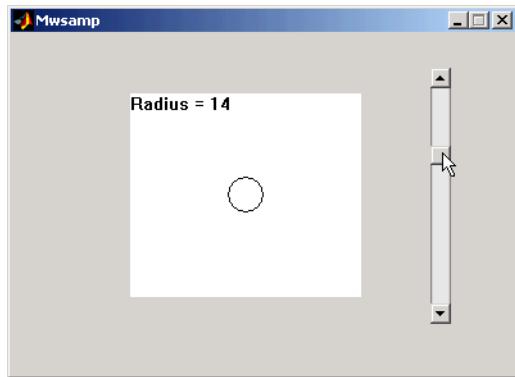
activex1_Click コールバックは、ユーザーが円をクリックすると、既定値の半径を使用してスライダーの値を更新します。

```
handles.default_radius = handles.activex1.radius;
```

- 3 activex1_Click コールバックでは、ユーザーが ActiveX コントロールの円をクリックする度に、スライダーの Value をリセットします。次のコマンドは、スライダーが新しい値の半径に対応するように位置を変更します。

```
set(handles.slider1,'Value',...
hObject.radius/handles.default_radius);
```

GUI を開いてスライダーをクリックしながら移動すると、次の図に示すように、半径が 0 と既定値 20 の間で変化します。



ActiveX コントロールに対するメソッドの表示

ActiveX コントロールに対して利用できるメソッドを表示するには、最初にコントロールに対するハンドルを取得する必要があります。これを行う方法の1つとして、以下の方法があります。

- 1 GUI M ファイルで、activex1_Click コールバックの別の行にコマンド keyboard を追加します。コマンド keyboard が MATLAB をデバッグ モードにし、ユーザーが ActiveX コントロールをクリックするとき、activex1_Click コールバックで止まります。
- 2 GUI を実行し、ActiveX コントロールをクリックします。ここで、コントロールのハンドル番号は、hObject に設定されています。
- 3 コントロールに対するメソッドを表示するには、MATLAB プロンプトで、次のように入力します。

```
methodsview(hObject)
```

これにより、次の図に示すように、新しいウィンドウに利用できるメソッドが表示されます。

Return Type	Name	Arguments	Inherited From
	AboutBox	(handle)	COM.mwsamp.mwsampctrl.1
	Beep	(handle)	COM.mwsamp.mwsampctrl.1
	FireClickEvent	(handle)	COM.mwsamp.mwsampctrl.1
string	GetBSTR	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetBSTRArray	(handle)	COM.mwsamp.mwsampctrl.1
int32	GetI4	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetI4Array	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetI4Vector	(handle)	COM.mwsamp.mwsampctrl.1
handle	GetDispatch	(handle)	COM.mwsamp.mwsampctrl.1
double	GetR8	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetR8Array	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetR8Vector	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetVariantArray	(handle)	COM.mwsamp.mwsampctrl.1
Variant	GetVariantVector	(handle)	COM.mwsamp.mwsampctrl.1
	Redraw	(handle)	COM.mwsamp.mwsampctrl.1
string	SetBSTR	(handle, string)	COM.mwsamp.mwsampctrl.1
Variant	SetBSTRArray	(handle, Variant)	COM.mwsamp.mwsampctrl.1
int32	SetI4	(handle, int32)	COM.mwsamp.mwsampctrl.1
Variant	SetI4Array	(handle, Variant)	COM.mwsamp.mwsampctrl.1
Variant	SetI4Vector	(handle, Variant)	COM.mwsamp.mwsampctrl.1

あるいは、次のように入力することもできます。

```
methods (hObject)
```

これにより、MATLAB コマンド ウィンドウに利用できるメソッドが表示されます。

ActiveX コントロールに対するメソッドの詳細は、「外部インターフェイス」ドキュメンテーションで“Using Methods”について参照してください。これらの関数の詳細は、`methodsview` と `methods` のリファレンス ページを参照してください。

ActiveX コントロール を含む GUI の保存

ActiveX コントロールを含む GUI を保存すると、GUIDE は各コントロールに対して、現在のフォルダーにファイルを作成します。ファイル名は、アンダーライン (_) と `activexn` が続く GUI の名前で構成されます。ここで、`n` は連続した番号です。たとえば、GUI が `mygui` という名前の場合、ファイル名は `mygui_activedx1` となります。このファイル名は、拡張子をもちません。

ActiveX コントロールを含む GUI のコンパイル

MATLAB Compiler の `mcc` コマンドを使用して ActiveX コントロールを含む GUI をコンパイルする場合には、GUIDE によって現在のフォルダーに保存される ActiveX ファイルを、`-a` フラグを用いて CTF アーカイブに追加する必要があります。コマンドは、次のようにになります。

```
mcc -m mygui -a mygui_activex1
```

ここで、`mygui_activex1` は ActiveX ファイルの名前です。詳細は、“MATLAB Compiler™” ドキュメンテーションを参照してください。このようなファイルが 2 つ以上ある場合は、各ファイルについて別々に `-a` フラグを使用してください。GUI をコンパイルするには、MATLAB Compiler をインストールしている必要があります。

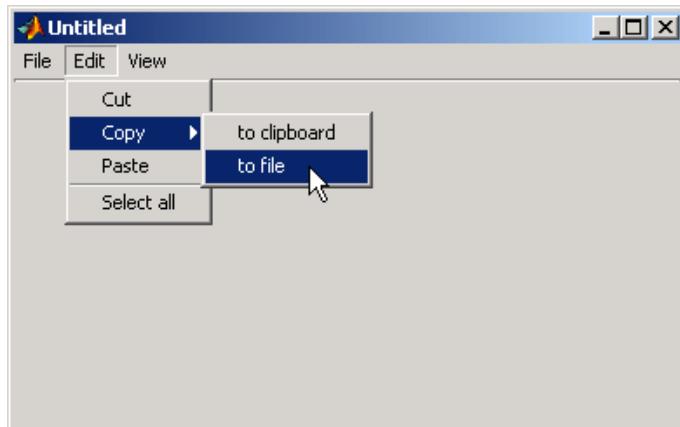
メニュー項目

メニュー エディターは、メニュー タイトルを含む各メニュー項目に対して、空のコールバック サブファンクションを生成します。

メニュー タイトルのプログラミング

メニュー タイトルをクリックすると、サブメニューが自動的に表示されるので、ユーザーは、タイトル レベルのメニューに対するコールバックをプログラムする必要はありません。しかし、メニュー タイトルに関連するコールバックは、サブメニューを有効にしたり無効にするために使用できます。

次の図に示す例を考えます。



ユーザーが、[Edit] メニューの [Copy] 項目の下で、[to file] オプションを選択するときには、アクションを実行するために to file コールバックのみが必要になります。

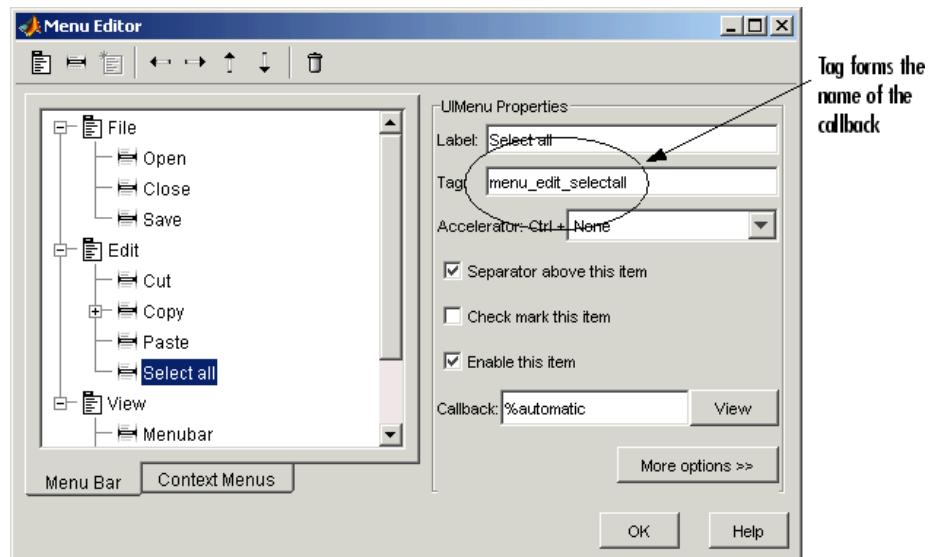
ここでは、あるオブジェクトのみがファイルにコピーされると仮定します。ユーザーは [Copy] 項目の Callback コールバックを使用して、選択されるオブジェクトのタイプに依存して、[to file] 項目を使用可能、あるいは使用不可能にすることができます。

メニュー コールバックからダイアログ ボックスを開く

[to file] メニュー項目に対する Callback コールバックは、次のようなコードを含み、ファイルを保存するための標準のダイアログ ボックスを表示します。

```
[file,path] = uiputfile('animinit.m','Save file name');
```

'Save file name' はダイアログ ボックス タイトルです。ダイアログ ボックスでは、ファイル名のフィールドに animinit.m が設定され、フィルタに M files (*.m) が設定されます。詳細は、uiputfile のリファレンス ページを参照してください。



メニュー項目のチェックの更新

チェックはメニュー項目の現在の状態を表すのに非常に有効です。メニュー エディターで [項目をチェック] を選択すると、その項目は最初チェックされた状態になります。ユーザーがメニュー項目を選択する度に、その項目に対するコールバックはチェックを on あるいは off にする必要があります。次の例は、メニュー項目の Checked プロパティの値を変更することでこれを行う方法を示します。

```
if strcmp(get(hObject,'Checked'),'on')
    set(hObject,'Checked','off');
else
    set(hObject,'Checked','on');
end
```

`(hObject` はコールバックされたコンポーネントのハンドルです。関数 `strcmp` は 2 つの文字列を比較し、同じ場合には論理値 1 (`true`) を返し、そうでない場合は論理値 0 (`false`) を返します。

GUI が最初に表示されるときのチェックの利用は、表示と一致しなければなりません。たとえば、ユーザー GUI に、最初に GUI を開いたときに表示される座標軸があり、GUI が [Show axes] メニュー項目をもつとき、メニュー項目を作成するときにメニュー項目の Checked プロパティを必ず on に設定して、[Show axes] メニュー項目の隣に最初にチェックが現れるようにします。

メモ メニュー エディターから、メニュー項目を選択して [表示] ボタンをクリックして、エディター内にメニュー項目の Callback コールバックを表示できます。

GUIDE でのアプリケーション データの管理と共有

- ・ “データ管理の仕組み” (p.9-2)
- ・ “複数の GUI を連携して動作させる方法” (p.9-22)

データ管理の仕組み

このセクションの内容...
“概要” (p.9-2)
“入れ子関数” (p.9-4)
“UserData プロパティ” (p.9-4)
“アプリケーション データ” (p.9-5)
“GUI データ” (p.9-7)
“GUI のコールバック間でデータを共有する例” (p.9-10)

概要

大部分の GUI は、アプリケーションに固有のデータを作成または使用します。GUI コンポーネントは、互いにデータをやりとりする必要がある場合があります。そのために、役立つ基本的な仕組みがいくつかあります。

多くの GUI は単独の Figure ですが、ユーザー アプリケーションが 2 つ以上の Figure を必要とする場合には、複数の GUI を同時に機能させることができます。たとえば、GUI で使用されるパラメーターをいくつか表示して取得するために、GUI は複数のダイアログ ボックスを必要とします。ユーザー GUI は、同時にまたは連続して、同時に機能する個々のツールを複数含むことができます。ファイルまたはワープスペース変数でのやりとりを回避するために、以下の表で述べるメソッドのいずれかを使用できます。

データ共有方法	動作	利用
プロパティ/値の組	入力引数として渡すことで、新たに呼び出された GUI または既存の GUI にデータを送ります。	新規の GUI にデータを送ります。
出力	起動した GUI からデータを出力します。	起動された GUI のハンドル構造体を戻すなど、起動された GUI にデータを戻します

データ共有方法	動作	利用
関数ハンドルまたは private のデータ	以下の 4 つのメソッドのいずれか 1 つで、関数ハンドルまたはデータを渡します。	GUI 内または GUI 間で機能を表す
	“Nested Functions”: すべての上位の関数の名前空間を共有します	通常は、単一の GUI の Figure 内の、直接的または間接的に囲まれている関数で定義された変数へのアクセスと修正
	UserData:この Figure またはコンポーネント プロパティにデータを格納し、ハンドルの参照を利用して他の GUI に渡します。	GUI 内または GUI 間でのデータのやりとり。UserData は、構造体として渡される場合が多く、1 つの変数に制限されます
	アプリケーションデータ (getappdata と setappdata):Figure またはコンポーネントに名前の付いたデータを格納し、ハンドルの参照を利用して他の GUI に渡します。	GUI 内または GUI 間でのデータのやりとり。変数の数やタイプは、この API を利用してアプリケーションデータとして格納できます
	guidata:GUI の handles 構造体にデータを格納し、ハンドルの参照を利用して他の GUI に渡します。	GUI 内または GUI 間でのデータのやりとり – アプリケーションデータを管理する簡単な方法。GUI データは、GUI の Figure に関連する構造体です。

例 “アイコン エディター” (p.15-60) では、さらに GUI の Figure 間でのデータの共有を説明します。

以下の 3 つの節では、GUI 内に格納されたアプリケーション定義のデータを管理する方法を与える仕組みについて説明します。

- 入れ子関数 – より高いレベルで定義された変数を共有し、呼び出された関数が、上位か下位、または呼び出し側の兄弟である場合、互いに呼び出します。

- UserData プロパティ – GUI コンポーネントに割り当て、その他のプロパティのように取り出す MATLAB ワークスペース変数。
- アプリケーション データ – アプリケーションが、指定したオブジェクトに関連するデータを格納し取得するために利用します。GUI に対し、この特定のオブジェクトとは通常 GUI の Figure ですが、任意のコンポーネントでも可能です。データは、名前/値の組として保存されます。アプリケーション データを使うと、オブジェクトに対するユーザー定義のプロパティの作成が可能になります。
- GUI データ – GUI データを管理するために関数 `guidata` を使用する。この関数は MATLAB 構造体に GUI データとして 1 つの変数を格納できます。この構造体は GUIDE ではハンドル構造体と呼ばれます。ハンドル構造体を更新したり、取得したりするために関数を使用します。

“GUI のコールバック間でデータを共有する例”(p.9-10)において、簡単な作業 GUI に適用される 3 つの方法を比較できます。

入れ子関数

GUI M ファイルに入れ子関数を置くと、コールバック関数は、データを引数として渡される必要がなくなり、データを自由に共有することができます。

- 1 ユーザー コードの初期化セグメントでは、コンポーネントの作成、変数の定義、データの生成をします。
- 2 GUI コールバックとユーティリティ関数を初期化の下のレベルに入れ子します。

データとコンポーネント ハンドルは、より上のレベルで定義されているので、コールバックとユーティリティ関数はこれらへのアクセスを自動的にもちます。この手法を利用すると、UserData、アプリケーション データ、または多くのインスタンスの GUI データを格納する必要がなくなります。

メモ 入れ子関数の利用に適用される規則と制約については、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”を参照してください。

UserData プロパティ

メニュー や Figure 自体を含む、すべての GUI コンポーネントは UserData プロパティをもらいます。UserData プロパティの値として MATLAB ワークスペースの有効な値を割り当てることができますが、一度に存在できる値は 1 つに限られます。そのデータを

取得するには、そのデータが格納されているコールバックがコンポーネントのハンドルを知る必要があります。適切なオブジェクトのハンドルを用いて `get` と `set` を利用して、`UserData` にアクセスします。以下の例では、このパターンを説明します。

- 1 エディット テキスト コンポーネントは、その `UserData` プロパティにユーザーが入力する文字列を格納します。

```
function mygui_edittext1(hObject, eventdata, handles)
mystring = get(hObject,'String');
set(hObject,'UserData',mystring);
```

- 2 メニュー項目は、エディット テキスト コンポーネントの `UserData` プロパティから文字列を取得します。コールバックは、エディット テキスト ハンドルを指定するために `handles` 構造体とエディット テキストの Tag プロパティ `edittext1` を使用します。

```
function mygui_pushbutton1(hObject, eventdata, handles)
string = get(handles.edittext1,'UserData');
```

たとえば、メニュー項目が [元に戻す] である場合、そのコードは、`edittext1` の `String` をリセットして、その `UserData` に格納された値に戻すことができます。元に戻す操作を容易にするために、`UserData` は、スタックまたは循環バッファとして扱われる、文字列のセル配列になることができます。

アプリケーション データ

`UserData` のようなアプリケーション データは、ユーザー アプリケーションで定義されユーザー アプリケーションに必要なデータです。アプリケーション データ、または `UserData` を使用するかどうかは選択します。関数 `setappdata` と `getappdata` を用いて、Figure または GUI コンポーネント (ActiveX コントロール以外) にアプリケーション データを追加します。アプリケーション データと `UserData` の主な違いは以下のとおりです。

- ・ アプリケーション データには複数のデータを割り当てることができますが、`UserData` に割り当てることができる値は 1 つに限られます。
- ・ コードは、(Tag を使用する場合のように) 名前で、アプリケーション データを参照する必要がありますが、他のプロパティのように `UserData` にアクセスできます。

MATLAB の Handle Graphics オブジェクトのみが、このプロパティを使用します。次の表は、アプリケーション データへのアクセスを提供する関数をまとめています。詳細は、各関数のリファレンス ページを参照してください。

アプリケーション データを管理するための関数

関数	目的
setappdata	オブジェクト (GUI の Figure または他の Handle Graphics オブジェクト) に対して、名前の付いたアプリケーション データを指定します。1 つのオブジェクトに対して名前付きのアプリケーション データを複数指定することができます。ただし、各名前はそのオブジェクトに対して固有でなければならず、また、1 つの値のみ関連付けられます。通常は構造体と関連付けられます。
getappdata	名前付きのアプリケーション データを取得します。名前付きのアプリケーション データを取得するには、アプリケーション データに関連する名前、およびアプリケーション データが関連付けられたオブジェクトのハンドルを知る必要があります。ハンドルのみを指定すると、オブジェクトのアプリケーション データすべてが返されます。
isappdata	名前付きのアプリケーション データが存在する場合、真値を返します。そうでない場合、偽になります。
rmpappdata	名前付きのアプリケーション データを削除します。

GUIDE でのアプリケーション データ作成

関数 `setappdata` を使用して、アプリケーション データを作成します。この例は、`opening` 関数で正常な分布をした乱数からなる 35×35 行列を作成し、この行列を管理するためにアプリケーション データ `mydata` を作成します。

```
function mygui_OpeningFcn(hObject, eventdata, handles, varargin)
matrices.rand_35 = randn(35);
setappdata(hObject,'mydata',matrices);
```

このコードは `opening` 関数 (`mygui_OpeningFcn`) に記述するので、`hObject` は GUI の Figure のハンドルであり、コードは `mydata` を Figure に対するアプリケーション データとして設定します。

GUIDE でのアプリケーション データ構造体へのフィールド追加

アプリケーション データは、通常、構造体として定義されるので、ユーザーは必要に応じてフィールドを追加できます。この例では、プッシュ ボタンは、前節で作成されたアプリケーション データ構造体 `mydata` にフィールドを追加します。

1 `getappdata` を使用して構造体を取得します。

アプリケーション データ構造体の名前は、`mydata` です。これは、`Tag` が `figure1` である `Figure` に関連しています。`handles` 構造体は各コールバックに渡されるため、コードは `Figure` のハンドルを `handles.figure1` として指定します。

```
function mygui_pushbutton1(hObject, eventdata, handles)
matrices = getappdata(handles.figure1, 'mydata');
```

2 新しいフィールドを作成し、値を割り当てます。

```
matrices.randn_50 = randn(50);
```

は、`matrices` 構造体にフィールド `randn_50` を追加し、正規分布を示す乱数から構成される 50×50 行列に設定します。

3 `setappdata` を使用してデータを保存します。次のコマンドは、`setappdata` を使用して、アプリケーション データ構造体 `mydata` として `matrices` 構造体を保存します。

```
setappdata(handles.figure1, 'mydata', matrices);
```

GUI データ

GUI データは、通常、GUI の `Figure` に関連付けられます。GUIDE で作成されるすべての GUI コンポーネントのコールバックが GUI データを利用できます。GUI データを保存、または、取得するときにコンポーネントのハンドルを指定すると、MATLAB はデータをコンポーネントの親の `Figure` と自動的に関連させます。GUI データでは、

- ・ コンポーネントのハンドルを用いると、`Figure` ハンドルを見つける必要なくコールバック ルーチン内からデータにアクセスできます。
- ・ ユーザーのソース コードの中で、データに対するハードコードされた名前の作成や保持の必要はありません。

関数 `guidata` を使用して GUI データを管理します。この関数は、1 つの変数を GUI データとして保存できます。GUI データは、以下の点でアプリケーション データとは異なります。

- ・ GUI データは 1 つの変数です。ただし、構造体として定義されると、ユーザーはフィールドの追加や削除を行うことができます。
- ・ アプリケーション データは多くの変数で構成できます。各変数は、別々の 1 つしかない名前で格納されます。

- GUIDE は GUI データを用いて GUI のハンドル構造体を使用します。ユーザーは、この構造体にフィールドを追加できますが、フィールドの削除は行わないでください。
- ユーザーは、関数 `guidata` を用いて GUI データにアクセスします。この関数は GUI データの格納と取得を行います。
- GUI データを格納するために `guidata` を使用すると、既存の GUI データが上書きされます。
- 関数 `getappdata`、`setappdata`、`rmappdata` を使用することは、GUI データに影響しません。

GUIDE における GUI データ

GUIDE は、`guidata` を使用して `handles` 構造体を管理します。`handles` 構造体は、すべての GUI コンポーネントのハンドルを含みます。GUIDE は、各コールバックに対して、`handles` 構造体を入力引数として自動的に渡します。

GUIDE を用いて作成した GUIにおいて、`handles` 構造体以外の変数を管理するためには `guidata` を使用しないでください。`handles` 構造体が上書きされ、GUI が動作しなくなるためです。コールバック間でアプリケーション定義のデータを共有するために GUI データを使用するには、`handles` 構造体へ追加したフィールドにデータを保存しなければなりません。詳細は、“`handles` 構造体”(p.8-23)を参照してください。GUIDE テンプレートは、`handles` 構造体を利用して、アプリケーション定義データの保管も行います。テンプレートについての詳細は、“GUI テンプレートの選択”(p.6-5)を参照してください。

handles 構造体へのフィールドの追加

GUIDE で各コールバックへの引数として渡される `handles` 構造体にフィールドを追加するには、以下の手順を行います。

- 1 新しいフィールドに値を割り当て、構造体にフィールドを追加します。以下に例を示します。

```
handles.number_errors = 0;
```

は、`handles` 構造体にフィールド `number_errors` を追加し、そのフィールドを 0 に設定します。

- 2 データを保存するには、次のコマンドを使用します。

```
guidata(hObject, handles)
```

ここで、`hObject` はコールバックされたコンポーネントのハンドルです。GUIDE が、各コールバックに `hObject` を自動的に渡します。

GUIDE で作成された M ファイルの GUI データの変更

GUIDE が作成する M ファイルにおいて、`handles` 構造体は常に GUI データを表します。次の例は `handles` 構造体を更新してから、保存します。

`handles` 構造体が、値が 'now' であるアプリケーション定義フィールド `handles.when` を含むことを仮定します。

- 1 GUI コールバックにおいて、`handles.when` の値を 'later' に変更します。`handles` 構造体は、これによって保存されません。

```
handles.when = 'later';
```

- 2 次のコマンドを使って、変更した `handles` を保存します。

```
guidata(hObject, handles)
```

ここで、`hObject` はコールバックされたコンポーネントのハンドルです。`handles` 構造体を `guidata` を用いて保存しない場合、前の手順で構造体に行った変更は失われます。

初期化をコントロールするために GUI データを使用する

GUI は "シングルトン" として宣言できます。これは、GUI の同時に実行できるインスタンスが 1 つに限られることを意味します。“GUI は 1 つのインスタンスのみの実行を許可 (シングルトン)” (p.5-12)を参照してください。1 つの GUI のコンポーネントの `CreateFcn` は、最初に実行するときに限り呼び出されます。GUI を引き続き起動しても、すべてのオブジェクトがすでに存在しているので `CreateFcn` を実行しません。ただし、`opening` 関数は、シングルトンの GUI が起動するたびに呼び出されます。

GUI がその `OpeningFcn` において初期化を実行すると、GUI が最初に実行するときに限り、それらのいくつかまたはすべてが起こるようにしたい場合があります。この場合、GUI の実行中に、ユーザーがコマンドライン、または他の方法で GUI を再び起動させても、その内部状態が再び初期化される必要はありません。これを行う 1 つの方法として、フラグを設定してハンドル構造体に格納することができます。`opening` 関数はフラグの存在をテストできます。そして、フラグが存在しない場合に限り、初期化を行います。以下のコード部分は、このパターンを説明します。

```
function mygui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to spingui (see VARARGIN)

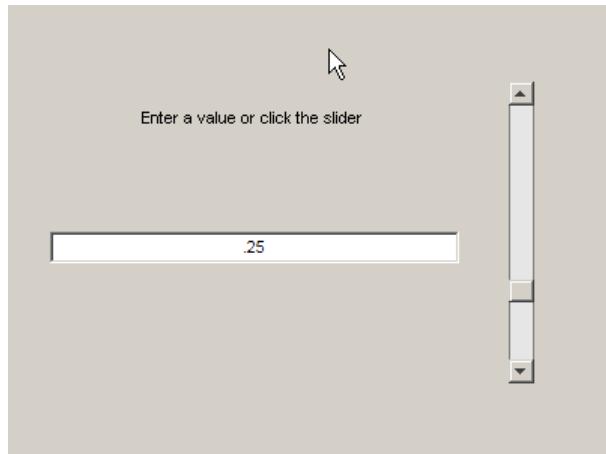
% Check whether initialization has already been performed
if ~isfield(handles,'initialized')
    % Flag not present, so create and store it
    handles.initialized = true;
    guidata(hObject,handles)
    % perform initialization; it will only happen once.
    initialize_mygui()      % Insert code or function call here
end
...
...
```

GUI のコールバック間でデータを共有する例

- ・ “はじめに” (p.9-10)
- ・ “データを UserData と共有する” (p.9-11)
- ・ “アプリケーション データとデータを共有する” (p.9-14)
- ・ “GUI データとデータを共有する” (p.9-17)

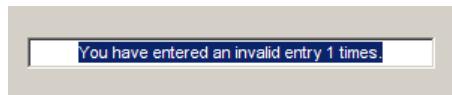
はじめに

以下の例は、スライダーとエディット テキストの GUI コンポーネント間でデータを共有する 3 つの方法の違いを説明しています。これは、以下の図に示すように、スライダーとエディット テキスト コンポーネントを含みます。ステティック テキストは、エディット テキストに値を入力するか、スライダーをクリックするようユーザーに指示します。ユーザーが無効な値を入力すると、エディット テキスト フィールドはエラー メッセージを表示します。



ユーザーがエディット テキスト コンポーネントに 0 と 1 の間の値を入力したり、
[Enter] を押したり、エディット テキストの外側をクリックしたりすると、コールバックは
handles.slider1 に新しい値を設定し、スライダーは対応する位置に移動します。

入力が無効（例えば 2.5）である場合、GUI はエラー カウンターに格納された値を
増やし、エラーのカウントを含むエディット テキスト コンポーネントにメッセージを
表示します。



データを UserData と共有する

この例の GUI ファイルのコピーを取得するには、以下の手順に従います。MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを変更しようとする場合には、最初に現在のフォルダーにその M ファイルと FIG-ファイルのコピーを保存する必要があります。これを行うには現在のフォルダーへの書き込み権限が必要です。現在のフォルダーに例のファイルをコピーし、それらを開くには、以下の手順に従ってください。

1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。

2 `guide sliderbox_userdata` と入力するか、または ここをクリックすると、GUIDE に GUI が開きます。.

3 `edit sliderbox_userdata` と入力するか、または ここをクリックすると、エディターに GUI M ファイルが開きます。

コンポーネントのプロパティ インスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を変更することができます。次に、GUIDE で [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存できます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUI を実行するか、または GUIDE で調べようとする場合、以下の手順に従います。

1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)

2 `sliderbox_userdata` の GUI を実行するには、ここをクリックします。

3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします(読み取り専用)。

4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の `examples` フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

UserData とのデータ共有の仕組み. 各 GUI コンポーネントと Figure 自身は、アプリケーション定義データを保存するために使用できる UserData プロパティをもちます。UserData にアクセスするには、コールバックはプロパティが関連しているコンポーネントのハンドルを知る必要があります。コードは、関数 `get` を使用して UserData を取得し、関数 `set` を使用して UserData を設定します。

メモ 詳細は、“UserData プロパティ”(p.9-4)を参照してください。

ここでは、エディット テキスト コンポーネントの UserData プロパティにエラー カウントを格納することで、GUI データを用いてエラー カウンターの初期化と保持を行う方法を説明します。

- 1 opening 関数に次のコードを追加して、エディット テキスト コンポーネントの UserData プロパティを初期化します。このコードでは、必要となる他のデータが可能になるように構造体内のデータを初期化します。

```
% INITIALIZE ERROR COUNT AND USE EDITTEXT1 OBJECT'S USERDATA TO STORE IT.
data.number_errors = 0;
set(handles.edittext1,'UserData',data)
```

メモ あるいは、エディット テキストに対する CreateFcn コールバックを追加し、そこでエラー カウンターを初期化します。

- 2 次のステートメントを追加して、スライダーのコールバックからエディット テキストの値を設定します。

```
set(handles.edittext1,'String',...
num2str(get(hObject,'Value')));
```

`hObject` は、スライダーのハンドルです。

- 3 エディット テキストのコールバックに以下のコード行を追加し、エディット テキストのコールバックからスライダーの値を設定します。

```
val = str2double(get(hObject,'String'));
% Determine whether val is a number between 0 and 1.
if isnumeric(val) && length(val)==1 && ...
    val >= get(handles.slider1,'Min') && ...
    val <= get(handles.slider1,'Max')
    set(handles.slider1,'Value',val);
else
    % Retrieve and increment the error count.
    % Error count is in the edit text UserData,
```

```
% so we already have its handle.  
    data = get(hObject,'UserData');  
    data.number_errors = data.number_errors+1;  
% Save the changes.  
    set(hObject,'UserData',data);  
% Display new total.  
    set(hObject,'String',...  
        [' You have entered an invalid entry ',...  
        num2str(data.number_errors), ' times.']);  
% Restore focus to the edit text box after error  
    uicontrol(hObject)  
end
```

エラー数を更新するために、コードは最初にエディット テキストの `UserData` プロパティの値を取得してから、カウントを増やす必要があります。コードは次に、`UserData` プロパティに更新されたエラー カウントを保存し、新しいカウントを表示します。

このコードは、エディット テキスト コンポーネントのコールバックに現れるので、`hObject` はエディット テキスト コンポーネントのハンドルです。コールバックの最後から 2 つめの行

```
uicontrol(hObject)
```

は有効ですが、コールバックが適切に動作するために必須ではありません。
`uicontrol` を呼び出すと、エディット テキスト ボックスをフォーカスに置く効果があります。エディット テキスト コントロールは、ユーザーが Return を押すか、あるいはコントロールをクリックした後、そのコールバックを実行します。これらの動作により、エディット テキスト ボックスはフォーカスされなくなります。エラーのイベントでエディット テキスト ボックスへのフォーカスを元に戻すと、どのような動作がエラーの原因となったかがわかります。そして、ユーザーはエディット テキスト ボックスに再び入力することでエラーを訂正できます。

アプリケーション データとデータを共有する

この例の GUI ファイルのコピーを取得するには、以下の手順に従います。MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを変更しようとする場合には、最初に現在のフォルダーにその M ファイルと FIG-ファイルのコピーを保存する必要があります。これを行うには現在のフォルダーへの書き込み権限が必要です。現在のフォルダーに例のファイルをコピーし、それらを開くには、以下の手順に従ってください。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に GUI を開くには、`guide sliderbox_appdata` と入力するか、または ここをクリックします。
- 3 エディター内に GUI M ファイルを開くには、`edit sliderbox_appdata` と入力するか、または ここをクリックします。

コンポーネントのプロパティインスペクターを開くには、レイアウトエディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を変更することができます。次に、GUIDE で [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存できます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更することもできます。

GUI を実行するか、または GUIDE で調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 `sliderbox_appdata` の GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウトエディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の `examples` フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

アプリケーション データとデータを共有する仕組み アプリケーション データを、コンポーネント、メニュー、または Figure 自身などの任意のオブジェクトと関連付けることができます。アプリケーション データにアクセスするには、コールバックはデータの名前とコンポーネントが関連付けられたハンドルを知る必要があります。関数 setappdata、getappdata、isappdata、rmappdata を使用して、アプリケーション データを管理します。

メモ 詳細は、“アプリケーション データ”(p.9-5)を参照してください。

“GUI データとデータを共有する”(p.9-17) の節では、GUI データを使用してエラー カウンターの初期化と保持を行います。この例は、アプリケーション データを使用して同じことを行う方法を示します。

- 1 opening 関数に以下のコードを追加して、opening 関数でエラー カウンターを定義します。

```
% INITIALIZE ERROR COUNT AND USE APPDATA API TO STORE IT IN FIGURE.  
slider_data.number_errors = 0;  
setappdata(hObject,'slider',slider_data);
```

このコードは最初に構造体 slider_data を作成し、それから、これを名前付きのアプリケーション データ slider に割り当てます。このコードは、opening 関数で現れるので、hObject は、アプリケーション データを Figure と関連付けます。

- 2 スライダーの Value プロパティを文字列に変換して、次のステートメントをコールバックに追加することによって、スライダーのコールバックからエディット テキスト コンポーネントの String プロパティの値を設定します。

```
set(handles.edittext1,'String',...  
num2str(get(hObject,'Value')));
```

このステートメントは、スライダーのコールバックに現れるので、hObject はスライダーのハンドルです。

- 3 エディット テキスト コンポーネントのコールバックからスライダーの値を設定します。次のコードをコールバックに追加します。ここでは、Figure の Tag プロパティが figure1 であるとします。

エラー数を更新するために、このコードは最初に名前付きのアプリケーション データ `slider` を取得してから、カウントを増やす必要があります。コードは次にアプリケーション データを保存し、新しいエラー カウントを表示します。

```

val = str2double(get(hObject,'String'));
% Determine whether val is a number between 0 and 1.
if isnumeric(val) && length(val)==1 && ...
    val >= get(handles.slider1,'Min') && ...
    val <= get(handles.slider1,'Max')
    set(handles.slider1,'Value',val);
else
    % Retrieve and increment the error count.
    slider_data = getappdata(handles.figure1,'slider');
    slider_data.number_errors = slider_data.number_errors+1;
    % Save the changes.
    setappdata(handles.figure1,'slider',slider_data);
    % Display new total.
    set(hObject,'String',...
        [' You have entered an invalid entry ',...
        num2str(slider_data.number_errors), ' times.']);
end

```

このコードは、エディット テキスト コンポーネントのコールバックに現れるので、`hObject` はエディット テキスト コンポーネントのハンドルです。コールバックの最後から 2 つめの行

```
uicontrol(hObject)
```

は有効ですが、コールバックが適切に動作するために必須ではありません。`uicontrol` を呼び出すと、エディット テキスト ボックスをフォーカスに置く効果があります。エディット テキスト コントロールは、ユーザーが Return を押すか、あるいはコントロールをクリックした後、そのコールバックを実行します。これらの動作により、エディット テキスト ボックスはフォーカスされなくなります。エラーのイベントでエディット テキスト ボックスへのフォーカスを元に戻すと、どのような動作がエラーの原因となったかがわかります。そして、ユーザーはエディット テキスト ボックスに再び入力することでエラーを訂正できます。

GUI データとデータを共有する

この例の GUI ファイルのコピーを取得するには、以下の手順に従います。MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下の

リンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを変更しようとする場合には、最初に現在のフォルダーにその M ファイルと FIG-ファイルのコピーを保存する必要があります。これを行うには現在のフォルダーへの書き込み権限が必要です。現在のフォルダーに例のファイルをコピーし、それらを開くには、以下の手順に従ってください。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に GUI を開くには、`guide sliderbox_guifile` と入力するか、または ここをクリックします。
- 3 エディターに GUI M ファイルを開くには、`edit sliderbox_guifile` と入力するか、または ここをクリックします。

コンポーネントのプロパティインスペクターを開くには、レイアウトエディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を変更することができます。次に、GUIDE で [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存できます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUI を実行するか、または GUIDE で調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 `sliderbox_guifile` の GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウトエディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

GUI データとデータ共有する機能. GUI コールバックは、GUI データにアクセスできます。1 つのコンポーネントに対するコールバックは、他のコンポーネントに対するコールバックから読むことができる GUI データに値を設定できます。この例は、GUI データを使用して、エラー カウンターの初期化と保持を行います。

メモ 詳細は、“GUI データ” (p.9-7)を参照してください。

GUI の動作は、以下のようになります。

- ・ ユーザーがスライダーを移動するとき、エディット テキスト コンポーネントは、スライダーの現在の値を表示します。
- ・ ユーザーがエディット テキスト コンポーネントに値を入力すると、スライダーは値を更新します。
- ・ スライダーに対しての範囲外の値 – つまり、0 と 1 の間の値ではない値 – をエディット テキストに入力する場合、アプリケーションはユーザーが誤った値を入力した回数を示すメッセージをエディット テキスト コンポーネントに出力します。

以下の手順のコマンドは、エラー カウンターを初期化し、スライダーとエディット テキスト コンポーネント間のやりとりを実現します。

1 opening 関数にエラー カウンターを定義します。GUI は、ユーザーがエディット テキスト コンポーネントに誤った値を入力した回数を記録し、この値を handles 構造体のフィールドに保存します。

opening 関数の中で次のように、number_errors フィールドを定義します。

```
% INITIALIZE ERROR COUNT AND USE GUIDATA TO UPDATE THE HANDLES STRUCTURE.
handles.number_errors = 0;
```

このフィールドは、以下の行よりも前に置きます。この行は、GUIDE が自動的に opening 関数に挿入します。

```
guidata(hObject,handles);
```

GUI コールバックの中でこの handles 構造体が取得できるように、guidata コマンドは、handles 構造体を保存します。

- 2 スライダーのコールバックからエディット テキストコンポーネントの String プロパティの値を設定します。スライダー コールバックの次のコマンドは、ユーザーがスライダーを動かすか、マウス ボタンをはなすときに、エディット テキストコンポーネントに表示された値を更新します。

```
set(handles.edittext1,'String',...
    num2str(get(handles.slider1,'Value')));
```

このコードは、3 つのコマンドを連結します。

- ・ コマンド get は、スライダーの現在の値を取得します。
- ・ コマンド num2str は、値を文字列に変換します。
- ・ コマンド set は、エディット テキストの String プロパティを設定し、値を更新します。

- 3 エディット テキストコンポーネントのコールバックからスライダーの値を設定します。エディット テキストコンポーネントのコールバックは、ユーザーが入力した値が 0 と 1 の間の数値であるかどうかをチェック後、スライダーの値をその値に設定します。値が範囲外の場合、エラーのカウントが増え、エディット テキストは、ユーザーが何回無効な数を入力したかを知らせるメッセージを表示します。このコードは、エディット テキストコンポーネントのコールバックに現れるので、hObject はエディット テキストコンポーネントのハンドルです。

```
val = str2double(get(hObject,'String'));
% Determine whether val is a number between 0 and 1.
if isnumeric(val) && length(val)==1 && ...
    val >= get(handles.slider1,'Min') && ...
    val <= get(handles.slider1,'Max')
    set(handles.slider1,'Value',val);
else
    % Increment the error count, and display it.
    handles.number_errors = handles.number_errors+1;
    guidata(hObject,handles); % Store the changes.
```

```
set(hObject,'String',...
[' You have entered an invalid entry ',...
 num2str(handles.number_errors), ' times.' ]);
% Restore focus to the edit text box after error
uicontrol(hObject)
end
```

このコードは、エディット テキスト コンポーネントのコールバックに現れるので、`hObject` はエディット テキスト コンポーネントのハンドルです。コールバックの最後から 2 つめの行

```
uicontrol(hObject)
```

は有効ですが、コールバックが適切に動作するために必須ではありません。`uicontrol` を呼び出すと、エディット テキスト ボックスをフォーカスに置く効果があります。エディット テキスト コントロールは、ユーザーが Return を押すか、あるいはコントロールをクリックした後、そのコールバックを実行します。これらの動作により、エディット テキスト ボックスはフォーカスされなくなります。エラーのイベントでエディット テキスト ボックスへのフォーカスを元に戻すと、どのような動作がエラーの原因となつたかがわかります。そして、ユーザーはエディット テキスト ボックスに再び入力することでエラーを訂正できます。

複数の GUI を連携して動作させる方法

このセクションの内容…

“データを共有する方法” (p.9-22)

“例 – ユーザーの入力に対するモーダル ダイアログ ボックスの取り扱い” (p.9-23)

“例 – アイコン操作ツールとして協同している個々の GUIDE GUI” (p.9-31)

データを共有する方法

GUI 内でデータを共有するために “GUI のコールバック間でデータを共有する例” (p.9-10) で述べた手法のいくつかにより、複数の GUI 間でデータを共有することもできます。最初の GUI のオブジェクトに対するハンドルが、他の GUI に対しても利用可能である限り、GUI 間でやりとりするために、GUI データ、アプリケーションデータ、UserData プロパティを使用できます。ここでは、以下の手法を説明する例を 2 つ提供します。

- ・ “例 – ユーザーの入力に対するモーダル ダイアログ ボックスの取り扱い” (p.9-23)
この例では、簡単な GUI がどのように開かれるか、およびモーダル ダイアログ ボックスからデータを取得する方法を説明します。
- ・ “例 – アイコン操作ツールとして協同している個々の GUIDE GUI” (p.9-31)
このより複雑な例では、アイコン エディターの 3 つのコンポーネントがやりとりする様子を説明します。

メモ 以下の例では、データを共有する手法を簡潔に伝えるために、コード部分を省略します。省略は、省略記号で表されます。

例の完成した M ファイルや FIG-ファイルをコピー、実行、表示、変更することができます。

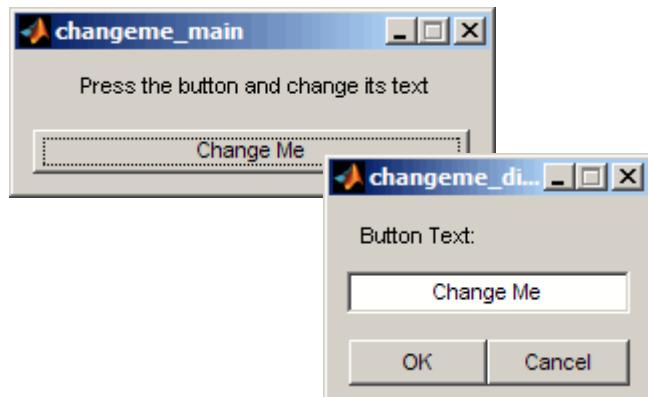
例 – ユーザーの入力に対するモーダル ダイアログ ボックスの取り扱い

- ・ “changeme の GUI の表示と実行” (p.9-24)
- ・ “Text Change ダイアログ ボックスの起動” (p.9-25)
- ・ “テキスト変更ダイアログの管理” (p.9-26)
- ・ “[Text Change] ダイアログの保護と位置決め” (p.9-27)
- ・ “[Text Change] ダイアログ ボックスのテキストの初期化” (p.9-29)
- ・ “[Text Change] ダイアログ ボックスのキャンセル” (p.9-30)
- ・ “テキスト変更の適用” (p.9-30)
- ・ “メイン GUI を閉じる” (p.9-30)

この例では、複数の GUI を共に動作させる際に呼び出される共通のタスクを行う方法を説明します。メイン GUI に対して第 2 の GUI を配置する方法を説明し、GUIDE GUI から呼び出されたモーダル ダイアログ ボックスにデータがどのように渡されるかを説明します。ダイアログ ボックスは、エディットフィールドにテキスト データを表示します。ユーザーがエディットフィールドに行った変更は、メイン GUI に戻されます。メイン GUI は、このデータを様々な方法で使用します。モーダル ダイアログ ボックスのデータを変更すると、メイン GUI のコンポーネントの 1 つの外見を更新できます。

メイン GUI、changeme_main は、1 つのボタンと説明を与えるスタティック テキスト フィールドを含みます。ボタンをクリックすると、モーダル ダイアログ ボックス changeme_dialog が開き、ボタンの現在の文字列が編集可能なテキスト フィールドに現れ、ユーザーによる変更が可能になります。

[OK] をクリックすると、テキスト フィールドの値はメイン GUI に返されます。これは、ボタンの文字列のプロパティを入力した値に設定します。メイン GUI とそのモーダル ダイアログ ボックスを、次の図に示します。



メモ `changeme_dialog` の GUI は、同じ目的をもつ組み込みのダイアログ ボックスである、MATLAB の関数 `inputdlg` にならって作成されています。これは、GUI と他のプロセスの呼び出しをロックするために `uiwait` も呼び出します。プログラミングにより GUI を作成すると、`inputdlg` を使用できます。

changeme の GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルと FIG-ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下の手順を行います。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に 2 つの GUI を開くには、コマンド `guide changeme_main; guide changeme_dialog` を入力するか、ここをクリックしてください。
- 3 エディターに GUI M ファイルを開くには、コマンド `edit changeme_main.m; edit changeme_dialog.m` を入力するか、または ここをクリックしてください。

コンポーネントのプロパティ インスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を変更することができます。次に、GUIDE から [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存することができます。これにより、GUI とその M ファイルが両方保存されます。このように GUI の 1 つを保存すると、同様に他のものも保存する必要があります。

GUI を実行するか、または GUIDE で調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 `changeme_main` の GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の `examples` フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を使用します。これにより、GUI FIG-ファイルと GUI M ファイルが両方とも保存されます。

[別名で保存] を利用するときには、いずれの GUI のファイル名も変更しないでください。`changeme_main` M コードは、関数 `changeme_dialog` を呼び出します。そのファイル名は GUI が動作しなくなるように修正します。

Text Change ダイアログ ボックスの起動

[Change Me] ボタンをクリックすると、[Text Change] ダイアログ ボックスが開きます。プロパティ/値の組を用いてそのメイン関数を呼び出すことで、このダイアログ ボックスを起動します。

- ・ 名前:’`changeme_main`’ (メイン GUI の名前)
- ・ 値:メイン GUI の Figure ハンドル

```
function buttonChangeMe_Callback(hObject, eventdata, handles)

% Call the dialog to change button name giving this figure's handle
changeme_dialog('changeme_main', handles.figure);
```

ダイアログ ボックスは、メイン GUI のデータにアクセスするためにハンドルを使用します。メイン GUI のデータがない場合、ダイアログ ボックスはコマンド ウィンドウに適切な使用方法を示すエラーを表示してから終了します。

テキスト変更ダイアログの管理

1 [Text Change] ダイアログ ボックスの Figure に対するプロパティ インスペクターにおいて、*WindowStyle* プロパティを 'Modal' に設定します。これにより、ダイアログ ボックスがアクティブであるとき、ユーザーは他の Figure とやりとりすることができません。

2 *uiresume* が呼び出されるまで、*output* 関数の呼び出しを延期するために、ダイアログ ボックスの *OpeningFcn* で *uiwait* を呼び出します。これによって、そのときまで GUI の起動の呼び出しが返らないようにすることができます。

```
function changeme_dialog_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
.
```

```
.
```

```
.
```

3 Figure に対する *CloseRequestFcn* 内の *uiresume*、[Cancel] ボタン、および [OK] ボタンを呼び出します。GUI を閉じるコールバックは、それぞれ *uiresume* を呼び出す必要があります。

```
function buttonCancel_Callback(hObject, eventdata, handles)
```

```
uiresume(handles.figure);
```

```
function figure_CloseRequestFcn(hObject, eventdata, handles)
```

```

uiresume(hObject);

function buttonOK_Callback(hObject, eventdata, handles)
.

.

uiresume(handles.figure);

```

[Text Change] ダイアログの保護と位置決め

- 1 ユーザーは、メイン GUI の buttonChangeMe_Callback コールバックをトリガーすることで、[Text Change] ダイアログ ボックスを開きます。これは、メイン GUI の Figure ハンドルを changeme_main と呼ばれるプロパティとして与えます。
- 2 ダイアログ ボックスに対する OpeningFcn は、varargin セル配列を検索し、インデックスを付けることで入力を有効にします。'changeme_main' とハンドルが連続する引数として見つけられると、それは uifigure を呼び出します。これにより、ダイアログ GUI は、OutputFcn が Figure を閉じるのを待たずに終了できます。これがプロパティを見つからない、または無効な値を見つけると、モーダル ダイアログ ボックスがエラーを表示して終了します。

```

function changeme_dialog_OpeningFcn(hObject, ...
    eventdata, handles, varargin)

% Is the changeme_main gui's handle is passed in varargin?
% if the name 'changeme_main' is found, and the next argument
% varargin{mainGuilnput+1} is a handle, assume we can open it.

dontOpen = false;
mainGuilnput = find(strcmp(varargin, 'changeme_main'));
if (isempty(mainGuilnput))
    || (length(varargin) <= mainGuilnput)
    || (~ishandle(varargin{mainGuilnput+1}))
    dontOpen = true;
else
.

.

end

```

```
        .
        .
if dontOpen
    disp(' _____, );
    disp(' Improper input arguments. Pass a property value pair' )
    disp(' whose name is "changeme_main" and value is the handle' )
    disp(' to the changeme_main figure.' );
    disp(' eg:' );
    disp('     x = changeme_main() );
    disp('     changeme_dialog('changeme_main' , x)' );
    disp(' _____, );
else
    uiwait(hObject);
end
```

- 3 その Figure に渡されたハンドルを利用して、changeme_dialog_OpeningFcn はメイン GUI の中央に [Text Change] ダイアログ ボックスを置きます。そのため、メイン Figure が移動されてダイアログ ボックスが起動すると、これは常に固定した位置ではなく、相対的に同じ位置に開きます。

```
function changeme_dialog_OpeningFcn(hObject, ...
    eventdata, handles, varargin)

        .
        .
mainGuiInput = find(strcmp(varargin, 'changeme_main'));

        .
        .

handles.changeMeMain = varargin{mainGuiInput+1};

        .
        .

% Position to be relative to parent:
parentPosition = getpixelposition(handles.changeMeMain);
currentPosition = get(hObject, 'Position');
% Sets the position to be directly centered on the main figure
newX = parentPosition(1) + (parentPosition(3)/2 ...
    - currentPosition(3)/2);
```

```

newY = parentPosition(2) + (parentPosition(4)/2 ...
    - currentPosition(4)/2);
newW = currentPosition(3);
newH = currentPosition(4);

set(hObject, 'Position', [newX, newY, newW, newH]);
.
.
.
.
```

[Text Change] ダイアログ ボックスのテキストの初期化

- 1 [Text Change] ダイアログ ボックスのテキストを [Change Me] ボタンの現在のテキストに初期化するには、メインの GUI の handles 構造体を、モーダル ダイアログ ボックスに渡したそのハンドルから取得します。

```

function changeme_dialog_OpeningFcn(hObject, ...
    eventdata, handles, varargin)

mainGuilnput = find(strcmp(varargin, 'changeme_main'));
.
.
.

handles.changeMeMain = varargin{mainGuilnput+1};
```

- 2 [Change Me] ボタンの String プロパティを取得し、エディット ボックスの String プロパティを、ダイアログ ボックス OpeningFcn の値に設定します。

```

% Obtain handles using GUIDATA with the caller's handle
mainHandles = guidata(handles.changeMeMain);
% Set the edit text to the String of the main GUI's button
set(handles.editChangeMe, 'String', ...
    get(mainHandles.buttonChangeMe, 'String'));
```

[Text Change] ダイアログ ボックスのキャンセル

[キャンセル] をクリックするか、あるいはウインドウを閉じると、`uiresume` を呼び出してモーダル ダイアログ ボックスを閉じます。モーダル ダイアログ ボックスを閉じるためには、メイン GUI を修正しないでください。

```
function buttonCancel_Callback(hObject, ...
    eventdata, handles)
uiresume(handles.figure);

function figure_CloseRequestFcn(hObject, ...
    eventdata, handles)
uiresume(hObject);
```

テキスト変更の適用

テキストに変更を適用するには、モーダル ダイアログ ボックスの `OpeningFcn` に保存された `handles` 構造体のメイン GUI への参照を使用します。`[OK]` をクリックすると、テキストの変更が適用されます。これにより、メイン GUI の `[Change Me]` ボタン ラベルが、モーダル ダイアログ ボックスのテキスト フィールドに入力された値に設定されます。

```
function buttonOK_Callback(hObject, ...
    eventdata, handles)
text = get(handles.editChangeMe, 'String');
main = handles.changeMeMain;
mainHandles = guidata(main);
changeMeButton = mainHandles.buttonChangeMe;
set(changeMeButton, 'String', text);
uiresume(handles.figure);
```

メイン GUI を閉じる

ユーザーが `changeme_dialog` GUI を閉じると、`changeme_main` GUI は待ち状態になります。ユーザーは、名前を再び変更するためにプッシュ ボタンをクリックするか、または `[X]` クローズ ボックスをクリックして GUI を閉じることができます。ユーザーが GUI を閉じると、GUI の `Figure` を削除する前にその `OutputFcn` はプッシュ ボタンの現在のラベル（その `String` プロパティ）を返します。

```
function varargout = changeme_dialog_Dialog_OutputFcn...
    (hObject, eventdata, handles)
% Get pushbutton string from handles structure and output it
varargout{1} = get(handles.buttonChangeMe, 'String');
```

```
% Now destroy yourself  
delete(hObject);
```

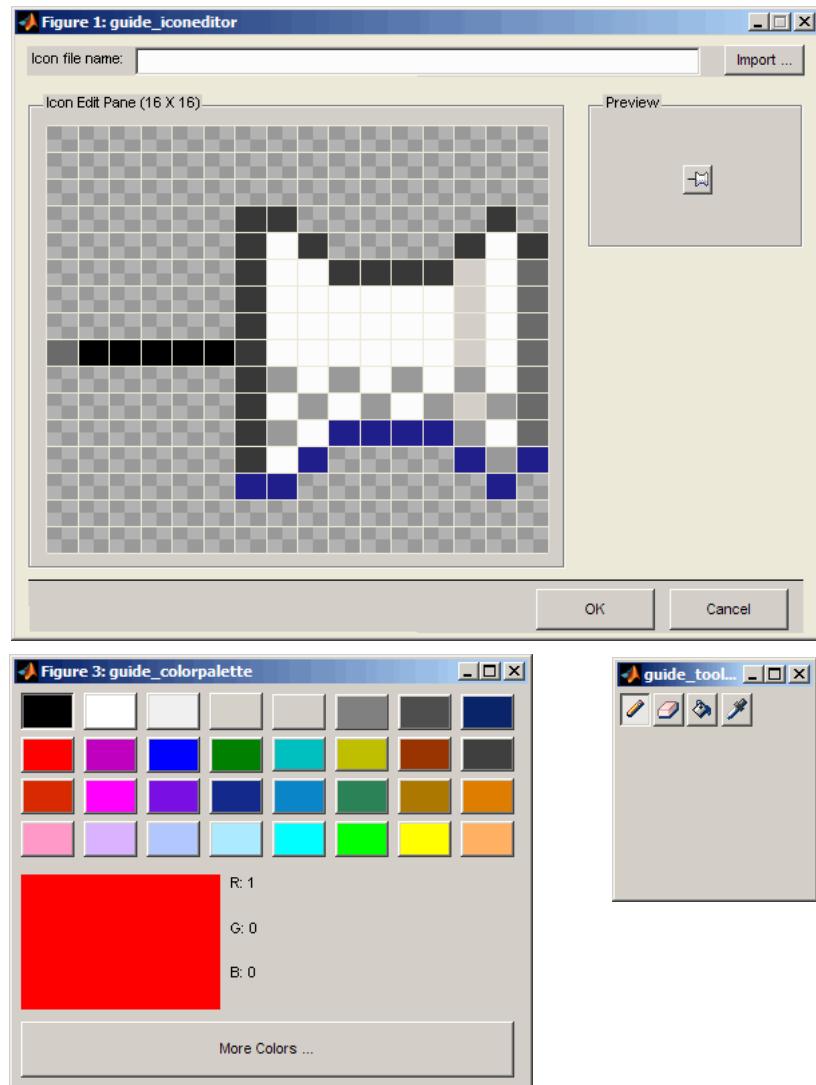
CloseRequestFcn も必要になります。これを指定しないと、既定値の CloseRequestFcn、MATLAB 関数 closreq は、任意の OutputFcn が呼び出しできる前に Figure を直ちに削除するので、GUI はデータを出力できません。この figure_CloseRequestFcn はそれを行いますが、GUI が待ち状態にない場合に限ります。GUI が待ち状態にあるときは、GUI は uiresume を呼び出し、返ります。OutputFcn を呼び出し可能にします。

```
function figure_CloseRequestFcn(hObject, eventdata, handles)  
. . .  
if isequal(get(hObject, 'waitstatus'), 'waiting')  
    % The GUI is still in UIWAIT, use UIRESUME and return  
    uiresume(hObject);  
else  
    % The GUI is no longer waiting, so destroy it now.  
    delete(hObject);  
end
```

例 – アイコン操作ツールとして協同している個々の GUIDE GUI

この例は、メイン GUI から起動されたときに、GUIDE の 3 つの GUI がどのようにともに動作するかを示しています。ツールが下記にリストされて説明されています。

- ・ 描画エリア（アイコン エディター）
- ・ ツール選択ツールバー（ツール パレット）
- ・ 色の選択（カラー パレット）



これらの GUI はデータを共有し、いくつかの異なる手法を利用してそれぞれに対して機能します。

3つのアイコンを操作する GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルと FIG-ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下の手順を行います。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に 2 つの GUI を開くには、コマンド `guide guide_iconeditor; guide guide_toolpalette; guide guide_colorpalette` を入力するか、または ここをクリックしてください。
- 3 エディターに GUI M ファイルを開くには、コマンド `edit guide_iconeditor.m; edit guide_toolpalette.m; edit guide_colorpalette.m` を入力するか、ここをクリックしてください。

コンポーネントのプロパティインスペクターを開くには、レイアウトエディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を変更することができます。次に、GUIDE で [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存できます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUI を実行するか、または GUIDE で調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 `guide_iconeditor` GUI を実行するにはここをクリックします。
- 3 GUIDE レイアウトエディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

次の順序で、アイコン エディター アプリケーションの動作を説明します。

- ・ “M ファイルの実現” (p.9-34)
- ・ “アイコン エディター、ツール パレット、カラー パレットを開く” (p.9-36)
- ・ “カラー パレットに初期の色を設定する” (p.9-38)
- ・ “アイコン エディターからカラー パレットの現在の色にアクセスする” (p.9-39)
- ・ “データを共有するために UserData プロパティを利用する” (p.9-40)
- ・ “現在のツールのカーソルの表示” (p.9-41)
- ・ “終了するときにすべてのウィンドウを閉じる” (p.9-42)

M ファイルの実現

アイコン エディター アプリケーションは、GUIDE で実行される 3 つの M ファイルと FIG-ファイルを使用します。GUIDE 環境でそれらを変更したり、改善するように、ユーザーは選択できます。ファイルは、以下のとおりです。

- ・ guide_iconeditor.fig と guide_iconeditor.m – アイコン ファイルを描いたり、変更するためのメイン GUI
- ・ guide_colorpalette.fig と guide_colorpalette.m – 現在の色を選択するパレット
- ・ guide_toolpalette.fig と guide_toolpalette.m – 4 つの編集ツールを選択するためのパレット

M ファイルは以下の関数シグネチャと出力（出力がある場合）を含みます。

- ・ guide_iconeditor.m

```
varargout = guide_iconeditor(varargin)
guide_iconeditor_OpeningFcn(hObject, eventdata, handles, varargin)
varargout = guide_iconeditor_OutputFcn(hObject, eventdata, handles)
```

```
editFilename_CreateFcn(hObject, eventdata, handles)
buttonImport_Callback(hObject, eventdata, handles)
buttonOK_Callback(hObject, eventdata, handles)
buttonCancel_Callback(hObject, eventdata, handles)
editFilename_ButtonDownFcn(hObject, eventdata, handles)
editFilename_Callback(hObject, eventdata, handles)
figure_CloseRequestFcn(hObject, eventdata, handles)
figure_WindowButtonDownFcn(hObject, eventdata, handles)
figure_WindowButtonUpFcn(hObject, eventdata, handles)
figure_WindowButtonMotionFcn(hObject, eventdata, handles)
[toolPalette, toolPaletteHandles] = getToolPalette(handles)
[colorPalette, colorPaletteHandles] = getColorPalette(handles)
setColor(hObject, color)
color = getColor(hObject)
updateCursor(hObject, overIcon)
applyCurrentTool(handles)
localUpdateIconPlot(handles)
cdwithinan = localGetIconCDataWithNaNs(handles)
```

- `guide_colorpalette.m`

```
varargout = guide_colorpalette(varargin)
guide_colorpalette_OpeningFcn(hObject, eventdata, handles, varargin)
varargout = guide_colorpalette_OutputFcn(hObject, eventdata, handles)
buttonMoreColors_Callback(hObject, eventdata, handles)
colorCellCallback(hObject, eventdata, handles)
figure_CloseRequestFcn(hObject, eventdata, handles)
localUpdateColor(handles)
setSelectedColor(hObject, color)
```

- `guide_toolPalatte.m`

```
varargout = guide_toolpalette(varargin)
guide_toolpalette_OpeningFcn(hObject, eventdata, handles, varargin)
varargout = guide_toolpalette_OutputFcn(hObject, eventdata, handles)
toolPencil_CreateFcn(hObject, eventdata, handles)
toolEraser_CreateFcn(hObject, eventdata, handles)
toolBucket_CreateFcn(hObject, eventdata, handles)
toolPicker_CreateFcn(hObject, eventdata, handles)
toolPalette_SelectionChangeFcn(hObject, eventdata, handles)
```

```
figure_CloseRequestFcn(hObject, eventdata, handles)
[iconEditor, iconEditorHandles] = getIconEditor(handles)
cdata = pencilToolCallback(handles, toolstruct, cdata, point)
cdata = eraserToolCallback(handles, toolstruct, cdata, point)
cdata = bucketToolCallback(handles, toolstruct, cdata, point)
cdata = fillColor(cdata, rows, cols, color, row, col, seedcolor)
cdata = colorpickerToolCallback(handles, toolstruct, cdata, point)
```

アイコン エディター、ツール パレット、カラー パレットを開く

アイコン エディターを開くと、ツール パレットとカラー パレットが自動的に起動します。パレットはアイコン エディターの子であり、以下に述べるようなやりとりを行います。

- ・ プロパティ/値の組 – 入力引数として渡すことで、新規に起動または既存の GUI にデータを送ります。
- ・ GUI データ – GUI の handles 構造体にデータを格納します。1 つの GUI 内、または複数の GUI 間でデータをやりとりできます。
- ・ 出力 – 起動された GUI からデータを出力します。起動された GUI の handles 構造体など、起動している GUI にデータを返し、データをやりとりするために使用されます。

アイコン エディターは、プロパティ/値の組としてツール パレットとカラー パレットに渡されます。これにより、ツールとカラー パレットはアイコン エディターにコールバックできます。両方のパレットを呼び出すと、それらの GUI の Figure のハンドルが出力値になります。これらの Figure ハンドルは、アイコン エディターの handles 構造体に保存されています。

```
% in Icon Editor
function guide_Icon_Editor_OpeningFcn(hObject, eventdata, handles, varargin)
```

```

.
.
.

handles.colorPalette = guide_colorpalette('iconEditor', hObject);
handles.toolPalette = guide_toolpalette('iconEditor', hObject);
```

```
% Update handles structure  
guidata(hObject, handles);
```

カラー パレットは、後で使用するためにアイコン エディターを記憶する必要があります。

```
% in colorPalette  
function guide_colorpalette_OpeningFcn(hObject, eventdata, handles, varargin)  
handles.output = hObject;  
  
. . .  
  
handles.iconEditor = [];  
  
iconEditorInput = find(strcmp(varargin, 'iconEditor'));  
if ~isempty(iconEditorInput)  
    handles.iconEditor = varargin{iconEditorInput+1};  
end  
  
. . .  
  
% Update handles structure  
guidata(hObject, handles);
```

ツール パレットも、アイコン エディターを記憶する必要があります。

```
% in toolPalette  
function guide_toolpalette_OpeningFcn(hObject, ...  
    eventdata, handles, varargin)  
handles.output = hObject;  
  
. . .  
  
handles.iconEditor = [];  
  
iconEditorInput = find(strcmp(varargin, 'iconEditor'));  
if ~isempty(iconEditorInput)  
    handles.iconEditor = varargin{iconEditorInput+1};  
end  
  
. . .
```

```
% Update handles structure  
guidata(hObject, handles);
```

カラー パレットに初期の色を設定する

3 つの GUI すべてを作成後、初期の色を設定する必要があります。アイコン エディターからカラー パレットを呼び出すと、カラー パレットは、アイコン エディターに初期の色の設定方法を伝える関数ハンドルを渡します。この関数ハンドルは、その handles 構造体に格納されます。カラー パレットがハンドルを出力する対象となる Figure から handles 構造体を取得できます。

```
% in colorPalette  
function guide_colorpalette_OpeningFcn(hObject, ...  
    eventdata, handles, varargin)  
handles.output = hObject;  
  
.  
  
% Set the initial palette color to black  
handles.mSelectedColor = [0 0 0];  
  
% Publish the function setSelectedColor  
handles.setColor = @setSelectedColor;  
  
.  
  
% Update handles structure  
guidata(hObject, handles);  
  
% in colorPalette  
function setSelectedColor(hObject, color)  
handles = guidata(hObject);  
  
.  
  
handles.mSelectedColor = color;  
  
.
```

```
guidata(hObject, handles);
```

初期の色を' red' に設定し、アイコン エディターから public 関数を呼び出します。

```
% in Icon Editor
function guide_iconeditor_OpeningFcn(hObject, ...
    eventdata, handles, varargin)
.

.

handles.colorPalette = guide_colorpalette(' iconEditor', hObject);
.

.

colorPalette = handles.colorPalette;
colorPaletteHandles = guidata(colorPalette);
colorPaletteHandles.setColor(colorPalette, [1 0 0]);
.

.

% Update handles structure
guidata(hObject, handles);
```

アイコン エディターからカラー パレットの現在の色にアクセスする
カラー パレットは、現在のカラー データを初期化します。

```
%in colorPalette
function guide_colorpalette_OpeningFcn(hObject, ...
    eventdata, handles, varargin)
handles.output = hObject;
.

.

handles.mSelectedColor = [0 0 0];
.

.

% Update handles structure
guidata(hObject, handles);
```

アイコン エディターは、カラー パレットの GUI データから handles 構造体により初期の色を取得します。

```
% in Icon Editor
function color = getColor(hObject)
handles = guidata(hObject);
colorPalette = handles.colorPalette;
colorPaletteHandles = guidata(colorPalette);
color = colorPaletteHandles.mSelectedColor;
```

データを共有するために UserData プロパティを利用する

GUIDE GUI のコンポーネントの UserData プロパティを使用して、データを共有できます。アイコンの編集エリアでマウスをクリックして、ツールを選択します。ユーザーが編集しているアイコンを変更するために、ツールの CData を変更することで、ツール パレット内の各ツールを使用できます。GUI は、各ツールの UserData プロパティを使用して、ツールが選択され、アイコン編集エリアに適用されるときにユーザーが呼び出す関数を記録します。各ツールは、アイコン データの別の外見を変えます。鉛筆ツールの動作仕様の例を示します。

鉛筆ツールの CreateFcn において、鉛筆ツールの関数をポイントするユーザー データを追加します。

```
% in toolPalette
function toolPencil_CreateFcn(hObject, eventdata, handles)
set(hObject,'UserData', struct('Callback', @pencilToolCallback));
```

ツール パレットは、その handles 構造体の mCurrentTool フィールドで現在選択したツールをトラッキングします。ツール パレットの handles 構造体を作成後、他の GUI からこの構造体を取得できます。ツール パレットのボタンをクリックしてから guidata を呼び出すことによって、現在選択したツールを設定します。

```
% in toolPalette
function toolPalette_SelectionChangeFcn(hObject, ...
    eventdata, handles)
handles.mCurrentTool = hObject;
guidata(hObject, handles);
```

鉛筆ツールを選択してアイコン編集エリアをクリックすると、アイコン エディターが鉛筆ツール関数を呼び出します。

```
% in iconEditor
function iconEditor_WindowButtonDownFcn(hObject, ...
    eventdata, handles)
toolPalette = handles.toolPalette;
toolPaletteHandles = guidata(toolPalette);

.

.

userData = get(toolPaletteHandles.mCurrentTool, 'UserData');
handles.mlIconCData = userData.Callback(toolPaletteHandles, ...
    toolPaletteHandles.mCurrentTool, handles.mlIconCData, ...);
```

次のコードは、マウスでクリックされたアイコン編集エリアのピクセルの値(ツール アイコンの CData)が、現在カラー パレットで選択されている色に変わることを示します。

```
% in toolPalette
function cdata = pencilToolCallback(handles, toolstruct, cdata, ...)
iconEditor = handles.iconEditor;
iconEditorHandles = guidata(iconEditor);
x = ...
y = ...
% update color of the selected block
color = iconEditorHandles.getColor(iconEditor);
cdata(y, x, :) = color;
```

現在のツールのカーソルの表示

マウスが編集エリアにあり、既定の矢印が編集エリアの外側に表示する場合、カーソルに現在のツールのポインター アイコンを表示させることができます。これを行うには、ツール パレットの handles 構造体により、選択されたツールを識別する必要があります。

```
% in Icon Editor
function iconEditor_WindowButtonMotionFcn(hObject, ...
    eventdata, handles)
.

.

rows = size(handles.mlIconCData, 1);
cols = size(handles.mlIconCData, 2);
pt = get(handles.icon, 'currentpoint');
```

```
overicon = (pt(1,1)>=0 && pt(1,1)<=rows) && ...
            (pt(1,2)>=0 && pt(1,2)<=cols);

.
.

if ~overicon
    set(hObject,'pointer','arrow');
else
    toolPalette = handles.toolPalette;
    toolPaletteHandles = guidata(toolPalette);
    tool = toolPaletteHandles.mCurrentTool;
    cdata = round(mean(get(tool,'cdata'),3))+1;
    if ~isempty(cdata)
        set(hObject,'pointer','custom','PointerShapeCData',...
              cdata(1:16,1:16),'PointerShapeHotSpot',[16 1]);
    end
end
.
.
```

終了するときにすべてのウィンドウを閉じる

アイコン エディターが開くと、handles 構造体に、ハンドルや他のデータを保存して、アイコン エディターはカラー パレットとツール パレットを開きます。アイコン エディターの OpeningFcn は、最後に uiwait を呼び出し、GUI が完成するまで出力を引き延ばします。ウィンドウを閉じる必要があると、一連の複雑なクローズ過程が含まれているので、カラー パレットもツール パレットもアイコン エディターを独立に閉じます。次のメソッドのいずれかを用いて、すべてのウィンドウを閉じることができます。

- ツール パレットとカラー パレットの [OK] ボタンまたは [キャンセル] ボタンをクリックし、アイコン エディター ウィンドウを閉じる。
- アイコン エディター ウィンドウを直接閉じる。

閉じるボタン ([X]) を直接クリックすることで、カラー パレットとツール パレットの ウィンドウを閉じることはできません。

次の例では、アイコン エディターの出力を、アイコンの CData であると設定します。 uiwait をもつアイコン エディターの opening 関数は、次のコードを含みます。

```
% in Icon Editor
function guide_iconeditor_OpeningFcn(hObject, eventdata, ...
    handles, varargin)
.
.
.

handles.colorPalette = guide_colorpalette();
handles.toolPalette = guide_toolpalette('iconEditor', hObject);
.

.

% Update handles structure
guidata(hObject, handles);
uiwait(hObject);
```

結果として、各終了パスで uiresume を呼び出す必要があります。

```
% in Icon Editor
function buttonOK_Callback(hObject, eventdata, handles)
uiresume(handles.figure);

function buttonCancel_Callback(hObject, eventdata, handles)
% Make sure the return data will be empty if we cancelled
handles.mlIconCData = [];
guidata(handles.figure, handles);
uiresume(handles.figure);

function Icon Editor_CloseRequestFcn(hObject, eventdata, handles)
uiresume(hObject);
```

他の方法ではカラー パレットが閉じないようにするには、その closerequestfcn をオーバーライドして動作しないようにします。

```
% in colorPalette
function figure_CloseRequestFcn(hObject, eventdata, handles)
% Don't close this figure. It must be deleted from Icon Editor
```

ツール パレットに対して同じことを行います。

```
% in toolPalette
```

```
function figure_CloseRequestFcn(hObject, eventdata, handles)
% Don't close this figure. It must be deleted from Icon Editor
```

最後に、output 関数において 3 つの GUI をすべて削除します。

```
% in Icon Editor
function varargout = guide_iconeditor_OutputFcn(hObject, ...
    eventdata, handles)
% Return the cdata of the icon. If cancelled, this will be empty
varargout{1} = handles.mlIconCData;
delete(handles.toolbar);
delete(handles.colorPalette);
delete(hObject);
```

GUIDE GUI の例

- “複数の座標軸をもつ GUI” (p.10-2)
- “3-D 表示のアニメーションのための GUI” (p.10-15)
- “テーブルのデータを対話的に調べる GUI” (p.10-32)
- “リスト ボックス ディレクトリ リーダー” (p.10-54)
- “リスト ボックスからワークスペース変数にアクセス” (p.10-61)
- “Simulink モデル パラメーターを設定する GUI” (p.10-66)
- “アドレス ブック リーダー” (p.10-81)
- “操作確認のためのモーダル ダイアログの使用” (p.10-98)

複数の座標軸をもつ GUI

このセクションの内容…

- “複数の Axes の例について” (p.10-2)
- “複数の座標軸をもつ GUI の表示と実行” (p.10-3)
- “GUI の設計” (p.10-5)
- “Plot プッシュ ボタンのコールバック” (p.10-8)
- “ユーザー入力を数として有効にする” (p.10-11)

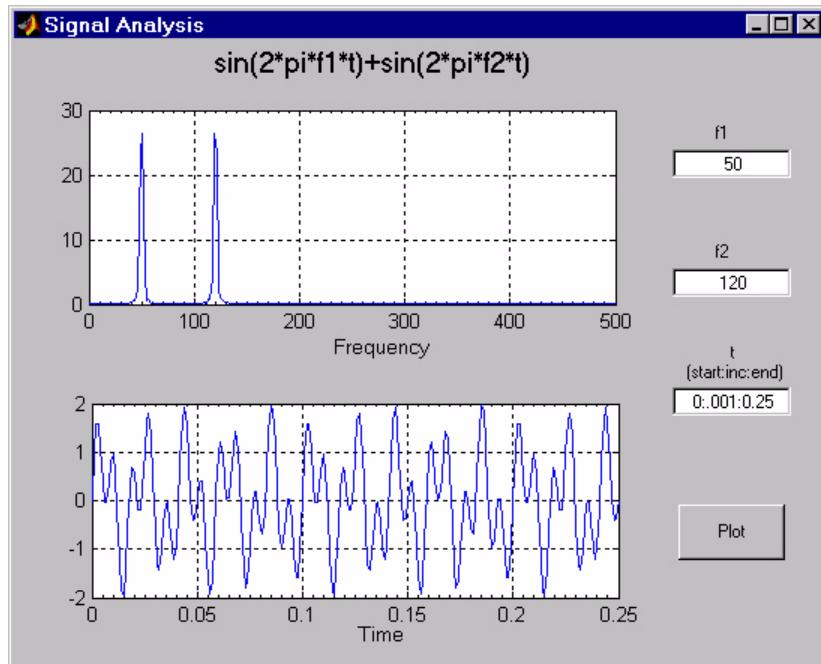
複数の Axes の例について

この例は、ユーザーが入力した 3 つのパラメーターから導出されるデータをプロットする GUI を作成します。パラメーターは、時間や周波数により変化する信号を定義します。GUI の 2 つの座標軸のうちの 1 つは、時間領域のデータを表示し、他方は周波数領域のデータを表示します。

この例で説明されている GUI 構築の手法として、以下があります。

- ・ どの Axes オブジェクトが、プロット コマンドの対象であるかを決める
- ・ エディット テキスト コントロールを使用して、数値の入力と MATLAB の式を読み込む
- ・ 入力を文字列から数に変換したり、結果を確認する
- ・ ユーザー入力の検証が失敗した場合、エディット テキスト ボックスのフォーカスを戻す

はじめて信号解析の GUI を開くときには、以下の図のような GUI が開きます。GUI では、ユーザーが入力するパラメーター f_1 , f_2 , t を用いて、Figure の最上部に示されている式が評価されます。上側の線グラフは、下側の線グラフに表示されている計算された信号のフーリエ変換を表示します。



メモ 以下の GUIDE の例 “テーブルのデータを対話的に調べる GUI” (p.10-32) では、時間プロットや周波数プロットも表示するさらに複雑な GUI を作成することができます。

複数の座標軸をもつ GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを変更しようとする場合には、最初に現在のフォルダーにその M ファイルと FIG-ファイルのコピーを保存する必要があります。これを行うには現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下のリンクをクリックします。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に GUI を開くには、guide two_axes と入力するか、または ここをクリックします。
- 3 edit two_axes と入力するか、または ここをクリックすると、エディターに GUI M ファイルが開きます。

コンポーネントのプロパティ インスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を変更することができます。次に、GUIDE で [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存できます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

メモ GUIDE から GUIDE GUI の名前の変更のみを行うフォルダー ウィンドウまたはコマンド ラインから GUIDE のファイル名を変更すると、元の名前を格納するまで正常な動作ができなくなります。

GUI を実行するか、または GUIDE で調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 two_axes の GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

GUI の設計

この GUI は 3 つの入力値を表現する 2 つのグラフをプロットします。

- ・ 周波数 1(f_1)
- ・ 周波数 2 (f_2)
- ・ 時間ベクトル (t)

ユーザーが [Plot] ボタンをクリックすると、GUI はこれらの値を 2 つの正弦関数の和である MATLAB 式に入力します。

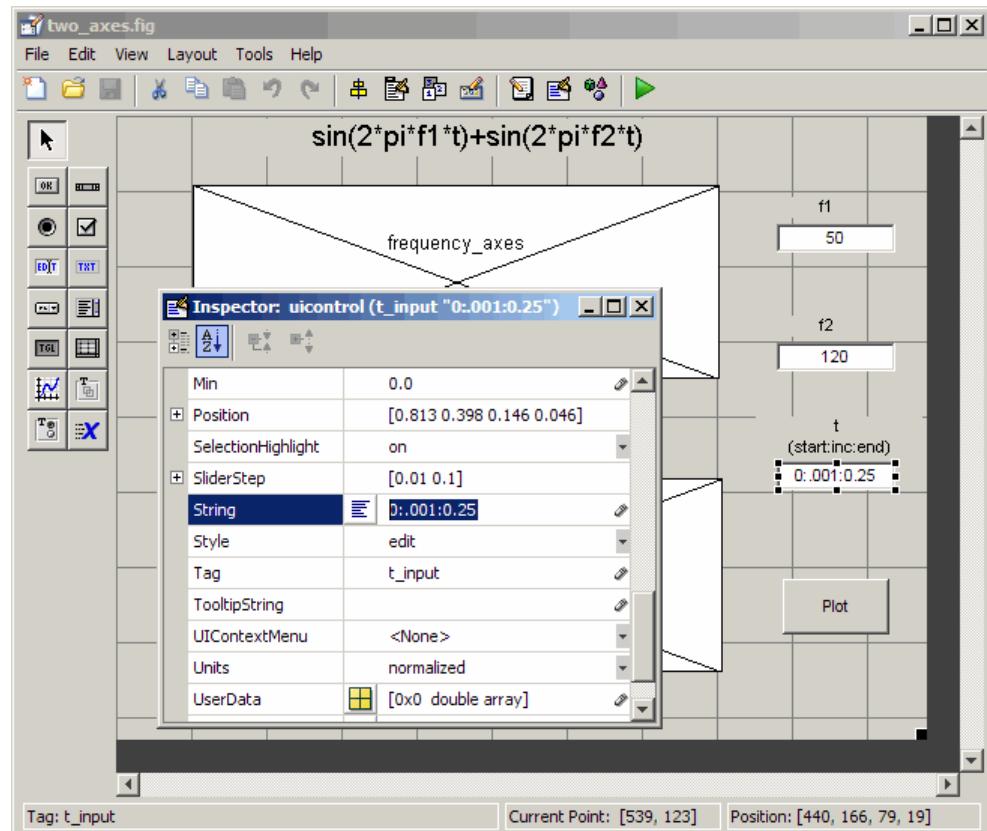
$$x = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$$

すると、GUI は x の FFT (高速フーリエ変換) を計算し、周波数領域と時間領域のデータを別々の座標軸にプロットします。

入力に対する既定値の指定

この GUI は 3 つの入力値に対して既定値を与えます。これによって、[Plot] ボタンをクリックでき、GUI が実行すると直ちに結果を見ることができます。既定値は、ユーザーが入力する典型的な値も示します。

既定値を作成するには、エディットテキストの String プロパティを設定します。次の図は、時間ベクトルに対する設定値を示します。

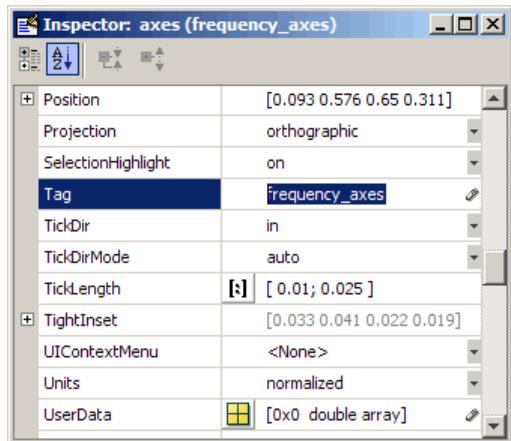


座標軸の同定

この GUI には 2 つの座標軸があるので、データをプロットする際には、どちらの座標軸にプロットするかを指定する必要があります。座標軸の識別には、handles構造体を使用してください。handles構造体は、GUI におけるすべてのコンポーネントのハンドルを含みます。handles 構造体は、GUIDE がすべてのコンポーネントのコールバックに対して入力引数として渡す変数です。

```
component_callback(hObject, eventdata, handles)
```

この構造体は、すべての GUI コンポーネントのハンドルを含みます。ユーザーは、GUIDE が各コンポーネントの Tag プロパティから導出するフィールド名を用いて、ハンドルにアクセスします。コードをさらに可読にし（記憶しやすくするために）、この例では、Tag プロパティを説明的な名前に設定します。次の図は、プロパティインスペクターで、上側の座標軸の Tag を 'frequency_axes' に設定した様子を示しています。



Tag を変更すると、GUIDE は周波数プロットの座標軸のフィールド名を、handles 構造体の frequency_axes に設定できます。plot_button_Callback 内では、handles.frequency_axes でその Axes のハンドルにアクセスします。グラフが正しい座標軸に表示されるように、以下のように plot の 1 つめの引数としてハンドルを使用します。

```
plot(handles.frequency_axes, f, m(1:257))
```

同様に、時間軸の Tag は、time_axes に設定します。これは、以下のように plot への呼び出しで使用されます。

```
plot(handles.time_axes, t, x)
```

詳細は、“handles 構造体”(p.8-23)を参照してください。対象とする座標軸を指定するハンドルを使用する方法の詳細については、“Plot プッシュ ボタンのコールバック”(p.10-8)を参照してください。

GUI オプションの設定

GUIDE では、[ツール] メニューから利用できる、GUI オプション で一連の設定をします。この GUI に対して 2 つの GUI オプション設定が特に重要です。

- ・ サイズ変更動作:比例 (プロポーショナル)
- ・ コマンド ラインへのアクセス:コールバック

プロポーショナル サイズ変更動作. サイズ変更動作として、[比例 (プロポーショナル)] を選択すると、プロットがさらによく見えるように GUI のサイズを変更することができます。このオプションの設定を利用すると、GUI のサイズを変更する際にテキスト以外は比例的に拡大または縮小されます。

オブジェクト ハンドルへのコールバックのアクセス可能性. GUI が座標軸を含む場合、ハンドルは、他のオブジェクトのコールバック内から参照できなければなりません。このため、コマンド ラインからの入力であるかのように、プロットのコマンドを使用することができます。コマンド ラインのアクセス可能性として、[コールバック] が既定値です。

詳細は、“GUI オプション” (p.5-9)を参照してください。

Plot プッシュ ボタンのコールバック

この GUI は [Plot] ボタンのコールバックのみを使用します。コンポーネントの入力を有効にしようとする場合を除き、エディット テキスト コンポーネントのコールバックをコーディングする必要はありません。ユーザーが [Plot] ボタンをクリックすると、コールバックは、3 つの基本的なタスクを実行します。コールバックは、エディット テキスト コンポーネントからユーザーの入力を得て、データを計算し、2 つのプロットを生成します。

ユーザー入力の取得

ユーザーは、2 つの周波数と時間のベクトルに対する値を、3 つのエディット テキスト ボックスに入力します。コールバックの最初のタスクは、これらの値を読むことです。これは、以下のことを含みます。

- ・ `handles` 構造体を使用して、エディット テキスト ハンドルにアクセスし、3 つのエディット テキスト ボックスの現在の値を読みます。
- ・ 2 つの周波数の値 (`f1` および `f2`) を、`str2double` を使用して、文字列から倍精度値に変換します。

- ・ コールバックが、数式を評価するためのベクトル t を生成するために、`eval` を使用して、`time` 文字列を評価します。

次のコードは、コールバックが入力を得る方法を示します。

```
% Get user input from GUI
f1 = str2double(get(handles.f1_input, 'String'));
f2 = str2double(get(handles.f2_input, 'String'));
t = eval(get(handles.t_input, 'String'));
```

[Plot] ボタンのコールバックは、不適切な入力を受け取ることで生じるエラーを回避します。入力 $f1$ 、 $f2$ 、 t が計算で使用できることを確かめるために、エディットテキストのコールバックは、入力された値を直ちにテストします。これがどのように行われるかに関しては、“ユーザー入力を数として有効にする”(p.10-11)を参照してください。

データの計算

文字列の入力パラメーターが、数値形式に変換されてローカル変数に割り当てられると、次の手順は 2 つのグラフのデータを計算することです。`plot_button_Callback` は \sin 式を使用して時間領域のデータを計算します。

```
x = sin(2*pi*f1*t) + sin(2*pi*f2*t);
```

コールバックは、時間領域のデータのフーリエ変換として周波数領域のデータを計算します。

```
y = fft(x, 512);
```

この計算の説明は、関数 `fft` を参照してください。

データのプロット

`plot_button_Callback` の最後のタスクは、2 つのプロットを生成することです。これは、以下のことを含みます。

- ・ 適切な座標軸にプロットする。たとえば、次のコードは時間座標軸にグラフをプロットします。

```
plot(handles.time_axes, t, x)
```

- ・ 関数 `plot` に適切なデータを与える。

- ・ 関数 plot が自動的にオフにするので、座標軸のグリッドをオンにする。

(plot を含む) 多くのプロット関数は、座標軸を消去し、グラフを作成する前にプロパティをリセットするので、最後に挙げた手順を実行することが必要になります。このことは、XMinorTick、YMinorTick、およびこの例の grid プロパティは、コールバックが plot を実行するとき、リセットされるので、これらのプロパティの設定のために、プロパティインスペクターを使用できないことを意味します。

次のコードのリストを見て、必要時に座標軸のハンドルにアクセスするために、どのように handles 構造体が使用されるかに注意してください。

[Plot] ボタンのコード リスト

```
function plot_button_Callback(hObject, eventdata, handles, varargin)
% hObject    handle to plot_button (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get user input from GUI
f1 = str2double(get(handles.f1_input,'String'));
f2 = str2double(get(handles.f2_input,'String'));
t = eval(get(handles.t_input,'String'));

% Calculate data
x = sin(2*pi*f1*t) + sin(2*pi*f2*t);
y = fft(x,512);
m = y.*conj(y)/512;
f = 1000*(0:256)/512;

% Create frequency plot in proper axes
plot(handles.frequency_axes,f,m(1:257))
set(handles.frequency_axes,'XMinorTick','on')
grid on

% Create time plot in proper axes
plot(handles.time_axes,t,x)
set(handles.time_axes,'XMinorTick','on')
grid on
```

ユーザー入力を数として有効にする

GUI のユーザーは、パラメーターを、テキストの文字列として 3 つのエディット テキスト ボックスに入力します。不適切な数や数以外の入力があると、グラフが情報を与えることができなかったり、生成されないこともあります。不適切な入力が処理されないようにすることは、計算を実行する多くの GUI で重要な機能になります。この GUI では、以下のことが重要です。

- ・ 3 つの入力すべてが、正または負の実数である。
- ・ t (時間) 入力は、単調に増加する、表示できる適切な長さのベクトルである。

不適切な入力に対して GUI が様々な方法で応答できるようにすることができます。以下のことが含まれます。

- ・ 無効な入力を消去し、ユーザーが他の入力をするようにする
- ・ 無効な入力を、前回の有効な値または既定値で置き換える
- ・ 入力の過程を開始するコントロールを無効化する
- ・ エラーの警告を表示し、音を鳴らす

これらの動作は、必要に応じて組合わせることができます。この例では、エディット テキスト コントロールのコールバックそれぞれは、それ自体の入力を有効にします。入力が検証に失敗すると、コールバックは [Plot] ボタンを無効にして、どのような問題に直面したかを示すために、その String を変更し、エディット テキスト コントロールのフォーカスを戻します。誤りを含む入力は、強調表示します。許容される値が再入力されると、[Plot] ボタンは、String の組を ' Plot' に戻して、有効になります。この方法はプロットのエラーを回避するので、エラー ダイアログが不要になります。

f1 と f2 の入力を確認することは、難しいことではありません。これらの入力は、正または負の実数のスカラーでなければなりません。関数 str2double はたいていのケースを取り扱いますが、数値でない場合やスカラーでない文字列表現に対しては NaN (Not a Number) を返します。関数 isreal を利用して追加のテストを行い、ユーザーが複素数（例、「4+2i」）を入力していないことを確認します。f1_input_Callback は、f1 に対するユーザー入力を確認する以下のコードを含みます。

```
f1 = str2double(get(hObject,'String'));
if isnan(f1) || ~isreal(f1)
    % isdouble returns NaN for non-numbers and f1 cannot be complex
    % Disable the Plot button and change its string to say why
    set(handles.plot_button,'String','Cannot plot f1')
```

```

set(handles.plot_button,'Enable','off')
% Give the edit text box focus so user can correct the error
uicontrol(hObject)
else
    % Enable the Plot button with its original name
    set(handles.plot_button,'String','Plot')
    set(handles.plot_button,'Enable','on')
end

```

同様のコードで、f2 の入力を確認します。

時間ベクトル入力 t を確認することは、さらに複雑になります。関数 str2double はベクトルには機能ないので、入力文字列を MATLAB 表現に変換するために関数 eval が呼び出されます。ユーザーの入力が、eval が取り扱えないものになるケースが多数考えられるので、最初のタスクは eval が成功することを確かめることです。t_input_Callback は、try と catch ブロックを用いて、以下を行います。

- try ブロック内で文字列 t_input を用いて、eval を呼び出します。
- eval が成功したら、try ブロック内で追加のテストを実行します。
- eval がエラーを生成したら、catch ブロックにコントロールを渡します。
- そのブロックにおいて、コールバックは [Plot] ボタンを無効にして、その String を ' Cannot plot t' に変更します。

try ブロックの残りのコードは、eval が返す変数 t は単調に増加する、わずか 1000 要素の数ベクトルであることを確認します。t がこれらすべてのテストにパスすると、コールバックは [Plot] ボタンを有効にして、その String を ' Plot' に設定します。いずれかのテストにパスしないときには、コールバックは [Plot] ボタンを無効にして、その String を適切な短いメッセージに変更します。以下は、コールバックからの try ブロックと catch ブロックです。

```

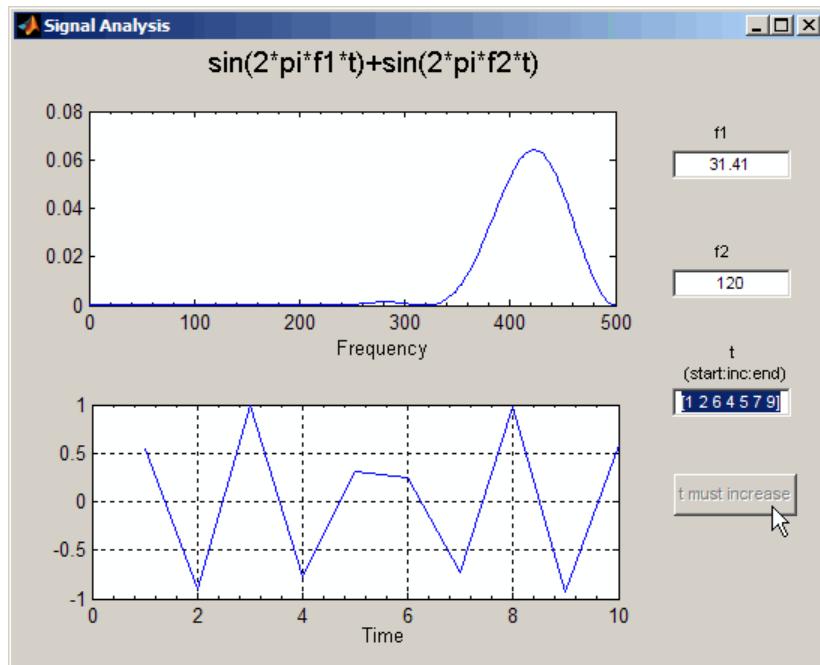
% Disable the Plot button ... until proven innocent
set(handles.plot_button,'Enable','off')
try
    t = eval(get(handles.t_input,'String'));
    if ~isnumeric(t)
        % t is not a number
        set(handles.plot_button,'String','t is not numeric')
    elseif length(t) < 2
        % t is not a vector

```

```
    set(handles.plot_button,'String','t must be vector')
elseif length(t) > 1000
    % t is too long a vector to plot clearly
    set(handles.plot_button,'String','t is too long')
elseif min(diff(t)) < 0
    % t is not monotonically increasing
    set(handles.plot_button,'String','t must increase')
else
    % All OK: Enable the Plot button with its original name
    set(handles.plot_button,'String','Plot')
    set(handles.plot_button,'Enable','on')
    return
end
% Found an input error other than a bad expression
% Give the edit text box focus so user can correct the error
uicontrol(hObject)
catch EM
    % Cannot evaluate expression user typed
    set(handles.plot_button,'String','Cannot plot t')
    % Give the edit text box focus so user can correct the error
    uicontrol(hObject)
end
```

エディット テキスト ボックスにテキストを入力して Return を押すか、GUI のどこかをクリックすると、エディット テキスト コールバックが実行します。ユーザーが [Plot] ボタンを直ちにクリックしても、エディット テキスト コールバックは [plot] ボタンのコールバックがアクティブになる前に実行します。コールバックが不適切な入力を受け取ると、[Plot] ボタンを無効にして、コールバックの実行を妨げます。最後に、ユーザーは値を再入力できるように、有効にならなかったテキストを選択してフォーカスします。

例として、単調増加ではない時間ベクトル [1 2 6 4 5 7 9] を入力したときの GUI の応答があります。



この図では、2つのプロットは、成功した最近の入力のセット $f1 = 31.41$ 、 $f2 = 120$ 、 $t = [1 2 3 4 5 7 9]$ を反映しています。時間ベクトル $[1 2 6 4 5 7 9]$ はハイライト表示されていて、ユーザーが新規の値を入力できます。上のリストでコマンド `uicontrol(hObject)` を実行すると、ハイライト表示されます。

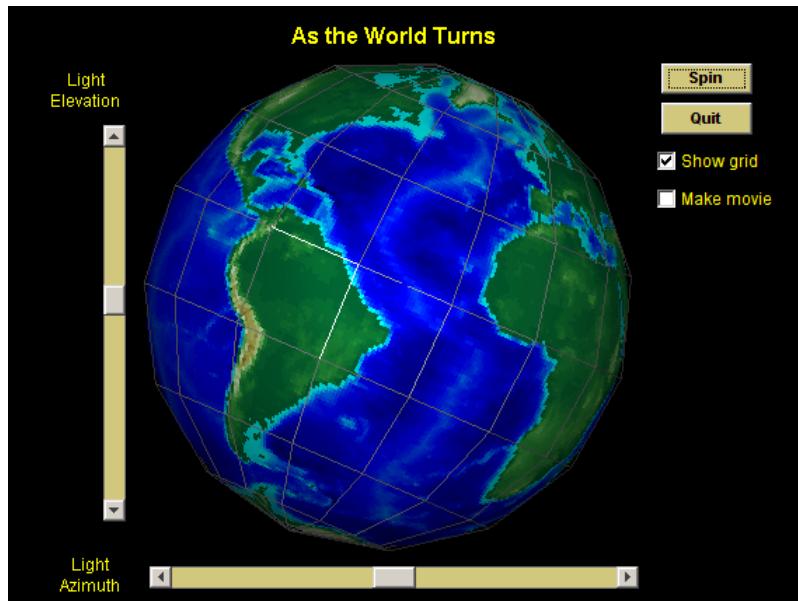
3-D 表示のアニメーションのための GUI

このセクションの内容…

- “3-D アニメーションの例について” (p.10-15)
- “3-D 地球儀の GUI の表示と実行” (p.10-16)
- “GUI の設計” (p.10-17)
- “グラフィックスの手法” (p.10-25)
- “グラフィックスをさらに調べる” (p.10-29)

3-D アニメーションの例について

この例の GUI には、3-D の軸があります。地球儀は、この軸について回転します。この例では、入力を受け取ませんが、地球儀全体の地形を評価する行列を読み込みます。この GUI は、以下のような外観をもちます。



GUI は、以下についてコントロールします。

- ・ 回転の開始と停止
- ・ 照明方向の変更
- ・ 緯線-経線グリッド(すなわち経緯網)の表示
- ・ MAT-ファイルに動画としてアニメーションを保存する
- ・ アプリケーションの終了

この例で説明されている GUI 構築の手法として、以下があります。

- ・ ボタンがクリックされるたびに、ボタンのラベル(文字列)を変更する
- ・ handles 構造体にデータを格納し、コールバックの実行をコントロールするために使用する
- ・ コールバックは、それ自身の実行を中断できる
- ・ 関連する 2 つのスライダーを用いて照明をコントロールする

さらに、コールバックのうちの 2 つが、3-D 表示の外観をどのように作成して操作するかについて説明します。

3-D 地球儀の GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを変更しようとする場合には、最初に現在のフォルダーにその M ファイルと FIG-ファイルのコピーを保存する必要があります。これを行うには現在のフォルダーへの書き込み権限が必要です。現在のフォルダーに例のファイルをコピーし、それらを開くには、以下の手順に従ってください。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に FIG-ファイルを開くには、`guide globegui` と入力するか、または ここをクリックします。
- 3 エディター内に M ファイルを開くには、`edit globegui` と入力するか、または ここをクリックします。

コンポーネントのプロパティ インスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を変更することができます。次に、GUIDE で [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存できます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUIDE で GUI を調べて実行する場合、代わりに以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 globegui の GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

GUI の設計

- ・ “関数 globegui のまとめ” (p.10-20)
- ・ “プッシュ ボタンのラベルを変更する” (p.10-21)
- ・ “Spin コールバックの中止” (p.10-22)
- ・ “アニメーションの動画を作成する” (p.10-23)

GUI には以下が含まれます。

- ・ 3 つの uipanel

- ・ 座標軸
- ・ 2 つのプッシュ ボタン
- ・ 2 つのスライダー
- ・ 2 つのチェック ボックス
- ・ スタティック テキスト

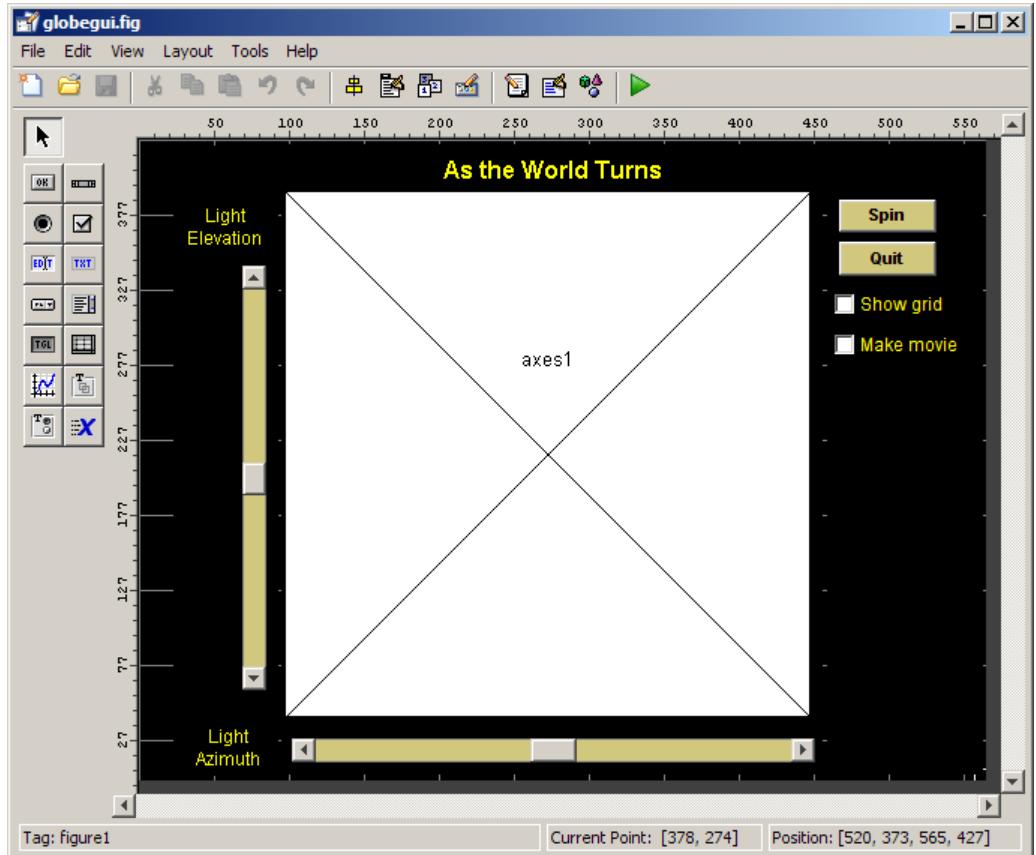
uicontrol とテキストをカスタマイズするには、プロパティ インスペクターを利用します。そのために、以下を行います。

- ・ 3 つの uipanel の `BackgroundColor`、`ForegroundColor`、`ShadowColor` の他に、Figure の `Color` を黒に設定する。これらは、コンテナとしてのみ使用されるので、表示されている必要はありません。
- ・ すべてのスタティック テキストを黄色に、uicontrol の背景を黒または黄灰色に色付けする。

メモ Solaris 2 システム上では、スライダーの背景色の設定は効果がありません。

- ・ すべての uicontrol にそれらの Tag 文字列で記憶しやすい名前を与える。
- ・ uicontrol の `FontSize` を 9 ポイントに設定する。
- ・ スライダーに対して、既定値でない `Min` と `Max` を指定する。
- ・ いくつかのコントロールに対してツールの情報を与える文字列を追加する。

GUIDE レイアウト エディターで、GUI は次のように見えます。



GUI は 3 つの uipanel を含みます。これらは全体が黒なので、この Figure でユーザーからはほとんど見えません。uipanel を利用すると、グラフィック関数がより効果的に動作するのに役立ちます。

Axes の CreateFcn (axes1_CreateFcn) は、グラフィック オブジェクトを初期化します。これは、GUI が何回開かれたかによらず、1 度だけ実行します。

[Spin] ボタンのコールバック (spinstopbutton_Callback) は、球面を回転するために while ループを含んでいるので、アニメーションを実行します。

アニメーションの途中で、ユーザーは照明方向と関数を 2 つのスライダーを用いて別々に変更できますが、表示を指定するために両方のスライダーのパラメーターが必要になるので互いの値を問合せます。

[Show grid] チェック ボックスは、経線網の Surface オブジェクトの Visible プロパティを切替えます。axes1_CreateFcn は経緯網を初期化してから、ユーザーがこのオプションを選択するまで非表示にします。

[Spin] ボタンのコールバックは、回転が停止するか、または完全に一回転した後に(いずれが最初に起こる場合でも)、ムービー フレームを蓄積するために [Make movie] チェック ボックスの値を読み込み、MAT-ファイルにムービーを保存します。地球儀を回転させる前に、[Make movie] を選択する必要があります。

以下の節では、GUI で利用される対話形式の手法を説明します。

関数 globogui のまとめ

以下の作成関数と、globogui の GUI のコールバックは、GUIDE で生成されます。これらの関数の大部分は、このアプリケーションのためにカスタマイズされます。MATLAB ヘルプ ブラウザーでこのページをお読みいただいている場合、関数名をクリックしてエディター内でスクロールします。

関数	関数の動作
globogui	GUI のフレームワークを初期化(カスタマイズされていない)
globogui_OpeningFcn	GUI のハンドル構造体を初期化(カスタマイズされていない)
globogui_OutputFcn	コマンド ウィンドウに書き込まれているものを指定します(カスタマイズされていない)
spinstopbutton_Callback	アニメーションの実行と停止を行い、動画ファイルを作成します
quitbutton_Callback	Figure を閉じることで GUI を終了します
spinstopbutton_CreateFcn	ボタンのラベルの文字列を割り当て、ハンドル構造体に格納します
axes1_CreateFcn	Axes とカラーマップを初期化し、Surface オブジェクトと Light オブジェクトを作成し、地球儀を表示します

関数	関数の動作
sunazslider_Callback	ユーザーがスライダーを動かすと光源の方位角を変更します
sunazslider_CreateFcn	光源の方位角のスライダー(カスタマイズされていない)を初期化します
sunelslider_Callback	ユーザーがスライダーを移動すると、光源の高度を変更します
sunelslider_CreateFcn	光源の高度のスライダー(カスタマイズされていない)を初期化します
showgridbutton_Callback	経緯網グリッドの表示/非表示を切り替えます
movie_checkbox_Callback	アニメーションの動画のキャプチャと保存を切り替えます
movie_checkbox_CreateFcn	動画ボタンを初期化し、その値を保存します

プッシュボタンのラベルを変更する

右上のボタンは、はじめに [Spin] とラベルされ、クリックされると [Stop] に変化します。2 度目にクリックされると [Spin] に戻ります。これは、String プロパティを handles 構造体にセル配列として格納される文字列の組と比較して行われます。以下のように、このデータを spinstopbutton_CreateFcn のハンドル構造体に挿入します。

```
function spinstopbutton_CreateFcn(hObject, eventdata, handles)
handles.Strings = {' Spin' ; ' Stop' };
guidata(hObject, handles);
```

guidata を呼び出すと、hObject を含む Figure の更新されたハンドル構造体が保存されます。hObject は、spinstopbutton プッシュボタンのオブジェクトです。GUIDE は、このオブジェクトに pushbutton1 という名前を付けています。名前は、プロパティインスペクターで Tag プロパティを変更することで変更されます。このように変更すると、GUIDE は、GUI が保存されたときに、GUI の M ファイルにコンポーネントに対するすべての参照を変更します。Tag の設定の詳細は、前の例の“座標軸の同定”(p.10-6) を参照してください。

handles.Strings のデータは、関数 spinstopbutton_Callback で使用されます。この関数は、ボタンのラベルを変更するために以下のコードを含みます。

```
str = get(hObject,'String');
state = find(strcmp(str,handles.Strings));
set(hObject,'String',handles.Strings{3-state});
```

関数 find は、ボタンの現在のラベルに一致する文字列のインデックスを返します。set を呼び出すと、ラベルの文字列が切替わります。state が 1 であるときには、3-state はラベルの文字列を 2 に設定します。state が 2 であるときには、1 に設定します。

Spin コールバックの中斷

[Spin]/[Stop] ボタンのラベルが [Stop] であるときにこのボタンをクリックすると、そのコールバックは Surface オブジェクトの回転を進めて、表示を更新するコードをループします。spinstopbutton_Callback は、そのようなイベントをリッスンするコードを含みますが、これを達成するために events 構造体は使用しません。その代わり、表示ループを終了するために、以下のコードを使用します。

```
if find(strcmp(get(hObject,'String'),handles.Strings)) == 1
    handles.azimuth = az;
    guidata(hObject,handles);
    break
end
```

表示が回転しているときにこのコード ブロックを入力すると、while ループから出てアニメーションを停止します。ただし、最初に次の回転を初期化するために、回転の現在の方位角を保存します。handles 構造体は、ハンドルだけでなく任意の変数を格納できます。ユーザーがここで [Spin] ボタンをクリックすると、キャッシュした高度の値を用いて、アニメーションは停止した位置から始まります。

ユーザーが [Quit] をクリックすると、GUI は Figure を破棄し、ただちに終了します。アニメーションが実行している間に停止によるエラーを回避するために、while ループは、Axes オブジェクトがまだ存在するかどうかを知る必要があります。

```
while ishandle(handles.axes1)
    % plotting code
    ...
end
```

関数 spinstopbutton_Callback は、while ループなしで記述できます。その場合、Figure がまだ存在しているかを判定する必要がなくなります。たとえば、グラフィックスの更新を取り扱う Timer オブジェクトを作成することができます。この例は、手法を調べませんが、MATLAB「プログラミングの基礎」ドキュメンテーションの“Using a MATLAB Timer Object”において、タイマーのプログラミングに関する情報を見つけることができます。

アニメーションの動画を作成する

[Spin] をクリックする前に、[Make movie] チェック ボックスを選択すると、アプリケーションが spinstopbutton_Callback ルーチンの while ループに表示される各フレームを記録するようになります。ユーザーがこのチェック ボックスを選択すると、次のコード ブロックが実行するので、アニメーションはさらにゆっくりと実行します。

```
filming = handles.movie;
...
if ishandle(handles.axes1) && filming > 0 && filming < 361
    globeframes(filming) = getframe; % Capture axes in movie
    filming = filming + 1;
end
```

これはチェック ボックスの値なので、handles.movie は 0 または 1 のいずれかです。1 である場合、そのコピー (filming) は、globeframes 行列に保存されたフレーム数のカウントを保持します。この行列は、Axes の CData と各フレームに対してカラー マップを含みます。while ループのコードは、[Make movie] のチェック ボックスの状態をモニターしないので、ユーザーは地球儀の回転中にムービーの保存のオンとオフを切替えることができません。

while ループが終了する前に Axes が破棄されると、ishandle の判定で getframe のエラーが生成されなくなります。

ユーザーが while ループを終了させると、コールバックは、コマンド ウィンドウに動画フレームをキャプチャした結果を表示し、MAT-ファイルに動画を書き込みます。

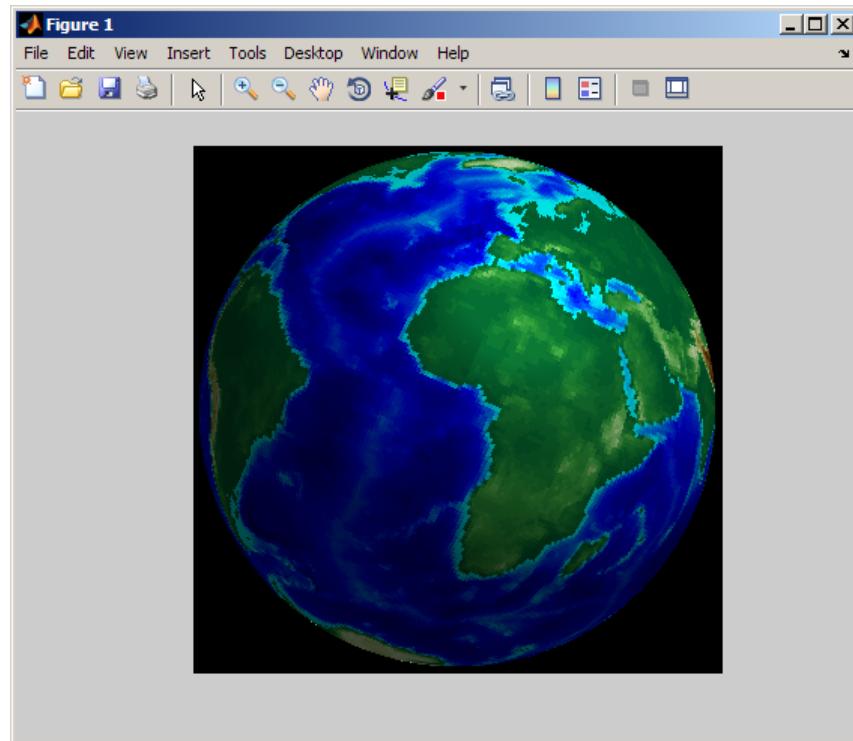
```
if (filming)
    filename = sprintf('globe%i.mat', filming-1);
    disp([' Writing movie to file ' filename]);
    save (filename, 'globeframes')
end
```

メモ GUI を利用して動画ファイルを作成する前に、現在のフォルダーに書き込み権限があることを確かめてください。

動画のファイル名の最後に、含まれているフレーム数を付けます。動画のファイル名が `globe360.mat` であるものとします。以下を用いて再生します。

```
load globe360
axis equal off
movie(globeframes)
```

次のように再生されます。



MATLAB エディターの `globegui.m` で `spinstopbutton_Callback` コードを参照するには、ここをクリックします。

グラフィックスの手法

この GUI が Handle Graphics を用いて 3-D オブジェクトを作成して表示する方法について、詳細は以下の節をお読みください。

- ・ “グラフィック オブジェクトの作成” (p.10-25)
- ・ “地球儀のテクスチャリングとカラーリング” (p.10-26)
- ・ “経緯網のプロット” (p.10-27)
- ・ “地球儀と経緯網の取得” (p.10-27)
- ・ “地球儀の照明と光源の移動” (p.10-28)

グラフィック オブジェクトの作成

関数 `axes1_CreateFcn` は、`Axes`、`Axes` に表示される 2 つのオブジェクト、地球儀の回転に作用する 2 つの `hgtransform` オブジェクトを初期化します。

- ・ `surface` を呼び出して生成される `surfaceplot` オブジェクト。このオブジェクトは地球儀を表します。
- ・ `mesh` を呼び出して生成される `surfaceplot` オブジェクト。このオブジェクトは、地理的な経緯線（緯線と経線）を表します。

これら 2 つのオブジェクトに対するデータは、関数 `sphere` により生成された長方形の x - y - z グリッドです。地球儀のグリッドは 50×50 であり、経緯網のグリッドは 8×15 です。`sphere` で返される 15×15 グリッドは、地球儀上で表示されたときにその南北方向と東西方向の長さが等しくなるように、1 つおきに行が除かれます。

`Axes` の x 、 y 、 z の範囲は、 $[-1.02 \text{ } 1.02]$ に設定されます。グラフィック オブジェクトは単位球であるので、その周りにわずかなスペースが残ります。3 つの `Axes` は、相対的にも絶対的にもすべて同じ大きさになります。経緯網グリッドも、2% 拡大されます。この値は、地球の不明瞭な構造をマップした表面で経緯網があいまいにならないために必要な最低限の拡大率です。注意して見てみると、地球儀が回転しているときに、経緯網の境界に欠けている部分がときどき見られます。

ヒント `uipanel` は、Axes と `uicontrol` を囲みます。これにより Axes は、Axes が含まれている `uipanel` の子になります。`uipanel` に Axes が含まれていることにより、Figure の一部をまとめることで、グラフィックレンダリングを高速化します。MATLAB のグラフィックス関数はグラフィックスを再描画します。

地球儀のテクスチャリングとカラーリング

`axes1_CreateFcn` のコードは、`FaceColor` プロパティを 'texturemap' に設定することで、地球儀に対する `CData` を、 180×360 (1 度) の地形グリッド `topo` に設定します。表面の質感を出すためにイメージやグリッドを使用できます。以下のように、`Surface` プロパティを、ユーザーが設定すべきプロパティごとに 1 つの要素を含む構造体として指定します。

```
props.FaceColor= 'texture' ;
props.EdgeColor = 'none' ;
props.FaceLighting = 'gouraud' ;
props.Cdata = topo;
props.Parent = hgrotate;
hsurf = surface(x, y, z, props);
colormap(cmap)
```

ヒント パラメーター セットの値を含む MATLAB 構造体を作成し、パラメーターと値の組ではなくそれらの構造体を関数に与え、後で使用するためにそれらの構造体を MAT-ファイルに保存することができます。

関数 `surface` は、Axes に表面をプロットします。`Surface` の `Parent` を `hgrotate` に設定すると、`Surface` オブジェクトを地球儀を回転する `hgtransform` のコントロール下に置きます。“地球儀と経緯網の取得”(p.10-27) の説明を参照してください。`EdgeColor` を 'none' に設定することで、地球儀は、グリッドライン(既定では黒で表示)は表示せず face カラーのみを表示します。関数 `colormap` は、表面に対するカラーマップを、コードで定義された 64×3 カラーマップ `cmap` に設定します。これは、地形の表示に適切です。さらに多くのカラーを使用できますが、テクスチャマップ(1×1 度の分解能)の相対的な粗さを考えて 64 カラーで十分です。

経緯網のプロット

地球儀のグリッドとは異なり、経緯網グリッドは、面のカラーではなくグレーの境界で表示されます。[Show grid] ボタンをクリックすることで、経緯網グリッドのオンとオフを切替えます。地形図のように、これは Surfaceplot オブジェクトです。ただし、以下のように、関数 `surface` ではなく、関数 `mesh` で作成します。

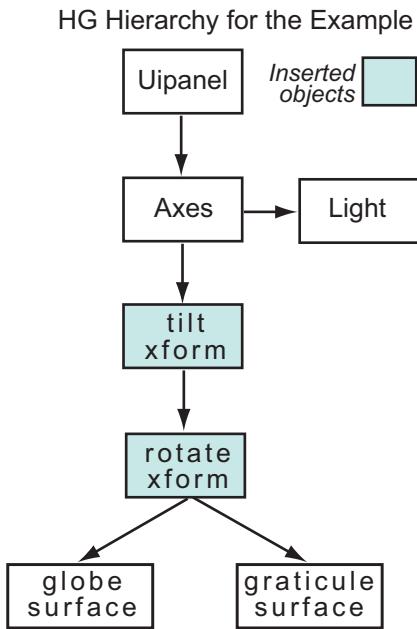
```
hmesh = mesh(gx, gy, gz, 'parent', hgrotate, ...
    'FaceColor', 'none', 'EdgeColor', [.5 .5 .5]);
set(hmesh, 'Visible', 'off')
```

[Show grid] ボタンの状態は、はじめは `off` です。そのため、経緯網は表示されません。[Show grid] は、Mesh オブジェクトの `Visible` プロパティを切替えます。

上述したように、プロットする前に経緯網を 2 % 拡大すると、地形表面があいまいにならざります。

地球儀と経緯網の取得

地球儀と経緯網は、`hgtransform` オブジェクトの組のコントロール下で 1 つのオブジェクトであるかのように回転します。Figure 内で、HG オブジェクトはこの階層にセットアップされます。



傾きの変換では、 x 軸について 0.5091 ラジアン (23.44 度) 回転させます。この角度は、回転に関する地軸の傾きです。回転変換は、最初は既定値の恒等行列をもちます。spinstopbutton_Callback は次に行列を更新して、 z 軸について繰り返しごとに 0.01745329252 ラジアン (1 度) ずつ回転させます。このとき、以下のコードを利用します。

```

az = az + 0.01745329252;
set(hgrotate,'Matrix',makehgtransform('zrotate',az));
drawnow % Refresh the screen
  
```

地球儀の照明と光源の移動

Light オブジェクトは、はじめは左から地球儀に光をあてます。2 つのスライダーは、光の位置をコントロールします。地球儀が静止または回転していても、ユーザーは光の位置を操作できます。light は、Axes の子であるので、hgtransform のいずれにも影響されません。light への呼び出しでは、その高度と方位角以外のパラメーターは使用しません。

```
hlight = camlight(0,0);
```

`light` を作成後、`axes1_CreateFcn` は、他のコールバックが `handles` 構造体に対して必要となる、いくつかのハンドルとデータを追加します。

```
handles.light = hlight;
handles.tform = hgrotate;
handles.hmesh = hmesh;
handles.azimuth = 0. ;
handles.cmap = cmap;
guidata(gcf, handles);
```

`guidata` を呼び出すと、`handles` に追加されるデータがキャッシュされます。

スライダーを移動して、光源の高度と方位角の両方を設定できます。ただし、各スライダーではいずれか 1 つのみが変更されます。光の高度を変化させるためのコールバックのコードは、以下のようになります。

```
function sunelslider_Callback(hObject, eventdata, handles)

hlight = handles.light; % Get handle to light object
sunaz = get(handles.sunazslider, 'value'); % Get current light azimuth
sunel = get(hObject, 'value'); % Varies from -72.8 -> 72.8 deg
lightangle(hlight, sunaz, sunel) % Set the new light angle
```

光の方位角のスライダーのコールバックも、同じように動作します。このコールバックは、`lightangle` への呼び出しで、値の変更を防ぐために高度スライダーの設定をクエリします。

グラフィックスをさらに調べる

以下の様々な方法で、`globegui` の表示を改良できます。

- 海面からの地形の高低を示す標高とカラーがどのように対応しているかを示すカラーバーを追加する。
カラーバーをセットアップするために GUIDE を使用することはできませんが、`axes1_CreateFcn` ではセットアップすることができます。カラーバー利用の詳細は、`colorbar` のリファレンス ページと、MATLAB「グラフィックス」ドキュメンテーションの“Adding a Colorbar to a Graph”を参照してください。
- 地球儀上で観測者に対して最も近い点の経線の値を表示する。

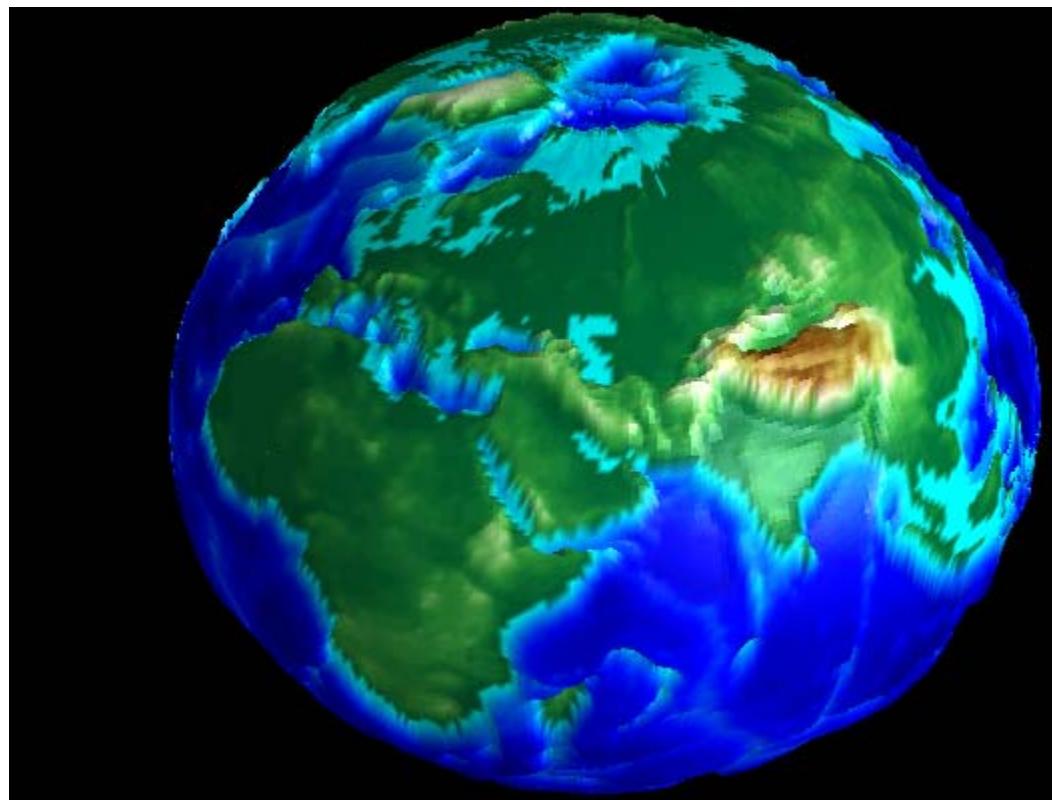
edit または text スタイルの uicontrol を使用して、(グリニッジ子午線から東または西方向への度単位で方位角を表して) 現在の方位角を用いて、関数はその while ループ内で更新できます。この値を handles.azimuth に格納します。回転 hgtransform を更新するために適切なコールバックをもつエディット ボックスを用いてこれを行う場合、ユーザーは地球儀の経線を対話的に更新できます。

ヒント 経線は地球儀が回転しているときに、手動で更新することができます。これを行うために、spinstopbutton_Callback の一時変数 az を削除することができます。この一時変数は handles.azimuth で置き換えます。

- ・ 真直ぐなエッジではなく曲がったエッジを用いて、経緯網に滑らかさを与える。
関数 sphere の仕様では、非常に粗い経緯網のグリッドを返します。この経緯網のグリッドでは、滑らかなグリッドラインを生成するために必要となる、グリッドライン間の値をもちません。これを解決するには、 x 、 y 、 z 座標のベクトルを利用します。これらのベクトルの値を単位球面よりもわずかに大きくなるようにスケーリングして、経線と緯線をユーザーが個別に生成できます。
- ・ 地球儀の表面からの光の反射をモデリングする。
関数 material を用いて、地球儀を明るく照らしたり、暗くしたりすることができます。詳細は、MATLAB「3 次元可視化」ドキュメンテーションの“Reflectance Characteristics of Graphics Objects”を参照してください。
- ・ 地球儀に 3-D の地形起伏を追加する。
topo は、地球儀の Surface オブジェクトの CData として使用します。topo グリッドは 1 にスケーリングされ、地球儀の Surface オブジェクトの ZData が topo グリッドに割り当てられます。sphere からの出力をプロットするには、関数 surfl を使用します。

ヒント 地球儀に、地形の起伏を加えるには、topo と同じサイズの球面グリッドを使用します。次に、topo の値を 1.0 未満にスケールして、それらを x 、 y 、 z 行列の要素にします。

結果は、次の図のようになります。この中で地形の起伏は、地球儀の半径に 10% 追加するようにスケールされるので、地形の起伏が非常に誇張されます。



テーブルのデータを対話的に調べる GUI

このセクションの内容…

- “tablestat の例について” (p.10-32)
- “tablestat の GUI の表示と実行” (p.10-34)
- “GUI の設計” (p.10-36)
- “Tablestat の拡張” (p.10-52)

tablestat の例について

この例では、以下を含む、対話的にデータを調べるためのコールバック プログラムの作成方法を示します。

- ・ テーブルとプロットを初期化する Opening 関数
- ・ ユーザーがデータ観測を選択するときに、選択されたデータをリアルタイムでプロットする `uitable` のセルを選択するコールバック
- ・ データの表示方法を変えた線グラフを作成するポップアップ メニューのコールバック
- ・ Axes に付加するコンテキスト メニュー

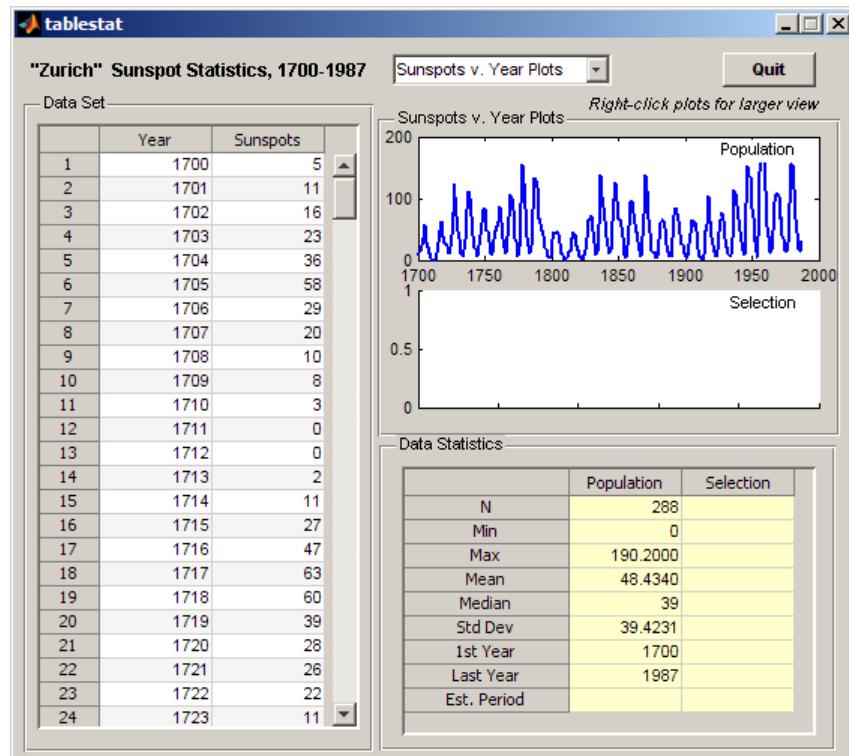
データセット全体または選択したデータに対して、異なる種類のグラフを別の Axes にプロットしたり、さらにフーリエ変換が時系列データの周期性をどのように識別できるかを見るために GUI を利用します。GUI には以下が含まれます。

- ・ 2 列（日付と観測）のデータをもつ黒点観測の表
- ・ 2 つ目のテーブルは、データと、そのデータからユーザーが選択したサブセットの統計量をまとめています。
- ・ データと、データからユーザーが選択したサブセットの時系列とフーリエ解析をプロットする 2 つの Axes。それぞれの Axes は、新規の Figure に対してそのコンテンツを出力するコンテキスト メニューをもちます。
- ・ 表示されているデータ グラフのタイプを変更するポップアップ メニュー。
- ・ 統計量の列を更新し、強調表示された観測をプロットする、セル選択のコールバック。
- ・ データ セット全体や選択したデータの周期性を描くプロット

- 別々の Figure ウィンドウに各コンテンツを表示させる、Axes のコンテキストメニュー

この GUI または更新した GUI の利用 – 周期的な事象を含む時系列データの解析と可視化のために使用します。

テーブルと Axes の他に、GUI は 3 つのパネルの機能として、アプリケーションを停止するプッシュ ボタン、スタティック テキスト、データの解析とプロットのための機能があります。この GUI を開いた外観は以下のようになります。



メモ tablestat の例は、MATLAB の sunspots デモとデータ セットを使用します。
MATLAB ヘルプ ブラウザーにそのデモ (GUI-ベースではない) を表示するには、ここをクリックしてください。

tablestat の GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルと FIG-ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下の手順を行います。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に FIG-ファイルを開くには、`guide tablestat` と入力するか、または ここをクリックします。
- 3 エディター内に M ファイルを開くには、`edit tablestat` と入力するか、または ここをクリックします。

コンポーネントのプロパティ インスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を修正してから、GUIDE から [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存することができます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUIDE で GUI を調べて実行する場合、代わりに以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 `tablestat` GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

関数 Tablestat のまとめ

以下の表では、tablestat.m のすべての関数について、関数の動作、GUIDE が関数に対する宣言を作成するかどうかを説明します。3 列目に示すように、GUIDE が生成するコールバックの大部分がカスタマイズされています。MATLAB エディターにそのコードを表示するには、関数名をクリックしてください。

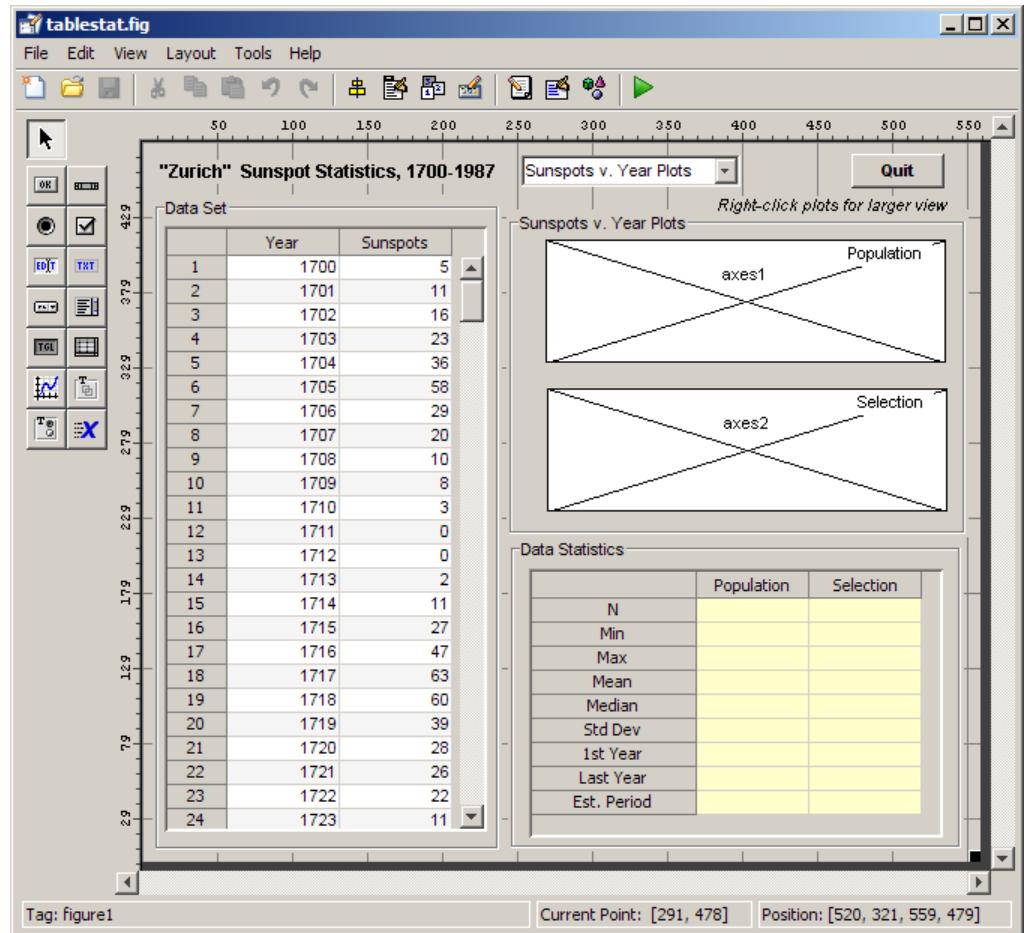
関数名	関数の動作	GUIDE による生成
tablestat	メイン関数	はい。カスタマイズされていません
tablestat_OpeningFcn	ハンドルにメンバーを追加し、母集団の統計データとプロットを生成します	はい
tablestat_OutputFcn	tablestat を終了する（使用されていない）とき、値を返します	はい。カスタマイズされていません
data_table_CellSelectionCallback	テーブル インデックスを固有な行番号に変換し、選択統計量とプロットを生成します	はい
plot_type_Callback	ユーザーが新規のプロット タイプを選択するときに表示を更新します	はい
plot_type_CreateFcn	ポップアップ メニューの作成中にその外見を管理します	はい。カスタマイズされていません
plot_ax1_Callback	Axes1 プロットのコピーをもつ新規 Figure を作成します	はい
plot_ax2_Callback	Axes2 プロットのコピーをもつ新規 Figure を作成します	はい
refreshDisplays	データ統計量テーブルとプロットの更新をコントロールします	いいえ

関数名	関数の動作	GUIDE による生成
setStats	母集団または抽出データの統計量を計算します	いいえ
plotPeriod	プロットを生成します (時系列またはピリオドグラム)	いいえ
quit_Callback	Figure を閉じます	はい

GUI の設計

- ・ “データ テーブルの初期化” (p.10-41)
- ・ “データ統計量の計算” (p.10-42)
- ・ “データ プロットのタイプの指定” (p.10-43)
- ・ “データ選択への応答” (p.10-45)
- ・ “統計量のテーブルとグラフの更新” (p.10-47)
- ・ “新規の Figure ウィンドウにグラフを表示する” (p.10-48)

GUIDE レイアウト エディターにおいて、tablestat GUI の外見は次のようにになります。



レイアウトを生成するには、GUIDE とプロパティ インスペクターで以下の手順を実行して、以下のオブジェクトを生成します。

- 1 [パネル] ツール を使用すると、上に示す位置に 3 つの uipanel をドラッグ アウトします。それらの Tag プロパティ (uipanel1, uipanel2, uipanel3) に対して既定値を保持します。順に、作成します。
 - ・ 左側の長いパネル。プロパティ インスペクターで Title を Data Set に名前を変更します。

- 最初のパネルの高さの 1/2 の右下のパネル。プロパティ インスペクターで Title を Data Statistics に名前を変更します。
- [Data Statistics] パネルの上のパネル。プロパティ インスペクターで Title を Sunspots v. Year Plots に名前を変更します。このパネルは、表示されているプロットの種類が変更されるときにパネル名を変更します。

2 [テーブル] ツール  を利用すると、[データ セット] パネル内に uitable をドラッグ アウトします。プロパティ インスペクターで次のプロパティを既定値ではない値に設定します。

- ColumnName を Year と Sunspot に設定します。
- Data。これは、以下の節 “データ テーブルの初期化” (p.10-41) に述べる ように設定できます。
- Tag を data_table に設定します。
- TooltipString を Drag to select a range of 11 or more observations に 設定します。
- GUIDE は、CellSelectionCallback を data_table_CellSelectionCallback に自動的に設定して、ユーザーが “鉛筆と紙” が描かれたアイコン  をクリックするときに M ファイルに宣言します。

3 [Data Statistics] パネル内に、2 つめの uitable をドラッグ アウトします。プロパティ インスペクターで、これらのプロパティを設定します。

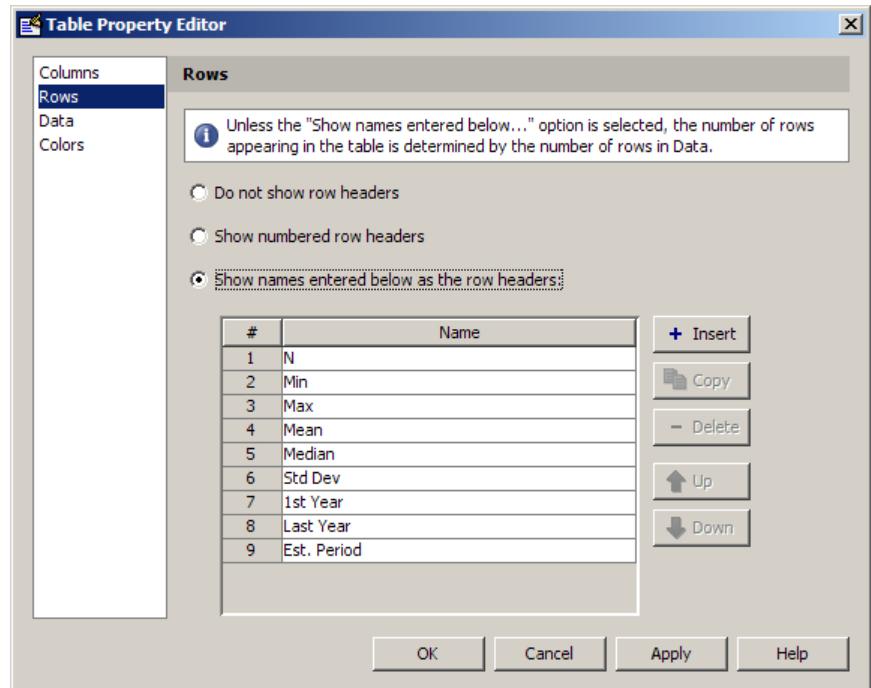
- BackgroundColor を yellow に設定します (色の選択を利用)。
- ColumnName を Population と Selection に設定します。
- Tag を data_stats に設定します。
- TooltipString を statistics for table and selection に設定します。
- RowName を 9 つの文字列、N、Min、Max、Mean、Median、Std Dev、1st Year、Last Year、Est. Period に設定します。

以下のように、テーブル プロパティ エディターを用いてこれらのラベルを簡単に設定できます。

- [Data Statistics] テーブルをダブルクリックして、プロパティ インスペクターに開きます。

- b プロパティ インスペクターで、RowName プロパティの右にあるテーブル プロパティ エディター アイコン  をクリックしてテーブル プロパティ エディターを開きます。
- c テーブル プロパティ エディターで、左側の列のリストから [行] を選択します。
- d 下にあるラジオ ボタン [行ヘッダーとして下記に入力された名前を表示] を選択します。
- e データ入力ペインの個々の行に順に上にリストされる 9 つの文字列を入力して、[OK] をクリックします。

テーブル プロパティ エディターは、閉じる前には次のようになっています。



[Data Statistics] の表は、コールバックを使用しません。

- 4 [Axes] ツール  を使用して、[Sunspots v. Year Plots] パネルの上半分内に、Axes をドラッグ アウトします。その名前は axes1 として残します。

- 5 2つめの Axes をドラッグ アウトします。[Sunspots v. Year Plots] パネル内で、1つめの Axes の真下に、その名前は axes2 として残します。
- x Axis のラベルを表示するために、各座標軸の下に十分な余白を残します。
- 6 ラベルで Axes を識別します。[Text] ツールを使用して、上側の Axes (axes1) の右上隅に小さな長方形をドラッグ アウトします。この長方形をダブルクリックして、プロパティインスペクターにおいて、その String プロパティを Population に変更し、その Tag プロパティを poplabel に変更します。
- 7 下側の Axes (axes2) に 2 つ目のラベルを置きます。この Text オブジェクト Selection の名前を変更して、その Tag プロパティを sellabel に設定します。
- 8 GUI のタイトルを作成します。[Text] ツールを使用して、データ テーブルの上の GUI の左上にスタティック テキスト オブジェクトをドラッグ アウトします。スタティック テキスト オブジェクトをダブルクリックして、プロパティインスペクターにおいて、String プロパティを "Zurich" Sunspot Statistics, 1700-1987 に、FontWeight プロパティを bold に変更します。
- 9 Axes の上にプロンプトを追加します。[Sunspots v. Year Plots] パネルの真上、その端の近くに、text ラベルを置きます。その Tag プロパティを newfig に、String プロパティを Right-click plots for larger view に、FontAngle プロパティを Italic に変更します。
- 10 ポップアップ メニューを作成してプロットするグラフの種類を指定します。[ポップアップ メニュー] ツール  を使用して、[Sunspots v. Year] パネルの真上にポップアップ メニューをドラッグ アウトし、パネルの左端に整列させます。プロパティインスペクターにおいて、次のプロパティを設定します。
- String を以下に設定。

Sunspots v. Year Plots
FFT Periodogram Plots
 - Tag を plot_type に設定。
 - Tooltip を Choose type of data plot に設定。
 - Callback プロパティのアイコンをクリックします。これは、M ファイル plot_type_Callback に宣言を作成します。このファイルに、後でユーザーがコードを追加します。

- 11 プッシュ ボタン ツール を選択し、Figure の右上にプッシュ ボタンをドラッグ アウトします。プロパティ インスペクターにおいて、その名前を [Quit] に変更して、そのコールバックを以下のように設定します。
- この長方形をダブルクリックして、プロパティ インスペクターにおいて、Tag プロパティを quit に変更し、String プロパティを Quit に変更します。
 - Callback プロパティをクリックして、M ファイル tablestat.m にボタンに対するコールバックを作成します。GUIDE は、項目 [Quit] の Callback を、quit_Callback に設定します。
 - M ファイルにおいて、関数 quit_Callback に以下を入力します。

```
close(ancestor(hObject,'figure'))
```

- 12 GUIDE で GUI を保存して tablestat.fig と名前を付けます。この操作でも、M ファイルを tablestat.m として保存します。

データ テーブルの初期化

テーブルにデータを読み込むために Opening 関数を使用することができます。ただし、この例では GUIDE を利用して [Data Set] テーブルにデータを置きます。そのため、Figure を保存した後、データは Figure の一部になります。テーブル データを初期化すると、テーブルは、テーブルが含む変数と同じ、行数と列数をもつことになります。

- 1 コマンド ウィンドウで、黒点のデモのデータ セットにアクセスします。次のように入力します。

```
load sunspot.dat
```

288×2 の double 配列である、変数 sunspot が MATLAB ワークスペースに表示されます。

- 2 [Data Set] テーブルをダブルクリックして、データ テーブルに対するプロパティ インスペクターを開きます。
- 3 プロパティ インスペクターで、Data プロパティの右側のテーブル エディター アイコン をクリックしてテーブル プロパティ エディターを開きます。
- 4 テーブル プロパティ エディターにおいて、左側の列のリストから [テーブル] を選択します。

- 5 下のラジオ ボタン [データ値を選択した以下のワークスペース変数に変更] を選択します。
- 6 ラジオ ボタンの下のボックス内のワークスペース変数のリストから、sunspot を選択して [OK] をクリックします。

GUIDE はテーブルに黒点のデータを挿入します。

メモ このような GUI を設計しているときに、GUI のユーザーが黒点データの代わりに他の数値データを読み込むことができるようにする必要がある場合、MATLAB ワークスペースを検索し、ユーザーに変数のリストを提示する方法が必要です。GUIDE の例 “リストボックスからワークスペース変数にアクセス” (p.10-61) は、GUIDE を使用してこの種の機能を与える方法を説明します。ある次元をもつクラス double のみをリストするために、その機能を拡張できます。

データ統計量の計算

Opening 関数は、データ テーブルからあらかじめ読み込まれたデータを取得し、setStats サブ関数を呼び出して、母集団の統計を計算して返します。ユーザーがデータ テーブルから 10 行以上を選択すると、data_table_CellSelectionCallback は同じ動作を実行します。これら 2 つの呼び出しの違いは、与えられる入力データと、[Data Statistics] テーブルのどの列が計算されるかということだけです。以下は、関数 setStats です。

```
function stats = setStats(table, stats, col, peak)
% Computes basic statistics for data table.
%   table  The data to summarize (a population or selection)
%   stats  Array of statistics to update
%   col    Which column of the array to update
%   peak   Value for the peak period, computed externally

stats{1,col} = size(table,1);      % Number of rows
stats{2,col} = min(table(:,2));
stats{3,col} = max(table(:,2));
stats{4,col} = mean(table(:,2));
stats{5,col} = median(table(:,2));
stats{6,col} = std(table(:,2));
stats{7,col} = table(1,1);        % First row
stats{8,col} = table(end,1);     % Last row
```

```

if ~isempty(peak)
    stats{9,col} = peak; % Peak period from FFT
end

```

メモ `uitable` にデータを割り当てるときに、`setStats` のコードに示すように、セル配列を使用します。ただし、`uitable` から取得したデータがすべて数値である場合は、数値配列に割り当てることができます。`uitable` のデータをセル配列に格納すると、テーブルは、数、文字列、またはそれらの組合せを保持することができます。

`stats` 行列は 9×2 セル配列です。各行は、`table` 引数から計算される別々の統計量です。最後の統計量は、`setStats` で計算されません。この統計量は、関数 `plotPeriod` で FFT ピリオドグラムを計算してプロットし、さらに `peak` パラメーターとして `setStats` に渡されます。

データプロットのタイプの指定

どの時点においても、`tablestat` のユーザーが、`plot_type` ポップアップメニューで表示できるようにするために 2 種類のプロットのいずれかを選択できます。

- ・ Sunspots v. Year Plots – 年ごとの黒点の発生を表示する時系列線グラフ（既定値）。
- ・ ピリオドグラム プロット – 何年かのうちのサイクルの長さにより、黒点の発生の FFT で導出されたパワー スペクトルを表示するグラフ

メモ フーリエ変換の詳細は、MATLAB「数学」ドキュメンテーションの Fourier Transforms と “The FFT in One Dimension” を参照してください。

プロットタイプが変わると、1 つまたは両方の `Axes` がリフレッシュします。これらは、常に、同じ種類のプロットを示しますが、下部の `Axes` は最初は空であり、ユーザーがデータテーブルの少なくとも 11 行を選択するまでグラフを表示しません。

`plot_type` コントロールのコールバックは `plot_type_Callback` です。GUIDE がコールバックを生成します。プロットを適切に更新するには、コールバックにコードを追加する必要があります。例の M ファイルにおいて、コールバックは以下のコードで構成されます。

```

function plot_type_Callback(hObject, eventdata, handles)
% hObject    handle to plot_type (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% ---- Customized as follows ----
% Determine state of the pop-up and assign the appropriate string
% to the plot panel label
index = get(hObject,'Value'); % What plot type is requested?
strlist = get(hObject,'String'); % Get the choice's name
set(handles.uipanel3,'Title',strlist(index)) % Rename uipanel3

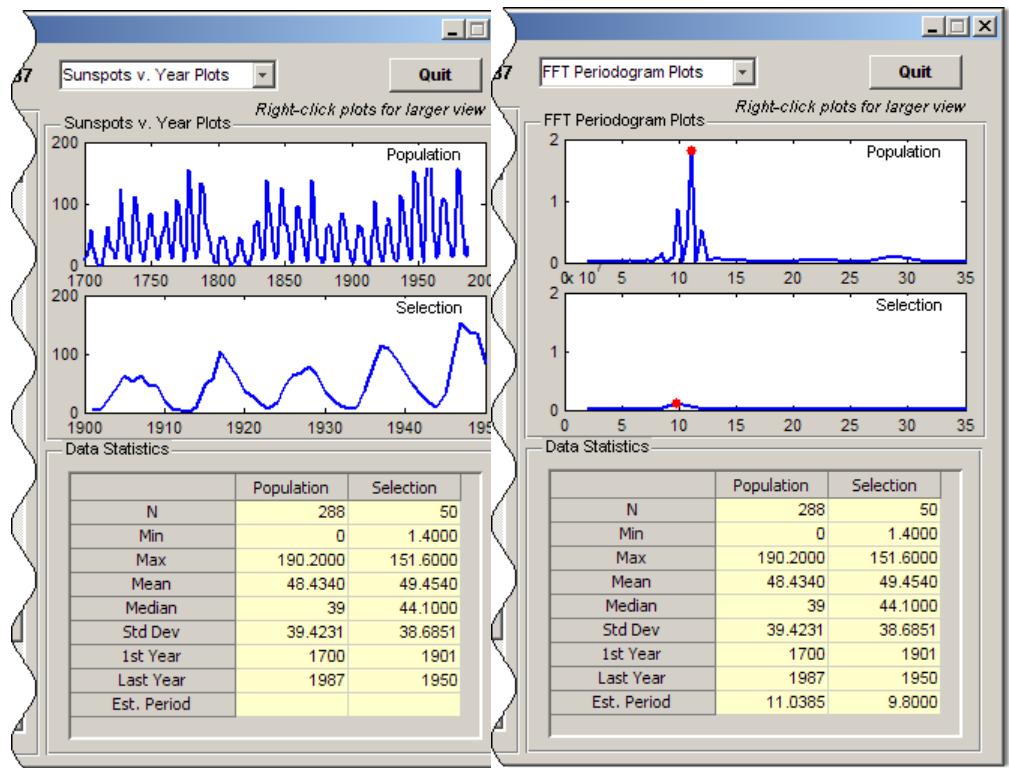
% Plot one axes at a time, changing data; first the population
table = get(handles.data_table,'Data'); % Obtain the data table
refreshDisplays(table, handles, 1)

% Now compute stats for and plot the selection, if needed.
% Retrieve the stored event data for the last selection
selection = handles.currSelection;
if length(selection) > 10 % If more than 10 rows selected
    refreshDisplays(table(selection,:), handles, 2)
else
    % Do nothing; insufficient observations for statistics
end

```

関数は、Data Statistics のテーブルとプロットを更新します。更新を実行するために、関数は関数 `refreshDisplays` を 2 度呼び出します。これは、M ファイルに追加されるカスタム コードです。2 つの呼び出しの間に、関数 `refreshDisplays` は `handles` 構造体の `currSelection` メンバーから現在選択されている行のインデックスを取り出します。行インデックスは `data_table_CellSelectionCallback` に格納されています。

以下の 2 つの図において、プロットの種類を切替える効果を見ることができます。左側のものは Sunspots v. Year プロットを示し、右側のものは FFT ピリオドグラムプロットを示します。両方の場合の選択は、1901 ~ 1950 年です。



データ選択への応答

[Data Set] テーブルには、[Year] と [Sunspots] という 2 列があります。データ テーブルの Cell Selection Callback は、ユーザーがどの列を強調表示するかに問わらず、その 2 列目からのデータを解析します。関数 setStats (GUIDE で生成されない) は、右側の [Data Statistics] テーブルに挿入するために第 2 列から観測の統計量を計算します。関数 plotPeriod (GUIDE で生成されない) は、生データまたはそのフーリエ変換をプロットします。

関数 data_table_CellSelectionCallback は、ユーザーが選択しているデータ範囲に対するアプリケーションの応答を管理します。範囲は、連続した行、または行の別々のグループになることができます。Ctrl キーを押していると、連続していない行を選択に追加できます。テーブル内で、ユーザーがマウスの左ボタンを押している

間は、Cell Selection Callback がトリガーされるので、選択が完了するまでに、選択した統計量と下側のプロットがリフレッシュされます。

選択データは、mouseDown イベント（データ テーブルでのマウス ドラッグ）中に生成されます。uitable は、セル インデックスのこの stream（ただし、セル値ではない）を eventdata 構造体により、data_table_CellSelectionCallback コールバックに渡します。コールバックのコードは、eventdata の Indices メンバーからインデックスを読み込みます。

(eventdata のそれぞれの新しい値に対して) コールバックが実行するとき、コールバックはイベント データから行のセットを作成します。

```
selection = eventdata.Indices(:, 1);
selection = unique(selection);
```

イベント データは、現在選択されている各テーブル セルに対して [row, column] のインデックスの列を含みます。行ごとに 1 つのセルがあります。前述のコードは、列のインデックスを削除し、選択された行のリストに対するインデックスのリストを削除します。すると、これは MATLAB 関数 unique を呼び出して、行の重複している要素を削除します。重複は、ユーザーが両方の列を選択するときに生じます。たとえば、eventdata.Indices は、以下を含むものとします。

```
1   1
2   1
3   1
3   2
4   2
```

これは、ユーザーが列 1 (Year) の最初の 3 行と、第 2 列の数を選択するときに Ctrl キーを押すことで、列 2 (Sunspots) の行 3 と行 4 を選択したこと示します。先行するコードは、そのインデックスを以下のベクトルに変換します。

```
1
2
3
4
```

このベクトルは、選択された行をすべて列挙します。選択した行数が（今の場合のように）11 行よりも少ない場合、このような小さなサンプルの統計量を計算することは有効ではないので、コールバックが返ります。

11 行以上が選択されているときには、データ テーブルが取得され、選択は handles 構造体にキャッシュされます。さらに、関数 refreshDisplays が呼び出され、選択統計量とプロットを更新し、ユーザーが選択したテーブルの一部を渡します。

```
table = get(hObject,'Data');
handles.currSelection = selection;
guidata(hObject,handles)
refreshDisplays(table(selection,:),handles,2)
```

ユーザーはプロットのタイプを変更することで選択データを再プロットできるので、選択している行のリストをキャッシュすることが必要です。plot_type_Callback はデータ テーブルのイベント データにアクセスしないので、最も最近の選択のコピーを必要とします。

統計量のテーブルとグラフの更新

以下の場合、コードは上述の Data Statistics テーブルとグラフを更新する必要があります。

- GUI が、tablestat_OpeningFcn において初期化される。
- ユーザーは、データ テーブルのセル、data_table_CellSelectionCallback を選択する。
- ユーザーが、plot_type_Callback で異なるプロット タイプを選択する。

それぞれの場合において、関数 refreshDisplays は更新を取り扱うために呼び出されます。これは、今度は、他の 2 つのカスタム関数を呼び出します。

- setStats — 選択に対して統計量を計算し、それらを返します。
- plotPeriod — 適切な Axes に現在要求されている種類のグラフをプロットします。

関数 refreshDisplays は、現在のプロットの種類を特定し、グラフをプロットする Axes を指定します。plotPeriod と setStats を呼び出した後、これは、再び計算された統計量をもつテーブル Data Statistics を更新します。以下は、refreshDisplays のコードです。

```
function refreshDisplays(table, handles, item)
if isEqual(item,1)
    ax = handles.axes1;
elseif isEqual(item,2)
```

```

    ax = handles.axes2;
end
peak = plotPeriod(ax, table, ...
    get(handles.plot_type, 'Value'));
stats = get(handles.data_stats, 'Data');
stats = setStats(table, stats, item, peak);
set(handles.data_stats, 'Data', stats);

```

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読み頂いている場合、上のアンダーラインされた関数名をクリックして、MATLAB エディターでそれらの完成した M コード（コメントを含む）を参照してください。

新規の Figure ウィンドウにグラフを表示する

- ・ “2 つのコンテキストメニューの作成” (p.10-49)
- ・ “Axes にコンテキストメニューを付加する” (p.10-50)
- ・ “コンテキストメニューのコールバックのコード化” (p.10-50)
- ・ “新規ウィンドウ機能でのプロットの利用” (p.10-51)

tablestat GUI は、ユーザーがいずれかの Axes を右クリックして、ポップアップメニュー項目 [Open plot in new window] を選択するときに、新規 Figure ウィンドウ内にそのグラフのいずれかを拡大して表示するコードを含みます。プロット パネルの上のステイック テキスト文字列 (newfig タグ付き)、[Right-click plots for larger view] は、この機能が利用できることをユーザーに知らせます。

Axes は、以下により応答します。

- 1 新規の Figure ウィンドウの作成
- 2 これらのコンテンツを新規の Figure を親にもつ新規の座標軸にコピーする
- 3 Figure の幅の 90 % を使用するために新規の Axes をサイズ変更する
- 4 タイトルの文字列を作成して新規の Figure に表示する
- 5 後で使用できるように、あるいは破棄するために、Figure と Axes のハンドルを handles 構造体に保存する

メモ ハンドルは、両方のプロットに対して保存されていますが、それらのそれぞれに対して新規の Figure が作成される度に、旧いハンドルがあれば新規のハンドルで置き換えます。そのため、以前の Figure は GUI からアクセスできなくなります。

2つのコンテキストメニューの作成. 2つのコンテキストメニューを作成するには、GUIDE の [ツール] メニューから、[メニュー エディター] を選択します。2つのコンテキストメニューを作成後、それぞれの Axes, axes1 と axes2 に 1つずつ追加します。メニュー エディターにおいて、それぞれのメニューに対して、以下を行います。

1 [コンテキストメニュー] タブをクリックして、ユーザーが作成しているメニューのタイプを選択します。

2 [新規コンテキストメニュー] アイコンをクリックします。

これにより、メニュー エディターのワークスペースに `untitled` というコンテキストメニューが作成されます。このコンテキストメニューには、メニュー項目がなく、GUI オブジェクトにはまだ追加されていません。

3 新規のメニューを選択し、[メニュー プロパティ] パネルの [タグ] エディット フィールドで、`plot_axes1` と入力します。

4 [新規メニュー項目] アイコンをクリックします。

メニュー項目は、メニュー エディターのワークスペースの項目 `plot_axes1` の下に表示されます。

5 [メニュー プロパティ] パネルにおいて、[ラベル] に `Open plot in new window` と入力し、[タグ] に `plot_ax1` と入力します。この項目に対しては、何も設定しないでください。

6 最後の 4 つの手順を繰り返して、2 つめのコンテキストメニューを作成します。

- ・ メニュー [plot_axes2] に対して [タグ] を作成します。
- ・ その下にメニュー項目を作成してその [ラベル] を `Open plot in new window` として、その [タグ] は `plot_ax2` とします。

7 [OK] をクリックして、メニューを保存し、メニュー エディターを終了します。

メニュー エディターの利用の詳細は、“メニューの作成”(p.6-99)を参照してください。

Axes にコンテキストメニューを付加する。直前に作成したコンテキストメニューを Axes に追加します。

- 1 GUIDE レイアウトエディターにおいて、axes1(右上隅にある上部の Axes) をダブルクリックしてプロパティインスペクターに開きます。
- 2 UIContextMenu の右側に列をクリックして、ドロップダウンリストを参照します。
- 3 このリストから、plot_axes1 を選択します。

axes2 に同じ手順を実行しますが、plot_axes2 をその UIContextMenu として選択します。

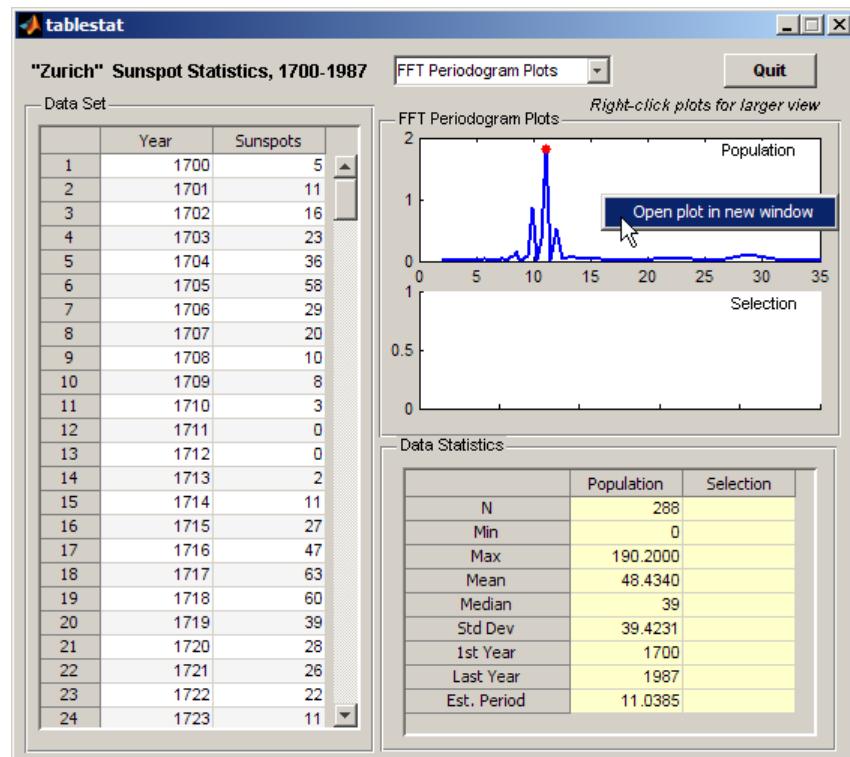
コンテキストメニューのコールバックのコード化。 2 つのコンテキストメニューの項目は同じ動作を行いますが、別々のオブジェクトを作成します。それぞれは、それ自身のコールバックをもちます。以下は、axes1 の plot_ax1_Callback コールバックです。

```
function plot_ax1_Callback(hObject, eventdata, handles)
% hObject    handle to plot_ax1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
% Displays contents of axes1 at larger size in a new figure

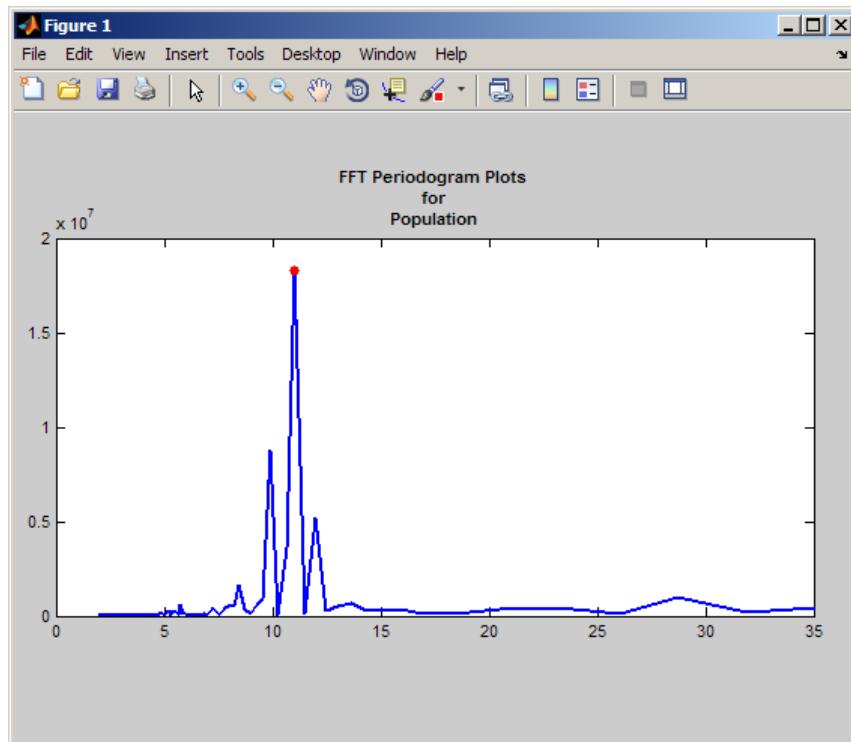
% Create a figure to receive this axes' data
axes1fig = figure;
% Copy the axes and size it to the figure
axes1copy = copyobj(handles.axes1, axes1fig);
set(axes1copy, 'Units', 'Normalized',...
    'Position', [.05,.20,.90,.60])
% Assemble a title for this new figure
str = [get(handles.uipanel3, 'Title') ' for ' ...
    get(handles.poplabel, 'String')];
title(str, 'Fontweight', 'bold')
% Save handles to new fig and axes in case
% we want to do anything else to them
handles.axes1fig = axes1fig;
handles.axes1copy = axes1copy;
guidata(hObject, handles);
```

他のコールバック、plot_ax2_Callback は plot_ax1_Callback と同じです。コード内の 1 のすべてのインスタンスが 2 で置き換えられることを除き、poplabel は sellabel で置き換えられます。Poplabel と Sellabel オブジェクトは、それぞれ axes1 と axes2 の、[Population] と [Selection] ラベルです。これらの文字列は、新規 Figure の axes1fig または axes2fig のプロットのタイトルを作成するために、uipanel3 に対する現在の Title に付加されます。

新規ウィンドウ機能でのプロットの利用. Axes の 1 つを右クリックして [Open plot in new window] を選択すると、Axes にグラフを含む新規の Figure が作成されます。コールバックは、Axes (ユーザーが [Data Set] 内のセルを選択するまで、axes2 は空です) 内にグラフが存在するかどうか、または以前に開いた Figure が同じグラフを含むかどうかを確認することができません。新規の Figure が常に作成されて、axes1 または axes2 のコンテンツがその中にコピーされます。たとえば、ユーザーは axes1 のピリオドグラムを右クリックして、[Open plot in new window] を選択します。



[Open plot in new window] をクリックすると、以下のコンテンツをもつ新規 Figure が表示されます。



新規ウィンドウが必要なくなった場合は、ユーザーが削除しなければなりません。これを行うためにコンテキストメニューはプログラムできます。それらのコールバックは、GUI の Axes それぞれに対して作成された最後の Figure のハンドルを保存するために guidata を呼び出すので、他のコールバックはいずれかの Figure を削除または再使用できます。たとえば、plot_ax1_Callback と plot_ax2_Callback コールバックは、handles.axes1copy または handles.axes2copy に格納された有効な Axes ハンドルに対する guidata をチェックすることができ、新規の Figure を作成せずに Axes を再使用します。

Tablestat の拡張

Tablestat の例の GUI は、適切な機能をもつように、いくつかの方法で拡張できます。

- ・ GUI が、MATLAB ワークスペースまたはデータファイルのデータ行列を読み込むことができる。そのために、以下を行います。
 - ファイル ダイアログ ボックスまたは入力ダイアログ ボックスを提供し、非数値、非行列データを取り除くことができるコードを与えます。
 - 列の既定の名前を付け、ユーザーが名前を変更する方法も提供します。
- ・ 解析とプロットのために、ユーザーがデータ列を選択することが可能になる。
 - 独立変数（通常、*x*）と従属変数（通常、*y*）として、どの列を使用するかをユーザーが示すための方法。
 - データのサブセットを特定するために Tablestat がすでにセルの選択を使用するので、どの列を処理するかを特定する uicontrol またはメニュー。
- ・ ユーザーが Axes の 1 つをダブルクリックするときに、新規の Figure ウィンドウにプロットを開くように GUI をプログラムします。（これを行うために、コンテキストメニューを利用する代わりに、またはコンテキストメニューを利用した上でさらに）。これは、以下のことを含みます。
 - 各 Axes に対して、Figure の現在の SelectionType プロパティを取得する ButtonDownFcn を与え、1 回または 2 回クリックされたかを判定します。

ヒント コールバックで `get(gcbf, 'SelectionType')` を使用し、「open」の値をチェックします。

- axes1 と axes2 の NextPlot プロパティを、ReplaceChildren に設定すると、グラフが Axes にプロットされる度に ButtonDownFcn のハンドルを Axes から削除されることを防ぎます。これは、NextPlot が既定の Add であるときに常に起こります。
- コンテキストメニューのコールバックが現在行うように、新規の Figure と Axes を作成し、クリックされた Axes のコンテンツをそれにコピーします。

リスト ボックス ディレクトリ リーダー

このセクションの内容…

“リスト ボックス ディレクトリの例について” (p.10-54)

“リスト ボックス ディレクトリの GUI の表示と実行” (p.10-55)

“リスト ボックス ディレクトリ GUI の実現” (p.10-56)

リスト ボックス ディレクトリの例について

この例では、フォルダー内のファイルを表示するために、リスト ボックスを使用します。リスト項目をダブルクリックするとき、次のいずれかが起こります。

- 項目がファイルである場合、GUI はファイルのタイプに従って適当なファイルを開きます。
- 項目がフォルダーである場合、GUI はそのフォルダーの内容をリスト ボックスに読み込みます。
- 項目がシングル ドット(.) である場合、GUI は現在のフォルダーの表示を更新します。
- 項目が 2 つのドット(..) である場合、GUI は 1 つ上にある親フォルダーに変更し、親フォルダーの内容をもつリスト ボックスを置きます。

次の図は、この GUI を説明します。



リスト ボックス ディレクトリの GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルと FIG-ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下の手順を行います。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に FIG-ファイルを開くには、`guide lbox2` と入力するか、または ここをクリックします。
- 3 エディター内に M ファイルを開くには、`edit lbox2` と入力するか、または ここをクリックします。

コンポーネントのプロパティ インスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を修正してから、GUIDE から [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存することができます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUIDE で GUI を調べて実行する場合、代わりに以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 lbox2 GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

リスト ボックス ディレクトリ GUI の実現

次の節では、実現を記述します。

- ・ “ディレクトリの指定”(p.10-56) – GUI が起動するとき、入力引数として、フォルダー パスを渡す方法を示します。
- ・ “リスト ボックスの読み込み”(p.10-58) – フォルダーの内容をリスト ボックスに読み込むサブ関数について説明します。このサブ関数は、また、handles 構造体に、フォルダーの内容についての情報を保存することもできます。
- ・ “リスト ボックス コールバック”(p.10-59) – リスト ボックス内の項目をダブルクリックしたときに、応答するようにリスト ボックスをプログラムする方法について説明します。

ディレクトリの指定

引数として、文字列 'create' を渡すか、フォルダーへの完全なパスを含む文字列を渡すことにより、GUI が最初に開く際にリストするフォルダーを指定することができます。構文は、`list_box('create', 'dir_path')` です。フォルダーを指定しない場合(すなわち、入力引数をもたない GUI M ファイルを呼び出す場合)、GUI は MATLAB の現在のフォルダーを使用します。

GUIDE が生成する GUI M ファイルの既定の動作は、入力引数がない場合に GUI を起動するか、あるいは第 1 入力引数が文字列である場合にサブ関数を呼び出すことです。この例では、この動作を変更して、次の場合に M ファイルを呼び出すことができます。

- ・ 入力引数がない場合 – MATLAB の現在のフォルダーを使用して GUI を起動します。
- ・ 第 1 入力引数が 'dir' であり、第 2 入力引数がフォルダーへの正しいパスを指定する文字列である場合 – 指定したフォルダーを表示して、GUI を実行します。
- ・ 第 1 入力引数がフォルダーではなく文字列で、2 つ以上の引数がある場合 – 引数によって同定されるサブ関数を実行します(コールバックを実行します)。

以下のコードリストは、次のいずれかを行う、GUI M ファイルの初期化部分の全体を示します。

- ・ フォルダーが指定されていない場合、リスト ボックス フォルダーを現在のフォルダーに設定します。
- ・ フォルダーが指定されている場合、現在のフォルダーを変更します。

```
function lbox2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to untitled (see VARARGIN)

% Choose default command line output for lbox2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

if nargin == 3,
    initial_dir = pwd;
elseif nargin > 4
    if strcmpi(varargin{1}, ' dir' )
        if exist(varargin{2}, ' dir' )
            initial_dir = varargin{2};
        else
            errordlg([' Input argument must be a valid' ...
                      ' folder' ], ' Input Argument Error!' )
            return
        end
    else
        errordlg(' Unrecognized input argument' ,...
                  ' Input Argument Error!' );
        return;
    end
end
% Populate the listbox
load_listbox(initial_dir, handles)
```

リスト ボックスの読み込み

この例では、リスト ボックスに項目を読み込むために、サブ関数を使用します。このサブ関数は、入力引数として、フォルダーへのパスと handles 構造体を受け取り、以下の手順を実行します。

- ・ GUI が必要に応じて、ツリー構造を上下に進むことができるよう、指定されたフォルダーに変更します。
- ・ 指定されたフォルダーのファイルのリストを得て、どれがフォルダ名、あるいはファイル名であるかを決定するために、dir コマンドを使用します。dir は、2 つのフィールド（前文のような情報を含む name および isdir）をもつ構造体 (dir_struct) を返します。
- ・ ファイル名とフォルダ名を並べ替え (sortrows)、この情報が、他の関数に渡されるように handles 構造体に、並べ替えられた名前と他の情報を保存してください。
name 構造体のフィールドは sortrows にセル配列として渡されます。これは、各行に 1 つのファイル名を得るように置き換えられます。isdir フィールドと並べ替えられたインデックスの値 sorted_index は handles 構造体にベクトルとして保存されます。
- ・ handles 構造体を保存するために、guidata を呼び出します。
- ・ ファイル名とフォルダ名を表示するためにリスト ボックスの String プロパティを設定し、Value プロパティを 1 に設定します。このため、Value は String の項目数を超えないことが保証されます。これは、MATLAB は Value プロパティを、選択が行われたときに限り更新し、String の内容が変更したときには更新しないためです。
- ・ コマンド pwd の出力に、String プロパティを設定することにより、テキスト ボックスの現在のフォルダーを表示します。

関数 load_listbox は、リスト ボックスのコールバックにより呼び出されますが、GUIDE M ファイルの opening 関数でも呼び出されます。

```
function load_listbox(dir_path, handles)
cd (dir_path)
dir_struct = dir(dir_path);
[sorted_names, sorted_index] = sortrows({dir_struct.name}' );
handles.file_names = sorted_names;
handles.is_dir = [dir_struct.isdir];
handles.sorted_index = sorted_index;
guidata(handles.figure1, handles)
set(handles.listbox1, 'String', handles.file_names, ...
    'Value', 1)
```

```
set(handles.text1, 'String', pwd)
```

リスト ボックス コールバック

リスト ボックス コールバックは、項目のダブルクリックのみを取り扱います。ダブルクリックは、リスト ボックスからファイルを開く標準の方法です。選択される項目がファイルである場合は、`open` コマンドに渡されます。フォルダーの場合は、GUI はそのフォルダーに変更し、その内容をリストします。

ファイル タイプを開く方法の定義. `open` コマンドは、異なるファイルの種類の数を取り扱うことができます。ただし、コールバックは、FIG-ファイルを別に取り扱います。FIG-ファイルをスタンドアロンの `Figure` として開く代わりに、編集のために `guide` を用いて開きます。

ユーザーが選択した項目の決定. 項目を一度クリックすると、リスト ボックスのコールバックも呼び出されるので、ユーザーがいつダブルクリックしたかを判別するため `Figure` の `SelectionType` プロパティにたずねる必要があります。項目をダブルクリックすると、`SelectionType` プロパティが `open` に設定されます。

リスト ボックスのすべての項目は、1 から `n` までのインデックスで参照されます。`1` は 1 つめの項目を参照し、`n` は `n` 番目の項目のインデックスです。MATLAB は、リスト ボックスの `Value` プロパティに、このインデックスを保存します。

コールバックはこのインデックスを使用して、`String` プロパティに含まれる項目のリストから選択される項目名を取得します。

選択項目がファイルであるか、ディレクトリであるかの決定. 関数 `load_listbox` は、項目が、ファイルであるか、あるいはフォルダーであるかを示す値のリストを得るために、`dir` コマンドを使用します。これらの値（フォルダーに対して 1、ファイルに対して 0）は、`handles` 構造体に保存されています。リスト ボックス コールバックは、現在の選択が、ファイルあるいはフォルダーであるかを決定するために、これらの値をたずね、次の動作を起こします。

- ・ フォルダーが選択される場合 - フォルダーに移動 (`cd`) して、新規フォルダーの内容をもつリスト ボックスを置くために、再び、`load_listbox` を呼び出します。
- ・ ファイルが選択される場合 - FIG-ファイルであるかどうかを決定するために、ファイル拡張子 (`fileparts`) を取得してください。FIG-ファイルは、`guide` とともに開きます。その他のタイプのファイルはすべて、`open` に渡されます。

open ステートメントは、コマンド ラインに戻る代わりに、エラー ダイアログ ボックス (errordlg)において、エラーを捉るために、try/catch ブロック内で呼び出されます。

```

get(handles.figure1,'SelectionType');
% If double click
if strcmp(get(handles.figure1,'SelectionType'),'open')
    index_selected = get(handles.listbox1,'Value');
    file_list = get(handles.listbox1,'String');
    % Item selected in list box
    filename = file_list{index_selected};
    % If folder
    if handles.is_dir(handles.sorted_index(index_selected))
        cd (filename)
        % Load list box with new folder.
        load_listbox(pwd,handles)
    else
        [path,name,ext,ver] = fileparts(filename);
        switch ext
            case '.fig'
                % Open FIG-file with guide command.
                guide (filename)
            otherwise
                try
                    % Use open for other file types.
                    open(filename)
                catch ex
                    errordlg',...
                        ex.getReport('basic'),'File Type Error','modal')
                end
            end
        end
    end
end

```

ファイルのタイプが未知のファイルを開く。3 文字の拡張子をもつ任意のファイルをインクルードするために、open コマンドが認識するファイルのタイプを拡張することができます。openxyz という名前をもつ M ファイルを作成してこれを行います。xyz はファイル拡張子です。ただし、openfig.m は GUI を開くのに必要な MATLAB 関数であるので、この方法で FIG-ファイルを開かないでください。詳細は、open と openfig を参照してください。

リスト ボックスからワークスペース変数にアクセス

このセクションの内容…

“ワークスペース変数の例について” (p.10-61)

“ワークスペース変数の GUI の表示と実行” (p.10-62)

“ワークスペース変数の読み込み” (p.10-63)

“リスト ボックスからの選択項目の読み込み” (p.10-64)

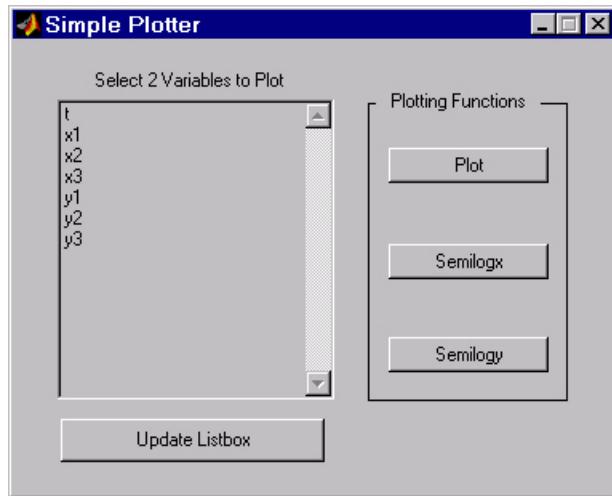
ワークスペース変数の例について

この GUI はリスト ボックスを使用して、ベース ワークスペースにプロット変数名を表示します。はじめに、リスト ボックスでは変数名が選択されません。GUI は、以下についてコントロールします。

- ・ リストを更新します。
- ・ リスト ボックスの複数の変数を選択します。厳密に 2 つの変数が選択されなければなりません。
- ・ 選択された変数の、線形で `semilogx` と `semilogy` の線グラフを作成します。

GUI は、ベース ワークスペースのプロット コマンドを評価します。GUI は、プロットの前に検証しません。ユーザーが、変数の組を選択し、一方の変数を他方の変数に対してプロットします。最も上の選択は、 x 変数として使用され、それよりも下の選択は y 変数として使用されます。

次の図は、GUI のレイアウトを説明します。



ワークスペース変数の GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルと FIG-ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下の手順を行います。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に FIG-ファイルを開くには、`guide lb` と入力するか、または ここをクリックします。
- 3 エディター内に M ファイルを開くには、`edit lb` と入力するか、または ここをクリックします。

コンポーネントのプロパティ インスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を修正してから、GUIDE から [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存すること

ができます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUIDE で GUI を調べて実行する場合、代わりに以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 lb GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

ワークスペース変数の読み込み

GUI を初期化するとき、ワークスペース変数を問合せ、リスト ボックスの String プロパティがこれらの変数名を表示するように設定する必要があります。GUI M ファイルに追加される次のサブ関数は、ベース ワークスペースで、コマンド who を実行するために、evalin を使用して、これを行います。who コマンドは、文字列のセル配列を返します。この文字列のセル配列が、リスト ボックスを配置するために使用されます。

```
function update_listbox(handles)
vars = evalin('base','who');
set(handles.listbox1,'String',vars)
```

関数の入力引数は、GUI M ファイルにより生成される handles 構造体です。この構造体は、GUI におけるその他すべてのコンポーネントのハンドルの他、リスト ボックスのハンドルを含みます。

Update Listbox プッシュ ボタンに対するコールバックは、また、`update_listbox` を呼び出します。

リスト ボックスからの選択項目の読み込み

GUI は、ユーザーが、ワークスペースから 2 変数を選択し、その後、グラフを生成する 3 つのプロットコマンド、`plot`、`semilogx`、`semilogy` の 1 つを選択するよう要求します。

GUI M ファイルには、リスト ボックスに対するコールバックは存在しません。プロットの動作はプッシュ ボタンにより開始されるので、リスト ボックスに対するコールバックは必要ありません。

複数選択を可能にする

リスト ボックスにおいて複数選択を可能とするために、 $\text{Max} - \text{Min} > 1$ であるように、 Min および Max プロパティを設定する必要があります。このため、この条件を満たすように、既定値の Min 値 0 および Max 値 1 を変更する必要があります。プロパティ インスペクターを使用して、これらのプロパティをリスト ボックスに設定します。

ユーザーが複数項目を選択する方法

リスト ボックスの複数の選択は、多くのシステムに対する標準に従います。

- Ctrl を押しながら左マウス ボタンをクリック – 隣接しない複数項目の選択
- Shift を押しながら左マウスボタンをクリック – 隣接する複数項目の選択

ユーザーは、プロットの生成に必要な 2 つの変数を選択するために、これらの手法のいずれかを使用する必要があります。

プロット関数に対して、変数名を返す

`get_var_names` サブ関数は、ユーザーが 3 つのプロットボタンの 1 つをクリックするとき、選択されている 2 つの変数名を返します。この関数は、次のことを行います。

- String プロパティからリスト ボックスのすべての項目のリストを取得する
- Value プロパティから選択項目のインデックスを取得する
- 2 つの項目が選択される場合、2 つの文字列変数を返すそうでない場合、`get_var_names` は、2 つの変数を選択する必要があることを伝えるエラー ダイアログ ボックスを表示します。

次は、`get_var_names` に対するコードを示します。

```
function [var1,var2] = get_var_names(handles)
list_entries = get(handles.listbox1,'String');
index_selected = get(handles.listbox1,'Value');
if length(index_selected) ~= 2
    errordlg(' You must select two variables' ,...
        ' Incorrect Selection' , 'modal')
else
    var1 = list_entries{index_selected(1)};
    var2 = list_entries{index_selected(2)};
end
```

プロット ボタンのコールバック

プロット ボタンのコールバックは、プロットする変数名を取得するために、`get_var_names` を呼び出し、その後、ベース ワークスペース内で `plot` コマンドを実行するために、`evalin` を呼び出します。

たとえば、以下は関数 `plot` に対するコールバックです。

```
function plot_button_Callback(hObject, eventdata, handles)
[x,y] = get_var_names(handles);
evalin('base',[ ' plot(' x ', ' y ')' ])
```

評価するコマンドは、文字列と変数を連結することで作成されます。次のようになります。

```
try
    evalin('base',[ ' semilogx(' ,x,' , ' ,y,' )' ])
catch ex
    errordlg(..., ...
        ex.getReport('basic'), 'Error generating semilogx plot', 'modal')
end
```

`try/catch` ブロックは、不適切なデータをグラフにしようとした結果発生したエラーを処理します。評価されると、コマンドの結果は次のようになります。

```
plot(x,y)
```

他の 2 つのプロット ボタンは同じように機能し、`semilogx(x,y)` や `semilogy(x,y)` になります。

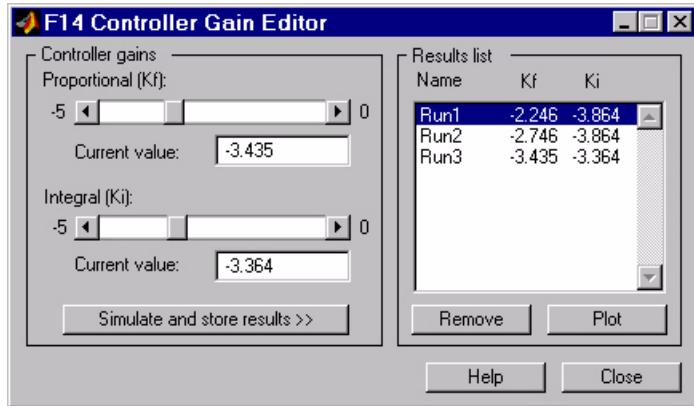
Simulink モデル パラメーターを設定する GUI

このセクションの内容…

- “Simulink モデル パラメーターの例について” (p.10-66)
- “Simulink パラメーター GUI の表示と実行” (p.10-67)
- “Simulink パラメーター GUI の利用法” (p.10-69)
- “GUI の起動” (p.10-70)
- “スライダーおよびエディット テキスト コンポーネントのプログラミング” (p.10-71)
- “GUI からのシミュレーションの実行” (p.10-73)
- “リスト ボックスから結果を削除する” (p.10-75)
- “結果データのプロット” (p.10-76)
- “GUI の [ヘルプ] ボタン” (p.10-78)
- “GUI のクローズ” (p.10-78)
- “リスト ボックス コールバックと作成関数” (p.10-79)

Simulink モデル パラメーターの例について

この例は、Simulink モデルのパラメーターを設定する GUI を作成する方法について説明します。この GUI は、さらに、シミュレーションを実行し、結果を Figure ウィンドウにプロットすることができます。次の図は、コントロール ゲインに異なる値をもつ 3 つのシミュレーションの実行後の GUI を示します。



この例では、多くの GUI 構築の手法を説明します。

- ・ GUI から、Simulink モデルを開いて、パラメーターを設定します。
- ・ テキスト ボックスと連動して機能するスライダーを実現します。ユーザー入力を受け取り、現在の値を表示します。
- ・ GUI の状態に依存して、使用可能あるいは使用不可能にします。
- ・ handles 構造体を使用して、様々なグローバルデータを管理します。
- ・ 隠蔽されたハンドルをもつ Figure にグラフィックス出力をします。
- ・ MATLAB ヘルプ ブラウザーの .html ファイルを表示する [ヘルプ] ボタンを追加します。

Simulink パラメーター GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルと FIG-ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下の手順を行います。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。

2 GUIDE に FIG-ファイルを開くには、guide f14ex と入力するか、またはここをクリックします。

3 エディター内に M ファイルを開くには、edit f14ex と入力するか、またはここをクリックします。

コンポーネントのプロパティインスペクターを開くには、レイアウトエディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を修正してから、GUIDE から [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存することができます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUIDE で GUI を調べて実行する場合、代わりに以下の手順に従います。

1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)

2 f14ex の GUI を実行するには、ここをクリックします。

3 GUIDE レイアウトエディターに GUI を表示するには、ここをクリックします。(読み取り専用)

4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

Simulink パラメーター GUI の利用法

メモ この GUI を実行するには、Simulink をインストールしている必要があります。GUI をはじめて実行するとき、Simulink が開き (Simulink がまだ実行していない場合)、f14 デモ モデルを読み込みます。これには、数秒かかることがあります。

GUI には [ヘルプ] ボタンがあります。[ヘルプ] ボタンをクリックすると、ヘルプ ブラウザに HTML ファイル f14ex_help.html が開きます。このファイルは、GUI FIG-ファイルや M ファイルとともに examples フォルダーにありますが、ヘルプ テキストをもつ以下の 5 つのセクションを含みます。

F14 コントローラー ゲイン エディター

比例-積分コントローラーで使用されるゲインを変更すると、飛行機の攻撃角度とパイロットが感じる重力加速度にどのような影響があるかを解析するために、F14 コントローラー ゲイン エディターを使用することができます。

この GUI を実行するために、Simulink ダイアグラム f14.mdl を開く必要があることに注意してください。F14 Simulink モデルを閉じると、モデルを実行する必要があるときはいつでも、GUI がモデルを再び開きます。

コントローラーのゲインの変更

2 つのブロックにおけるゲインを変更することができます。

1 Gain ブロックにおける比例ゲイン (K_f)

2 Transfer Function ブロックにおける積分ゲイン (K_i)

2 つの方法のいずれかで、どちらのゲインも変更することができます。

1 そのゲインに関連するスライダーを移動してください。

2 そのゲインに関連する Current value エディット フィールドに、新規の値を入力してください。

ブロックの値は、GUI に新しい値を入力すると直ちに更新されます。

シミュレーションの実行

一度、ゲインの値を設定すると、[Simulate and store results] ボタンをクリックすることによってシミュレーションを実行することができます。シミュレーション時間と出力ベクトルは、Results list に保存されます。

結果のプロット

プロットしたい Results list 内の結果 (Run1、Run2 など) の行を選択したり、[プロット] ボタンをクリックすることにより、1 つあるいは複数のシミュレーションのプロットを作成することができます。複数の行を選択する場合、グラフは各結果のプロットを含みます。

グラフは、Figure 内に表示され、[プロット] ボタンをクリックするたびにクリアされます。GUI のみが、このウィンドウ内にグラフを表示できるように Figure のハンドルは、隠されます。

結果の除去

Results list から結果を取り除くには、取り除きたい 1 行あるいは複数行を選択し、[削除] ボタンをクリックしてください。

GUI の起動

GUI は、解析ツールとして使用されるように設計されているので、ブロックされてなく、モーダルでもありません。

GUI オプションの設定

この GUI は、次の GUI オプション設定を使用します。

- ・ サイズ変更動作: サイズ変更不可
- ・ コマンド ラインへのアクセス: オフ
- ・ M ファイルのオプションが選択されます。
 - コールバック関数のプロトタイプを作成
 - GUI は 1 つのインスタンスの実行のみ可能とします。

Simulink ブロック線図を開く

この例は、Simulink モデル f14 と共に動作するように設計されています。GUI は、パラメーターを設定し、シミュレーションを実行するので、GUI が表示されるとき、f14 モ

モデルは開かれなければなりません。GUI M ファイルが、GUI を起動する場合、これは、model_open サブ関数を実行します。サブ関数は、以下を行います。

- ・ モデルが開かれているかを決定します (find_system)。
- ・ モデルとサブシステムに対して、ブロック線図を開き、まだ、開いていない場合、パラメーターが設定されます (open_system)。
- ・ ゲイン値 (set_param) を表示することができるよう、コントローラー Gain ブロックのサイズを変更します。
- ・ Simulink の図 (Figure) の手前に表示されるように GUI を最前面にもってきます。
- ・ GUI の現在の設定に一致するようにブロック パラメーターを設定します。

次は、model_open サブ関数に対するコードです。

```
function model_open(handles)
if isempty(find_system(' Name' , ' f14' )),
    open_system(' f14'); open_system(' f14/Controller' )
    set_param(' f14/Controller/Gain' , ' Position' ,[275 14 340 56])
    figure(handles.F14ControllerEditor)
    set_param(' f14/Controller Gain' , ' Gain' ,...
        get(handles.KfCurrentValue,' String' ))
    set_param(... , ...
        ' f14/Controller/Proportional plus integral compensator' ,...
        ' Numerator' ,...
        get(handles.KiCurrentValue,' String' ))
end
```

スライダーおよびエディット テキスト コンポーネントのプログラミング

GUIにおいて、各スライダーはエディット テキスト コンポーネントと連動しているため、次のことが起こります。

- ・ エディット テキストは、スライダーの現在の値を表示します。
- ・ ユーザーは値をエディット テキスト ボックスに入力し、スライダーがその値を更新するようにします。
- ・ 両方のコンポーネントは、ユーザーによりアクティブにされるとき、適当なモデル パラメーターを更新します。

スライダー コールバック

この GUI は、2 つのスライダーを使用して、ブロックのゲインを指定します。これら 2 つのコンポーネントでは、指定した範囲内の連続値を選択することができます。スライダーの値を変更すると、コールバックは次の手順を実行します。

- ・ シミュレーション パラメーターが設定されるように、Simulink® モデルが開いていることを確かめるために、model_open を呼び出します。
- ・ 新規のスライダー値を取得します。
- ・ スライダーに合うよう、現在の値のエディット テキスト コンポーネントの値を設定します。
- ・ 適当なブロック パラメーターを新規の値に設定します (set_param)。

以下は、Proportional (Kf) スライダーのコールバックです。

```
function KfValueSlider_Callback(hObject, eventdata, handles)
% Ensure model is open.
model_open(handles)
% Get the new value for the Kf Gain from the slider.
NewVal = get(hObject, 'Value');
% Set the value of the KfCurrentValue to the new value
% set by slider.
set(handles.KfCurrentValue,'String',NewVal)
% Set the Gain parameter of the Kf Gain Block to the new value.
set_param('f14/Controller/Gain','Gain',num2str(NewVal))
```

スライダーは数値を返し、エディット テキストは文字列を必要としますが、uicontrol は値を正しいタイプに自動的に変換します。

Integral (Ki) スライダーのコールバックは、Proportional (Kf) スライダーのコールバックと似たアプローチに従います。

現在の値のエディット テキスト コールバック

エディット テキスト ボックスでは、それぞれのパラメーター値を入力できます。テキスト ボックスにデータを入力後、GUI の他のコンポーネントをクリックするとき、エディット テキスト コールバックは、次の手順を実行します。

- ・ Simulink モデルが、シミュレーション パラメーターを設定することができるよう、Simulink モデルが開いていることを確かめるために、model_open を呼び出します。

- エディット ボックスの String プロパティにより返される文字列を倍精度値に変換します (str2double)。
- ユーザーにより入力された値が、スライダーの範囲内であるかどうかをチェックします。
値が範囲外にある場合、エディット テキスト String プロパティは、(ユーザーにより、入力される値を受け付けないで) スライダーの値に設定されます。
値が範囲内にある場合、スライダー Value プロパティは、新規の値に更新されます。
- 適当なブロック パラメーターを新規の値に設定します (set_param)。

これは、Kf Current value のテキスト ボックスに対するコールバックです。

```
function KfCurrentValue_Callback(hObject, eventdata, handles)
% Ensure model is open.
model_open(handles)
% Get the new value for the Kf Gain.
NewStrVal = get(hObject, 'String');
NewVal = str2double(NewStrVal);
% Check that the entered value falls within the allowable range.
if isempty(NewVal) || (NewVal<-5) || (NewVal>0),
    % Revert to last value, as indicated by KfValueSlider.
    OldVal = get(handles.KfValueSlider,'Value');
    set(hObject, 'String',OldVal)
else % Use new Kf value
    % Set the value of the KfValueSlider to the new value.
    set(handles.KfValueSlider,'Value',NewVal)
    % Set the Gain parameter of the Kf Gain Block
    % to the new value.
    set_param('f14/Controller/Gain','Gain',NewStrVal)
end
```

Ki Current value に対するコールバックは、同様のアプローチに従います。

GUI からのシミュレーションの実行

GUI の [Simulate and store results] ボタンのコールバックは、モデル シミュレーションを実行し、結果を handles 構造体に格納します。handles 構造体にデータを

格納すると、この構造体を引数として渡すことができるので、データを他のサブ関数に渡すプロセスが簡単になります。

[Simulate and store results] ボタンをクリックする場合、コールバックは、次の手順を実行します。

- ・ `sim` を呼び出します。これはシミュレーションを実行し、プロットのために使用されるデータを返します。
- ・ シミュレーションの結果、GUI により設定されるシミュレーション パラメーターの現在の値、実行名と数を保存するために、構造体を生成します。
- ・ `handles` 構造体に、その構造体を格納します。
- ・ 一番最近の実行をリストするために、リスト ボックス `String` を更新します。

以下は、[Simulate and store results] ボタンのコールバックです。

```
function SimulateButton_Callback(hObject, eventdata, handles)
[timeVector,stateVector,outputVector] = sim('f14');
% Retrieve old results data structure
if isfield(handles,'ResultsData') &
~isempty(handles.ResultsData)
    ResultsData = handles.ResultsData;
    % Determine the maximum run number currently used.
    maxNum = ResultsData(length(ResultsData)).RunNumber;
    ResultNum = maxNum+1;
else % Set up the results data structure
    ResultsData = struct('RunName',[],'RunNumber',[],...
        'KiValue',[],'KfValue',[],'timeVector',[],...
        'outputVector',[]);
    ResultNum = 1;
end
if isequal(ResultNum,1),
    % Enable the Plot and Remove buttons
    set([handles.RemoveButton,handles.PlotButton], 'Enable','on')
end
% Get Ki and Kf values to store with the data and put in the
% results list.
Ki = get(handles.KiValueSlider,'Value');
Kf = get(handles.KfValueSlider,'Value');
ResultsData.ResultNum.RunName = ['Run',num2str(ResultNum)];
```

```

ResultsData(ResultNum).RunNumber = ResultNum;
ResultsData(ResultNum).KiValue = Ki;
ResultsData(ResultNum).KfValue = Kf;
ResultsData(ResultNum).timeVector = timeVector;
ResultsData(ResultNum).outputVector = outputVector;
% Build the new results list string for the listbox
ResultsStr = get(handles.ResultsList,'String');
if isequal(ResultNum,1)
    ResultsStr = {[ ' Run1' , num2str(Kf), ' ', num2str(Ki)]};
else
    ResultsStr = [ResultsStr;...
        {[ ' Run' , num2str(ResultNum), ' ', num2str(Kf), ' ', ...
        num2str(Ki)]}];
end
set(handles.ResultsList,'String',ResultsStr);
% Store the new ResultsData
handles.ResultsData = ResultsData;
guidata(hObject, handles)

```

リスト ボックスから結果を削除する

GUI の [Remove] ボタンのコールバックは、Results list リスト ボックスから選択される任意の項目を削除します。これは、また、handles 構造体からの相当する実行データも削除します。[Remove] ボタンをクリックすると、コールバックは、次の手順を実行します。

- ・ [Remove] ボタンをクリックすると、リスト ボックスのどの項目が選択されているかを決定し、各項目を空の行列 [] に設定することにより、リスト ボックスの String プロパティからこれらの項目を取り除きます。
- ・ handles 構造体から削除されたデータを取り除きます。
- ・ 文字列 <empty> を表示し、リスト ボックス内のすべての項目が取り除かれる場合、(Enable プロパティ ボタンを使用して) [Remove] および [Plot] ボタンを使用不可とします。
- ・ handles 構造体に変更を保存します (guidata)。

[Remove] ボタンのコールバックです。

```

function RemoveButton_Callback(hObject, eventdata, handles)
currentVal = get(handles.ResultsList,'Value');

```

```

resultsStr = get(handles.ResultsList, 'String');
numResults = size(resultsStr, 1);
% Remove the data and list entry for the selected value
resultsStr(currentVal) =[];
handles.ResultsData(currentVal)=[];
% If there are no other entries, disable the Remove and Plot
button
% and change the list string to <empty>
if isequal(numResults, length(currentVal)),
    resultsStr = {'<empty>'};
    currentVal = 1;

set([handles.RemoveButton, handles.PlotButton], 'Enable', 'off')
end
% Ensure that list box Value is valid, then reset Value and String
currentVal = min(currentVal, size(resultsStr, 1));
set(handles.ResultsList, 'Value', currentVal, 'String', resultsStr)
% Store the new ResultsData
guidata(hObject, handles)

```

結果データのプロット

GUI の [Plot] ボタンのコールバックは、実行データのプロットを生成し、凡例を付けます。プロットするデータは、handles 構造体のコールバックに渡されます。これはまた、シミュレーションの実行時に使用されるゲインの設定を含みます。[Remove] ボタンをクリックするとき、コールバックは、次の手順を実行します。

- ・ 2 変数（時間ベクトルと出力ベクトル）と、各プロットのための実行結果に対するカラーを含む、Results list において選択される各実行のデータを集めます。
- ・ 格納されているデータから、凡例のための文字列を作成します。
- ・ プロットのための Figure と座標軸を生成し、[Close] ボタンのコールバックにより、使用のためにハンドルを保存します。
- ・ データのプロット、凡例の付加、Figure の可視化を行います。

隠蔽された Figure へのプロット

プロットを含む Figure は、非表示として作成され、プロットや凡例を追加後に表示されます。この Figure が、コマンドラインで実行されるプロットのコマンドあるいはその他の GUI のターゲットとなることを防ぐには、この HandleVisibility および

`IntegerHandle` プロパティを `off` に設定します。これは、`Figure` が `plot` および `legend` コマンドからも隠蔽されていることを意味します。

隠蔽された `Figure` にプロットするために、次の手順を使用してください。

- ・ 作成したとき、`Figure` のハンドルを保存します。
- ・ 座標軸を生成し、その `Parent` プロパティに `Figure` ハンドルを設定し、座標軸のハンドルを保存します。
- ・ (1つあるいは複数の `Line` オブジェクトである) プロットを生成し、これらのラインのハンドルを保存し、座標軸のハンドルをその `Parent` プロパティに設定します。
- ・ `Figure` を表示します。

[Plot] ボタンのコールバックのリスト

[Plot] ボタンのコールバックです。

```
function PlotButton_Callback(hObject, eventdata, handles)
currentVal = get(handles.ResultsList, 'Value');
% Get data to plot and generate command string with color
% specified
legendStr = cell(length(currentVal),1);
plotColor = {'b', 'g', 'r', 'c', 'm', 'y', 'k'};
for ctVal = 1:length(currentVal);
    PlotData{(ctVal*3)-2} =
    handles.ResultsData(currentVal(ctVal)).timeVector;
    PlotData{(ctVal*3)-1} =
    handles.ResultsData(currentVal(ctVal)).outputVector;
    numColor = ctVal - 7*( floor((ctVal-1)/7) );
    PlotData{ctVal*3} = plotColor{numColor};
    legendStr{ctVal} = ...
        [handles.ResultsData(currentVal(ctVal)).RunName, ' : Kf=' ,...
        num2str(handles.ResultsData(currentVal(ctVal)).KfValue),...
        ' ; Ki=' , ...
        num2str(handles.ResultsData(currentVal(ctVal)).KiValue)];
end
% If necessary, create the plot figure and store in handles
% structure
if ~isfield(handles,'PlotFigure') ||...
    ~ishandle(handles.PlotFigure),
```

```

handles.PlotFigure = ...
    figure(' Name' , ' F14 Simulation Output' ,...
    ' Visible' , ' off' , ' NumberTitle' , ' off' ,...
    ' HandleVisibility' , ' off' , ' IntegerHandle' , ' off' );
handles.PlotAxes = axes(' Parent' , handles.PlotFigure);
guidata(hObject, handles)
end
% Plot data
pHandles = plot(PlotData{:,} , ' Parent' , handles.PlotAxes);
% Add a legend, and bring figure to the front
legend(pHandles(1:2:end), legendStr{:,})
% Make the figure visible and bring it forward
figure(handles.PlotFigure)

```

GUI の [ヘルプ] ボタン

GUI の [Help] ボタンのコールバックは、MATLAB ヘルプ ブラウザー内の HTML ファイルを表示します。これは、2 つのコマンドを使用します。

- which コマンドは、ファイルが MATLAB パス上にあるとき、ファイルの完全パス名を返します。
- web コマンドは、ヘルプ ブラウザー内のファイルを表示します。

これは、[Help] ボタンのコールバックです。

```

function HelpButton_Callback(hObject, eventdata, handles)
HelpPath = which(' f14ex_help.html' );
web(HelpPath);

```

Web ブラウザーにヘルプ ドキュメンテーションを表示し、外部の URL を読み込むこともできます。これらのオプションの説明は、web の説明を参照してください。

GUI のクローズ

GUI [Close] ボタンのコールバックは、プロット Figure が存在する場合、プロット Figure を閉じ、それから GUI を閉じます。プロット Figure と GUI の Figure のハンドルは、handles 構造体から利用できます。コールバックは、2 つの手順を実行します。

- handles 構造体に PlotFigure フィールドがあるか、このフィールドが（ユーザーが手動で閉じた Figure の）正しい Figure ハンドルを含むかどうかをみるために、チェックします。
- GUI の Figure を閉じます。

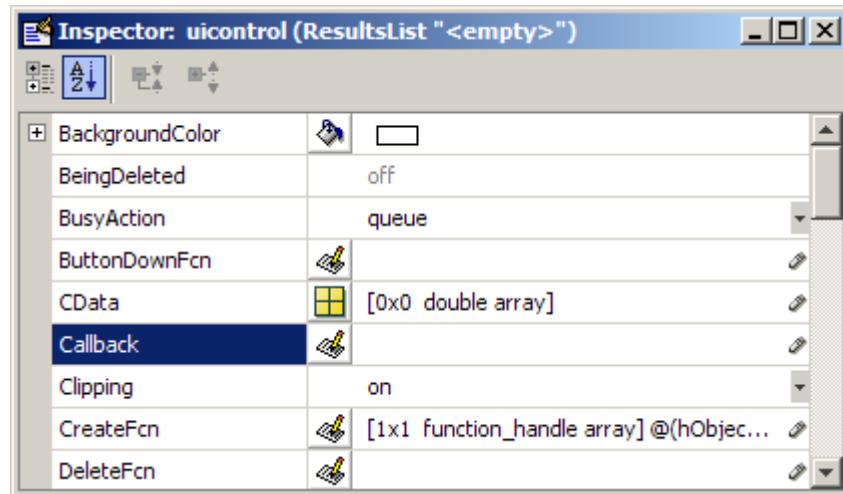
以下は、[Close] ボタンのコールバックです。

```
function CloseButton_Callback(hObject, eventdata, handles)
% Close the GUI and any plot window that is open
if isfield(handles,'PlotFigure') && ...
    ishandle(handles.PlotFigure),
    close(handles.PlotFigure);
end
close(handles.F14ControllerEditor);
```

リスト ボックス コールバックと作成関数

リスト ボックスの項目上で実行される動作は、プッシュ ボタン ([Simulate and store results]、[Remove]、および [Plot]) により実行されるため、この GUI はリスト ボックス コールバックを使用しません。GUIDE は、リスト ボックスを追加するとき、自動的に、コールバックのタブを挿入し、コールバックがトリガーされるとき（これは、ユーザーがリスト ボックスの項目を選択するときに起こります）はいつでもこのサブ関数を実行するように、Callback プロパティを自動的に設定します。

この例では、リスト ボックスのコールバックを実行する必要はありません。ソフトウェアがコールバックの実行を試みないように、このコールバックを GUI M ファイルから削除したり、さらにプロパティ インスペクターで Callback プロパティの文字列を削除することもできます。



リスト ボックスのコールバックをトリガーする方法の詳細は、リスト ボックス の説明を参照してください。

背景を白に設定

リスト ボックス作成関数は、リスト ボックスの背景色を決定します。次のコードは、ResultsList とタグの付いたリスト ボックスの作成関数を示します。

```
function ResultsList_CreateFcn(hObject, eventdata, handles)
% Hint: listbox controls usually have a white background, change
%       'usewhitebg' to 0 to use default. See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',...
        get(0,'defaultUicontrolBackgroundColor'));
end
```

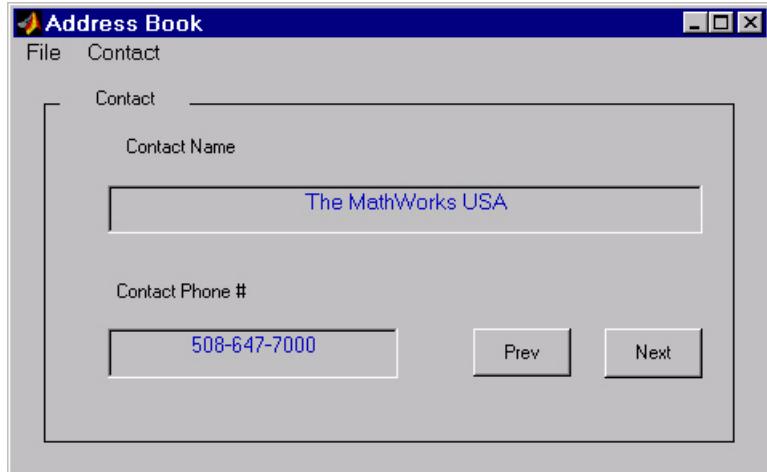
アドレス ブック リーダー

このセクションの内容…

- “アドレス ブック リーダの例について” (p.10-81)
- “アドレス ブック リーダ GUI の表示と実行” (p.10-82)
- “GUI の起動” (p.10-83)
- “アドレス ブックをリーダーに読み込む方法” (p.10-85)
- “Contact Name コールバック” (p.10-88)
- “Contact Phone Number コールバック” (p.10-90)
- “アドレス ブックのページをめくる – Prev/Next” (p.10-91)
- “メニューからアドレス ブックに変更を保存する” (p.10-93)
- “新規作成メニュー” (p.10-94)
- “アドレス ブック サイズ変更関数” (p.10-95)

アドレス ブック リーダの例について

この例では、MAT-ファイルから読み込む名前と電話番号を表示する GUI の実現方法を示します。



この例では、次の GUI プログラム手法を説明します。

- ・ [開く] および [保存] ダイアログ ボックスを使用して、アドレス ブックの MAT-ファイルを配置し開き、変更したアドレス ブック、あるいは新しいアドレス ブックの MAT-ファイルを保存する方法を提供します。
- ・ GUI メニューのために記述されるコールバックを定義します。
- ・ グローバル データを保存し、再び呼び出すために、GUI の handles 構造体を使用します。
- ・ GUI の Figure サイズ 変更 関数を使用します。

グローバル データの管理

この例で説明されているキー テクニックの 1つは、情報の通路を保持し、様々なサブ関数で利用可能とする方法です。この情報は、次の項目を含みます。

- ・ 現在の MAT-ファイルの名前
- ・ MAT-ファイルに保存される名前と電話番号
- ・ 現在の名前と電話番号を示すインデックス ポインター。これらは、アドレス ブックでユーザーのページとして更新する必要があります。
- ・ Figure の位置とサイズ
- ・ すべての GUI コンポーネントのハンドル

次のサブ関数についての記述は、ハンドル 構造体からの情報の保存と受け取り方法を説明します。この構造体の詳細は、“handles 構造体”(p.8-23)と“GUI データ”(p.9-7)を参照してください。

アドレス ブック リーダー GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルと FIG-ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下の手順を行います。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に FIG-ファイルを開くには、`guide address_book` と入力するか、またはここをクリックします。
- 3 エディター内に M ファイルを開くには、`edit address_book` と入力するか、またはここをクリックします。

コンポーネントのプロパティ インスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を修正してから、GUIDE から [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存することができます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUIDE で GUI を調べて実行する場合、代わりに以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 `address_book` の GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の `examples` フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUI ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

GUI の起動

GUI は、ユーザーが他の MATLAB タスクを実行する間、表示されるように設計されているので、ブロックされてなく、モーダルでもありません。

GUI オプションの設定

この GUI は、次の GUI オプション の設定を使用します。

- ・ サイズ変更動作:User-specified
- ・ コマンド ラインへのアクセス:Off
- ・ GUI M ファイル オプションが選択されます。
 - コールバック関数のプロトタイプを作成
 - アプリケーションは、1 つのインスタンスの実行のみ可能とします。

GUI の呼び出し

引数なしで GUI M ファイルを呼び出すことができますが、この場合、GUI は、既定のアドレス ブック MAT-ファイルを使用します。あるいは、GUI が情報を読み込む代わりの MAT-ファイルを指定できます。この例では、ユーザーは、引数の対 `address_book('book', 'my_list.mat')` を用いて GUI を呼び出します。第 1 引数、'book' は、M ファイルが `opening` 関数で探すキー ワードです。M ファイルがキー ワードを見つけると、第 2 引数をアドレス ブックの MAT-ファイルとして使用することができます。この構文を用いて GUI を呼び出すことは、('color', 'red') のような、プロパティ-値の対を用いて呼び出すことと似ています。しかし、'book' は、有効な Figure プロパティではないため、この例では、M ファイルの `opening` 関数は、対 ('book', 'my_list.mat') を識別するコードを含みます。

キー ワード 'book' を使用する必要はありません。構文

`address_book('my_list.mat')` を使用して、引数として MAT-ファイルだけを受け取るように M ファイルをプログラムすることができます。対 ('book', 'my_list.mat') を使用して、GUI を呼び出す利点は、プロパティ-値の対の構文を使用して、有効な Figure プロパティ同様、他のユーザー引数を受け取ることができるよう、GUI をプログラムできることです。すると、GUI は、プロパティ名から、どのプロパティをユーザーが指定したいかを識別することができます。

次のコードは、キー ワード 'book' を探すために `opening` 関数をプログラムする方法を示します。コードがキー ワードを見つけた場合、コンタクトのリストとして 2 番目の引数で指定される MAT-ファイルの使用方法を示します。

```
function address_book_OpeningFcn(hObject, eventdata, ...
    handles, varargin)
% Choose default command line output for address_book
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);
% User added code follows
if nargin < 4
    % Load the default address book
    Check_And_Load([], handles);
    % If the first element in varargin is 'book' and
    % & the second element is a MATLAB file, then load that file
elseif (length(varargin) == 2 && ...
        strcmpi(varargin{1}, 'book') && ...
        (2 == exist(varargin{2}, 'file')))
    Check_And_Load(varargin{2}, handles);
else
    errordlg(' File Not Found' , ' File Load Error' )
    set(handles.Contact_Name, ' String' , ' ')
    set(handles.Contact_Phone, ' String' , ' ')
end
```

アドレス ブックをリーダーに読み込む方法

GUI がアドレス ブック (MAT-ファイルに保存されたテキスト データ) にアクセスする方法は、2 とおりあります。

- ・ GUI を起動するとき、MAT-ファイルを引数として指定できます。たとえば、

```
address_book addrbook.mat
```

引数を指定しないとき、GUI は既定のアドレス ブック (3 つの例ファイルの 1 つである `addrbook.mat`) を読み込みます。

- ・ [ファイル] メニューから、[開く] を選択して、ファイルのダイアログ ボックスを表示し、他の MAT-ファイルをブラウズします。

MAT-ファイルを有効にする

アドレス ブックが有効であるためには、MAT-ファイルは、`Name` および `Phone` と呼ばれる 2 つのフィールドをもつ `Addresses` と呼ばれる構造体を含む必要があります。`Check_And_Load` サブ関数は、次の手順でデータを有効にし読み込みます。

- ・ 指定のファイル、あるいは何も指定されていない場合は既定のファイルを、読み込みます (`load`)。

- ・ MAT-ファイルが、有効なアドレス ブックであるかどうかを決定します。
- ・ データが有効であれば、データを表示します。正しくない場合、エラー ダイアログ ボックス (errordlg) を表示します。
- ・ 有効な MAT-ファイルに対し 1 を返し、(Open メニュー コールバックに使用され) 無効な場合、0 を返します。
- ・ handles 構造体に、次の項目を保存します。
 - MAT-ファイルの名前
 - Addresses 構造体
 - 現在表示されている名前と電話番号を示すインデックス ポインター

Check_And_Load コードのリスト

これは、関数 Check_And_Load です。

```
function pass = Check_And_Load(file, handles)
% Initialize the variable "pass" to determine if this is
% a valid file.
pass = 0;
% If called without any file then set file to the default
% filename.
% Otherwise, if the file exists then load it.
if isempty(file)
    file = 'addrbook.mat';
    handles.LastFile = file;
    guidata(handles.Address_Book, handles)
end
if exist(file) == 2
    data = load(file);
end
% Validate the MAT-file
% The file is valid if the variable is called "Addresses"
% and it has fields called "Name" and "Phone"
flds = fieldnames(data);
if (length(flds) == 1) && (strcmp(flds{1}, 'Addresses' ))
    fields = fieldnames(data.Addresses);
    if (length(fields) == 2) && ...
        (strcmp(fields{1}, 'Name' )) && ...
        (strcmp(fields{2}, 'Phone' ))
        pass = 1;
    end
end
```

```

        (strcmp(fields{2}, 'Phone' ))
    pass = 1;
end
end
% If the file is valid, display it
if pass
    % Add Addresses to the handles structure
    handles.Addresses = data.Addresses;
    guidata(handles.Address_Book, handles)
    % Display the first entry
    set(handles.Contact_Name, 'String', data.Addresses(1).Name)
    set(handles.Contact_Phone, 'String', data.Addresses(1).Phone)
    % Set the index pointer to 1 and save handles
    handles.Index = 1;
    guidata(handles.Address_Book, handles)
else
    errordlg('Not a valid Address Book', 'Address Book Error')
end

```

Open メニュー コールバック

アドレス ブック GUI は、アドレス ブック MAT-ファイルを読み込むための [Open] サブメニューをもつ [File] メニューを含みます。[Open] が選択されると、ファイルをブラウズするためのダイアログ ボックス (uigetfile) が表示されます。ダイアログ ボックスは MAT-ファイルのみ表示しますが、ユーザーはすべてのファイルが表示されるようにフィルターを変更できます。

ダイアログ ボックスは、ファイル名とファイルのパス名の両方を返します。その後、パスが、任意のプラットフォームに対して正しく構成されることを確かめるために fullfile に渡されます。Check_And_Load は、新規アドレス ブックを有効にし、読み込みます。

Open_Callback コードのリスト

```

function Open_Callback(hObject, eventdata, handles)
[filename, pathname] = uigetfile( ...
{'*.mat', 'All MAT-Files (*.mat)' ; ...
'*.*', 'All Files (*.*)' }, ...
'Select Address Book');
% If "Cancel" is selected then return
if isequal([filename, pathname], [0, 0])

```

```

        return
% Otherwise construct the fullfilename and Check and load
% the file
else
    File = fullfile(pathname,filename);
    % if the MAT-file is not valid, do not save the name
    if Check_And_Load(File,handles)
        handles.LastFile = File;
        guidata(hObject, handles)
    end
end

```

メニューの作成に関する詳細は、“メニューの作成”(p.6-99)を参照してください。

Contact Name コールバック

[Contact Name] テキスト ボックスは、アドレス ブックの項目の名前を表示します。新しい名前を入力して、Enter キーを押すと、コールバックは、次の手順を実行します。

- ・ 現在のアドレス ブックに名前が存在する場合、対応する電話番号が表示されます。
- ・ 名前が存在しない場合、質問ダイアログ ボックス (questdlg) が、新規項目を生成したいか、キャンセルして前に表示された名前に戻るかどうかを尋ねます。
- ・ 新規項目を作成する場合、[ファイル] > [保存] メニューで、MAT-ファイルを保存する必要があります。

データのスコアと受け取り

コールバックは、アドレス ブックの内容にアクセスし、インデックス ポインター (handles.Index) を保持するために、handles 構造体を利用します。インデックス ポインターを用いて、このコールバックは、ユーザーによる変更前に表示された名前がいずれであるかを決定することができます。インデックス ポインターは、現在表示される名前を示します。このアドレス ブックとインデックス ポインター フィールドは、GUI が起動されるとき、関数 Check_And_Load により追加されます。

ユーザーが、新規項目を追加する場合、コールバックは、アドレス ブックに新規の名前を追加し、新しい値が表示されるよう、インデックス ポインターを更新します。更新されたアドレス ブックとインデックス ポインターは、handles 構造体に、再び保存されます (guidata)。

Contact Name コールバック

```
function Contact_Name_Callback(hObject, eventdata, handles)
% Get the strings in the Contact Name and Phone text box
Current_Name = get(handles.Contact_Name, 'string');
Current_Phone = get(handles.Contact_Phone, 'string');
% If empty then return
if isempty(Current_Name)
    return
end
% Get the current list of addresses from the handles structure
Addresses = handles.Addresses;
% Go through the list of contacts
% Determine if the current name matches an existing name
for i = 1:length(Addresses)
    if strcmp(Addresses(i).Name, Current_Name)
        set(handles.Contact_Name, 'string', Addresses(i).Name)
        set(handles.Contact_Phone, 'string', Addresses(i).Phone)
        handles.Index = i;
        guidata(hObject, handles)
        return
    end
end
% If it's a new name, ask to create a new entry
Answer=questdlg('Do you want to create a new entry?' , ...
    'Create New Entry' , ...
    'Yes' , 'Cancel' , 'Yes');
switch Answer
case 'Yes'
    Addresses(end+1).Name = Current_Name; % Grow array by 1
    Addresses(end).Phone = Current_Phone;
    index = length(Addresses);
    handles.Addresses = Addresses;
    handles.Index = index;
    guidata(hObject, handles)
    return
case 'Cancel'
    % Revert back to the original number
    set(handles.Contact_Name, 'String', Addresses(handles.Index).Name)
```

```

)
set(handles.Contact_Phone, 'String', Addresses(handles.Index).Phone)
return
end

```

Contact Phone Number コールバック

[Contact Phone #] テキストボックスは、[Contact Name] テキストボックスにリストされる項目の電話番号を表示します。新規番号を入力し、プッシュボタンの 1つをクリックすると、コールバックは、既存の番号を変更するか、あるいはユーザーの変更をキャンセルするかどうかを尋ねる質問ダイアログボックスを開きます。

[Contact Name] テキストボックスのように、このコールバックは、インデックスポインター (handles.Index) を使用して、アドレスブックにおける新規番号を更新し、ユーザーが、質問ダイアログで [Cancel] をクリックすると、前に表示された番号に戻ります。現在のアドレスブックとインデックスポインターは、両方とも、データが、他のコールバックに利用できるように、handles 構造体に保存されます。

新規項目を作成する場合、[ファイル] > [保存] メニューで、MAT-ファイルを保存する必要があります。

Contact_Phone_Callback コードのリスト

```

function Contact_Phone_Callback(hObject, eventdata, handles)
Current_Phone = get(handles.Contact_Phone, 'string');
% If either one is empty then return
if isempty(Current_Phone)
    return
end
% Get the current list of addresses from the handles structure
Addresses = handles.Addresses;
Answer=questdlg('Do you want to change the phone number?', ...
    'Change Phone Number', ...
    'Yes', 'Cancel', 'Yes');
switch Answer
case 'Yes'
    % If no name match was found create a new contact
    Addresses(handles.Index).Phone = Current_Phone;

```

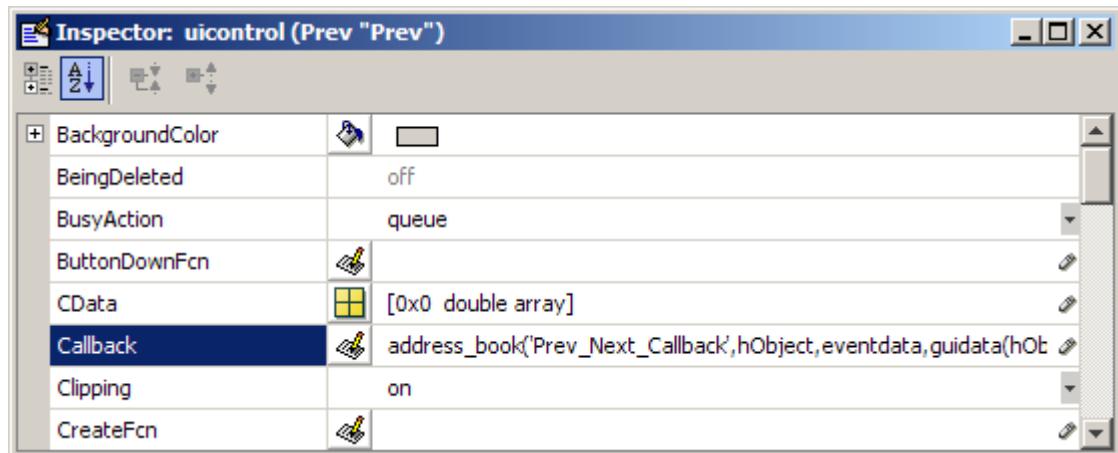
```

handles.Addresses = Addresses;
guidata(hObject, handles)
return
case ' Cancel'
    % Revert back to the original number
    set(handles.Contact_Phone, ...
        'String', Addresses(handles.Index).Phone)
    return
end

```

アドレス ブックのページをめくる - Prev/Next

[Prev] および [Next] ボタンをクリックすると、アドレス ブックの項目を前後して、ページをめくることができます。両方のプッシュ ボタンは、同じコールバック、Prev_Next_Callback を使用します。次の [Prev] プッシュ ボタン Callback プロパティ設定が示すように、両方のプッシュ ボタンのCallback プロパティを設定してこのサブ関数を呼び出す必要があります。



どのボタンがクリックされるかを判別

コールバック は、付加的な引数 str を定義します。これは、どのボタン [Prev] あるいは [Next] が、クリックされるかを示します。[Prev] ボタンの Callback プロパティに 対して、Callback 文字列は、最後の引数として、' Prev' を含みます。[Next] ボタンの Callback 文字列は、最後の引数として、' Next' を含みます。str の値は、各ボタンの機能を実現するため case 文の中で使用されます。下記のコードのリストを参照してください。

前後にページをめくる

Prev_Next_Callback は、handles 構造体から、現在のインデックス ポインターとアドレスを取得します。ユーザーがクリックするボタンに応じて、インデックス ポインターは、減らされるか、あるいは増やされ、対応するアドレスと電話番号が表示されます。最終手順では、handles 構造体のインデックス ポインターに対する新規の値を保存し、guidata を使用して、更新された構造体を保存します。

Prev_Next_Callback コードのリスト

```
function Prev_Next_Callback(hObject, eventdata, handles, str)
% Get the index pointer and the addresses
index = handles.Index;
Addresses = handles.Addresses;
% Depending on whether Prev or Next was clicked,
% change the display
switch str
case 'Prev'
    % Decrease the index by one
    i = index - 1;
    % If the index is less than one then set it equal to the index
    % of the last element in the Addresses array
    if i < 1
        i = length(Addresses);
    end
case 'Next'
    % Increase the index by one
    i = index + 1;
    % If the index is greater than the size of the array then
    % point to the first item in the Addresses array
    if i > length(Addresses)
        i = 1;
    end
end
% Get the appropriate data for the index in selected
Current_Name = Addresses(i).Name;
Current_Phone = Addresses(i).Phone;
set(handles.Contact_Name, 'string', Current_Name)
set(handles.Contact_Phone, 'string', Current_Phone)
% Update the index pointer to reflect the new index
```

```
handles.Index = i;
guidata(hObject, handles)
```

メニューからアドレス ブックに変更を保存する

アドレス ブックを変更するとき、現在の MAT-ファイルを保存するか、あるいは新規の MAT-ファイルとして保存する必要があります。[ファイル] サブメニューの [保存] および [別名で保存] により、これを行うことが可能になります。メニュー エディターを用いて生成されるこれらのメニューは、同じコールバック Save_Callback を使用します。

このコールバックは、[保存] あるいは [別名で保存] がコールバック オブジェクト(すなわち、そのハンドルが、コールバック関数の第 1 引数として渡されるオブジェクト)であるかどうかを同定するために、メニュー タグ (Tag) プロパティを使用します。ユーザーは、メニュー エディターを使用して、メニューのタグ (Tag) プロパティを指定します。

Addresses 構造体の保存

handles 構造体は、Addresses 構造体を含みます。ユーザーは、現在読み込まれる MAT-ファイル (handles.LastFile) の名前と同様に、この構造体 (handles.Addresses) を保存する必要があります。ユーザーが、名前、あるいは数に変更を行うとき、Contact_Name_Callback、あるいは Contact_Phone_Callback は、handles.Addresses を更新します。

MAT-ファイルの保存

[保存] を選択する場合、`save` コマンドは、新規の名前と電話番号をもつ現在の MAT-ファイルを保存するために呼び出されます。

[別名で保存] を選択する場合、ダイアログ ボックスが表示され (`uiputfile`)、既存の MAT-ファイルの名前を選択したり、あるいは新規のファイルを指定できるようにします。ダイアログ ボックスは、選択されるファイル名とパスを返します。最後の手順は、次の項目を含みます。

- ・ プラットフォームに依存しないパス名を生成するために、`fullfile` を使用します。
- ・ MAT-ファイルに新規データを保存するために、`save` を呼び出します。
- ・ 新規 MAT-ファイル名を含むために、handles 構造体を更新します。
- ・ handles 構造体を保存するために、`guidata` を呼び出します。

Save_Callback コードのリスト

```

function Save_Callback(hObject, eventdata, handles)
% Get the Tag of the menu selected
Tag = get(hObject, 'Tag');
% Get the address array
Addresses = handles.Addresses;
% Based on the item selected, take the appropriate action
switch Tag
case 'Save'
    % Save to the default addrbook file
    File = handles.LastFile;
    save(File, 'Addresses')
case 'Save_As'
    % Allow the user to select the file name to save to
    [filename, pathname] = uiputfile(...,
        {'*.mat'; '*.*' }, ...
        'Save as');
    % If 'Cancel' was selected then return
    if isequal([filename, pathname], [0, 0])
        return
    else
        % Construct the full path and save
        File = fullfile(pathname, filename);
        save(File, 'Addresses')
        handles.LastFile = File;
        guidata(hObject, handles)
    end
end

```

新規作成メニュー

[新規作成] メニューは、新規の名前と数の追加を容易にするために、[Contact Name] および [Contact Phone #] テキストフィールドをクリアします。新規項目を作成した後、ユーザーは、[保存] あるいは [別名で保存] メニューを使用して、アドレスブックを保存する必要があります。このコールバックは、テキスト String プロパティを空の文字列に設定します。

```

function New_Callback(hObject, eventdata, handles)
set(handles.Contact_Name, 'String', '')
set(handles.Contact_Phone, 'String', '')

```

アドレス ブック サイズ変更関数

アドレス ブックは、それ自身のサイズ変更関数を定義します。このサイズ変更関数を使用するには、[アプリケーション オプション] ダイアログ ボックスの [サイズ変更アクション] を User-specified に設定する必要があります。これは、今度は Figure の ResizeFcn プロパティを次のように設定します。

```
address_book('ResizeFcn', gcbo, [], guidata(gcbo))
```

ユーザーが Figure をサイズ変更するときはいつでも、MATLAB は、アドレス ブック M ファイル (address_book.m) 内の ResizeFcn サブ関数を呼び出します。

サイズ変更関数の動作

関数 `resize` を用いると、Figure の幅を広げることができますので、長い名前や番号を与えることができますが、Figure を元の幅よりも狭くすることはできません。また、ユーザーが、高さを変更することはできません。これらの制限は、関数 `resize` を簡単化します。この関数は、Figure サイズと GUI 内のコンポーネントとの適切な比率を保持する必要があります。

ユーザーが、Figure をサイズ変更し、マウスをはなすとき、関数 `resize` が実行されます。その時点で、サイズ変更した Figure の寸法が保存されます。次の節では、関数 `resize` の動作を説明します。

幅の変更

新規の幅が、オリジナルの幅よりも大きい場合、Figure を新規の幅に設定します。

[Contact Name] テキスト ボックスのサイズは、新規 Figure の幅に比例して変化します。これは、次のことによってなされます。

- ・ テキストボックスの `Units` を `normalized` に変更します。
- ・ テキスト ボックスの幅を、Figure の幅の 78.9% であるように再設定します。
- ・ `Units` を `characters` に、返します。

新規の幅が、オリジナルの幅より、小さい場合、オリジナルの幅を使用してください。

高さの変更

ユーザーが、高さを変更しようとする場合、元の高さを使用してください。しかし、ユーザーが、サイズの変更後、マウス ボタンをはなすときに、関数 `resize` が作動する

ので、関数 `resize` は、必ずしも、スクリーン上の GUI の元の位置を決定することはできません。したがって、関数 `resize` は、マウスがはなされるときに、垂直位置 (Figure Position ベクトルの 2 つめの要素) を補償します。これは、`height` に垂直位置を加え、元の `height` を差し引くことで行われます。

Figure がボトムからサイズ変更されるとき、Figure は同じ位置に留まります。頂点からサイズ変更されるとき、Figure はマウス ボタンがはなされる位置に移動します。

サイズ変更された Figure がスクリーン上にあるかどうかの確認

関数 `resize` は、ユーザーがマウスをはなす位置に関わらず、サイズ変更された Figure がスクリーン上にあることを確かめるために `movegui` を呼び出します。

これが最初に実行するときに、Figure の Position プロパティで指定される大きさと位置をもつ GUI が表示されます。ユーザーは、GUI を作成するとき、プロパティインスペクターを使用して、このプロパティを設定することができます。

ResizeFcn コードのリスト

```
function ResizeFcn(hObject, eventdata, handles)
% Get the figure size and position
Figure_Size = get(hObject, 'Position');
% Set the figure's original size in character units
Original_Size = [ 0 0 94.19.230769230769234];
% If the resized figure is smaller than the
% original figure size then compensate.
if (Figure_Size(3)<Original_Size(3)) | ...
(Figure_Size(4) ~= Original_Size(4))
if Figure_Size(3) < Original_Size(3)
    % If the width is too small then reset to original width.
    set(hObject, 'Position',...
        [Figure_Size(1), Figure_Size(2), ...
        Original_Size(3), Original_Size(4)]);
    Figure_Size = get(hObject, 'Position');
end
if Figure_Size(4) ~= Original_Size(4)
    % Do not allow the height to change.
    set(hObject, 'Position',...
        [Figure_Size(1),...
        Figure_Size(2)+Figure_Size(4)-Original_Size(4),...]
```

```
    Figure_Size(3), Original_Size(4)])  
end  
end  
% Adjust the size of the Contact Name text box.  
% Set the units of the Contact Name field to 'Normalized'.  
set(handles.Contact_Name, 'units', 'normalized')  
% Get its Position.  
C_N_pos = get(handles.Contact_Name, 'Position');  
% Reset it so that it's width remains normalized.  
% relative to figure.  
set(handles.Contact_Name, 'Position', ...  
    [C_N_pos(1) C_N_pos(2) 0.789 C_N_pos(4)])  
% Return the units to 'Characters'.  
set(handles.Contact_Name, 'units', 'characters')  
% Reposition GUI on screen.  
movegui(hObject, 'onscreen')
```

操作確認のためのモーダル ダイアログの使用

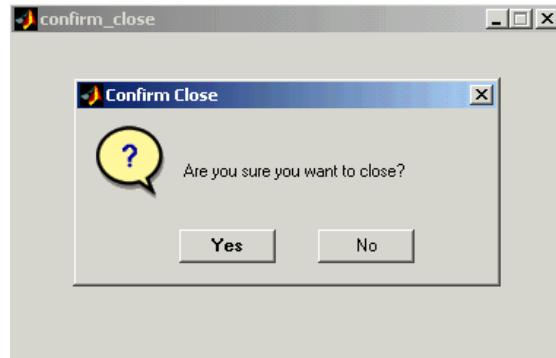
このセクションの内容…

- “モーダル ダイアログの例について” (p.10-98)
- “モーダル ダイアログ ボックス GUI の表示と実行” (p.10-99)
- “Close 確認ダイアログの設定” (p.10-100)
- “[Close] ボタンをもつ GUI の設定” (p.10-101)
- “クローズ確認の GUI の実行” (p.10-102)
- “閉じる確認の GUI の動作” (p.10-103)

モーダル ダイアログの例について

この例では、[Close] ボタンをもつ別の GUI と共にモーダル ダイアログ GUI を使用する方法を説明します。[Close] ボタンをクリックすると、モーダル ダイアログ ボックスが表示されます。これは、閉じる操作を本当に続けたいかどうか、ユーザーに尋ねて確認します。

次の図は、GUI アプリケーション上に表示された、ユーザーの応答を待つダイアログ ボックスを表しています。



モーダル ダイアログ ボックスを閉じるか、あるいは Ctrl キーを押しながら C キーを押すことでモーダル ダイアログ ボックスをノンモーダル ダイアログ ボックスにするまで、ユーザーは、コマンド ウィンドウまたは他の MATLAB ウィンドウにアクセスすることができません。GUI をモーダルにすることができますが、通常はユー

ユーザーが作業中にウィンドウのフォーカスを変更できるように、GUI はノンモーダルです。Figure の WindowStyle プロパティは、GUI がモーダルであるかどうかを判定します。Ctrl キーを押しながら C キーを押すと、'modal' の WindowStyle は、既定値である 'normal' に変更されます。

モーダルの Figure は、既にある Figure ウィンドウの上に重なるので、最も手前にある Figure がモーダルとして存在する限り、既存のウィンドウにはアクセスできなくなります。ただし、モーダルな Figure が表示された後に作成された新規の Figure (たとえば、モーダル GUI により作成されたプロットや別のダイアログ ボックス) は、その上に重ねられ、アクセス可能です。これらも、同様にモーダルになることができます。

モーダル ダイアログ ボックス GUI の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の FIG-ファイルと M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルと FIG-ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下の手順を行います。

- 1 ここをクリックして、ファイルを現在のフォルダーにコピーしてください。
- 2 GUIDE に GUI FIG-ファイルを開くには、`guide modaldlg; guide confirm_close` と入力するか、またはここをクリックします。
- 3 エディターに GUI M ファイルを開くには、`edit modaldlg; edit confirm_close` と入力するか、またはここをクリックします。

コンポーネントのプロパティインスペクターを開くには、レイアウト エディターでコンポーネントをダブルクリックすることで、コンポーネントのプロパティを表示することができます。Figure、M ファイル、またはその両方を修正してから、GUIDE から [ファイル] > [別名で保存] を利用して、現在のフォルダーに GUI を保存することができます。これにより、両方のファイルが保存されます。ユーザーは、名前を変更するように選択することもできます。

GUIDE で GUI を調べて実行する場合、代わりに以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 モーダル GUI を実行するには、ここをクリックします。
- 3 GUIDE レイアウト エディターに GUI を表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

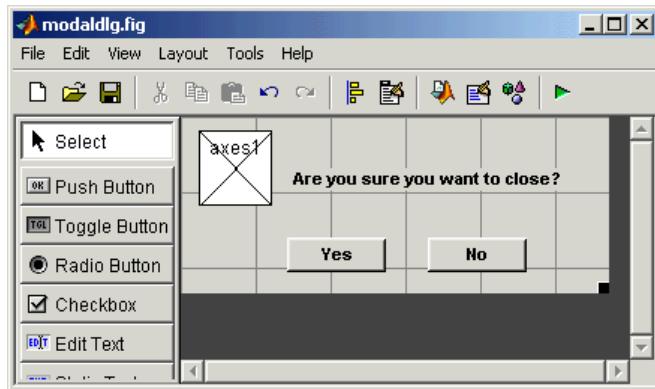
メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。GUIDE ファイルを保存する場合は、GUIDE から [ファイル] > [別名で保存] を利用します。これにより、GUI FIG-ファイルと GUI M ファイルの両方が保存されます。

Close 確認ダイアログの設定

ダイアログを設定するために、次のことを行います。

- 1 GUIDE レイアウト エディターで、[ファイル] メニューから [新規作成] を選択します。
- 2 [GUIDE のクイック スタート] ダイアログ ボックスで、[モーダル クエスチョン ダイアログ] テンプレートを選択し、[OK] をクリックします。
- 3 レイアウト エディターのステティック テキスト、Do you want to create a question dialog? を右クリックし、コンテキストメニューから [プロパティ インスペクター] を選択します。
- 4 プロパティ インスペクターで String までスクロールダウンし、String プロパティを Are you sure you want to close? に変更します。
- 5 [ファイル] メニューから [保存] を選択して、[ファイル名] フィールドに modalDlg.fig を入力します。

GUI は次の図のようになります。



メモ モーダルのダイアログ ボックス (WindowStyle を 'modal' に設定した Figure) は、メニューまたはツールバーを表示できません。

[Close] ボタンをもつ GUI の設定

[Close] ボタンをもつ GUI を設定するには、以下を行います。

- 1 GUIDE レイアウト エディターの [ファイル] メニューから、[新規作成] を選択します。
- 2 [GUIDE のクイック スタート] ダイアログ ボックスで、[空白 GUI (既定値)] を選択して [OK] をクリックします。これにより、新しいレイアウト エディター ウィンドウで、空の GUI が開きます。
- 3 レイアウト エディターのコンポーネント パレットからレイアウト エリアにプッシュ ボタンをドラッグします。
- 4 プッシュ ボタンを右クリックし、コンテキスト メニューから [プロパティ インスペクター] を選択します。
- 5 String プロパティを 'Close' に変更します。
- 6 Tag プロパティを 'close_pushbutton' に変更します。

7 レイアウト エディターのツールバー上の M ファイル エディター アイコンをクリックします。

8 M ファイル エディターのツールバー上の関数の表示アイコンをクリックし、ドロップダウン メニューから close_pushButton_Callback を選択します。

[Close] ボタンのコールバックのための以下の生成コードが、M ファイル エディターに表示されます。

```
% --- Executes on button press in close_pushButton.
function close_pushButton_Callback(hObject, eventdata, handles)
% hObject    handle to close_pushButton (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

9 前のコメントの後に、以下のコードを追加します。

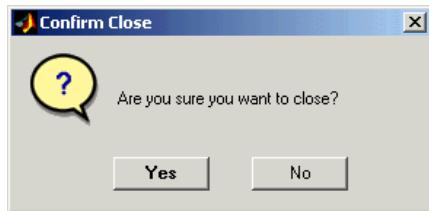
```
% Get the current position of the GUI from the handles structure
% to pass to the modal dialog.
pos_size = get(handles.figure1,'Position');
% Call modaldlg with the argument 'Position'.
user_response = modaldlg('Title','Confirm Close');
switch user_response
case {'No'}
    % take no action
case 'Yes'
    % Prepare to close GUI application window
%
%
%
    delete(handles.figure1)
end
```

クローズ確認の GUI の実行

レイアウト エディター ツールバー上の [実行] ボタンをクリックして、[Close] ボタンをもつ GUI を実行します。GUI は次の図のようになります。



GUI 上の [Close] ボタンをクリックすると、以下の図に示すようにモーダル ダイアログ ボックスが開きます。



[Yes] ボタンをクリックすると、ダイアログとそれを呼び出した GUI の両方が閉じます。[No] ボタンをクリックすると、ダイアログのみ閉じます。

閉じる確認の GUI の動作

ここでは、GUI 上の [Close] ボタンをクリックするときに起こることを説明します。

1 [Close] ボタンをクリックします。すると、そのコールバックは以下を行います。

- 次のコマンドを用いて、handles 構造体から GUI の現在の位置を取得します。

```
pos_size = get(handles.figure1, 'Position')
```

- 以下のコマンドを用いてモーダル ダイアログ ボックスを呼び出します。

```
user_response = modaldlg('Title', 'Confirm Close');
```

これは、プロパティ値の対と共に GUI を呼び出す例です。この場合、Figure プロパティは、「Title」であり、その値は、文字列「Confirm Close」で

す。この構文を用いて `modaldlg` を開くと、ダイアログ ボックスの上部にテキスト “Confirm Close” が開きます。

2 モーダル ダイアログ ボックスを呼び出す GUI から得られた 'Position' を用いてモーダル ダイアログを開きます。

3 モーダル ダイアログ ボックスの M ファイルの `opening` 関数は、次のようにになります。

- ・ ダイアログをモーダルにします。
- ・ `uiwait` コマンドを実行します。これにより、ダイアログ ボックスは、ユーザーが [Yes] または [No]、あるいはウィンドウの境界のクローズ ボックス (X) をクリックするのを待ちます。

4 2 つのプッシュ ボタンの 1 つをクリックすると、プッシュ ボタンのコールバックは次のようになります。

- ・ `handles` 構造体の出力フィールドを更新します。
- ・ `uiresume` を実行して、`uiwait` が呼び出されたところで `opening` 関数のコントロールを返します。

5 `output` 関数が呼び出されます。これは、出力引数として文字列 Yes あるいは No を返し、次のコマンドでダイアログ ボックスを削除します。

```
delete(handles.figure1)
```

6 [Close] ボタンをもつ GUI がコントロールを取り戻す場合、この GUI は、文字列 Yes あるいは No を受け取ります。答えが 'No' の場合、何も行いません。答えが 'Yes' の場合、[Close] ボタン コールバックは、次のコマンドで GUI を閉じます。

```
delete(handles.figure1)
```

プログラミングにより GUI を作成する

章 11, GUI のレイアウト (p. 11-1)

GUI M ファイルの作成とまとめの方法、そこから GUI を集める方法と、メニューとツールバーの作成方法を示します。クロスプラットフォーム互換のための GUI を設計するガイドを提供します。

章 12, GUI のプログラミング (p. 12-1)

ユーザーが記述したコールバックルーチンが、どのように GUI の動作をコントロールするかについて説明します。コールバックを特定のコンポーネントと関連させる方法を示し、コールバックの構文と引数を説明します。各種コンポーネントをプログラミングする簡単な例を説明します。

章 13, アプリケーション定義のデータの管理 (p. 13-1)

アプリケーション定義のデータを管理するための仕組みを説明し、GUI のコールバック間でデータを共有する方法を説明します。

章 14, コールバック実行の管理 (p. 14-1)

コールバックがどのように実行し、それらのやりとりをどのようにコントロールするかについて説明します。

章 15, プログラミングで作成する GUI の例 (p. 15-1)

GUI を作成するために用いられるいくつかのプログラム手法の応用を説明する 3 つの例を提供します。

GUI のレイアウト

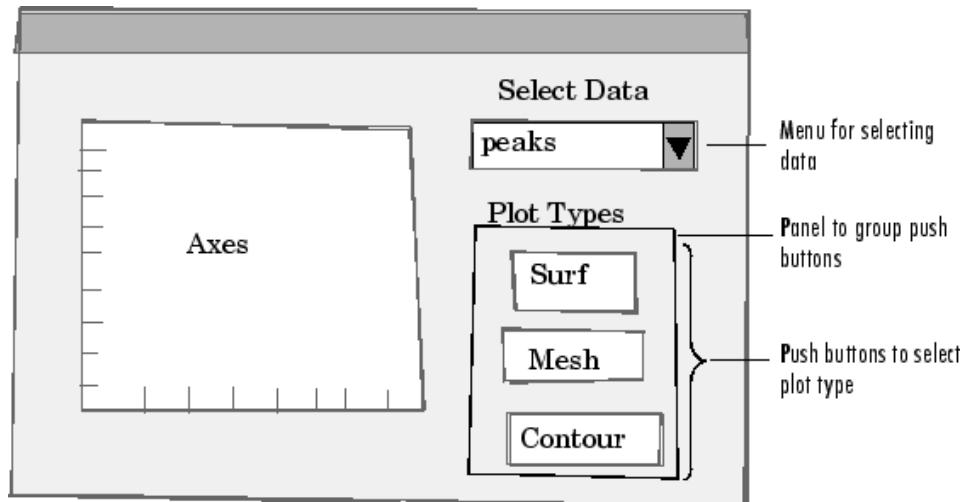
- ・ “GUI の設計” (p.11-2)
- ・ “GUI M ファイルの作成と実行” (p.11-4)
- ・ “GUI の Figure の作成” (p.11-6)
- ・ “GUI にコンポーネントを追加” (p.11-9)
- ・ “対話型ツールを使用した GUI の作成およびコーディング” (p.11-42)
- ・ “タブの順序の設定” (p.11-69)
- ・ “メニューの作成” (p.11-74)
- ・ “ツールバーの作成” (p.11-87)
- ・ “クロスプラットフォーム互換性のための設計” (p.11-93)

GUI の設計

実際の GUI を作成する前に、GUI にさせたいこと、および、どのような動作をさせたいかを決めるることは重要です。GUI を紙に描き、ユーザーが見たり、行うアクションについて予測することは役立ちます。

メモ MATLAB では、1度の関数呼び出しで作成できる標準的なダイアログ ボックスを選択できます。これらのダイアログ ボックスと作成に使用する関数についての詳細は、MATLAB 関数リファレンス ドキュメンテーションの “Predefined Dialog Boxes” を参照してください。

この例で使用する GUI には、ポップアップ メニューから選択するデータを、表面、メッシュ、あるいは等高線図のいずれかの形式で表示できる座標軸コンポーネントが含まれています。次の絵は、設計のための出発点として用いるスケッチを示します。



このパネルは 3 つのプッシュ ボタンを含み、ユーザーの希望でプロットのタイプを選択できます。ポップアップ メニューは、MATLAB 関数に相当する 3 つの文字列、`peaks`、`membrane`、`sinc` を含み、プロットするデータを作成します。プロットするデータをこのメニューから選択できます。

以下のような多くの Web サイトや市販の出版物で、GUI の設計のためのガイドラインが提供されています。

- ・ AskTog – 良い設計についてのエッセイとユーザー インターフェイス設計のための原則のリスト著者 Bruce Tognazzini は一流のユーザー インターフェイス デザイナーです。<http://www.asktog.com/basics/firstPrinciples.html>
- ・ Galitz, Wilbert, O., *Essential Guide to User Interface Design*. Wiley, New York, NY, 2002.
- ・ GUI Design Handbook – GUI コントロールの利用のための詳細なガイド (Web edition) – <http://www.fast-consulting.com/desktop.htm>
- ・ Johnson, J., *GUI Bloopers:Don'ts and Do's for Software Developers and Web Designers*. Morgan Kaufmann, San Francisco, CA, 2000.
- ・ Usability Glossary – GUI 設計、有用性、関連トピックスについての広範囲の用語集。<http://www.usabilityfirst.com/glossary/main.cgi>.
- ・ UsabilityNet – 設計原理、ユーザー中心の設計、さらに他の有用性と設計に関するトピックスを説明します。http://www.usabilitynet.org/management/b_design.htm.

GUI M ファイルの作成と実行

このセクションの内容…

“ファイル編成” (p.11-4)

“ファイル テンプレート” (p.11-4)

“GUI の起動” (p.11-5)

メモ M ファイル作成の例として、このドキュメンテーションの「ご利用の前に」の 章 3, “簡単な GUI をプログラミングで作成する” を参照してください。

ファイル編成

一般に、GUI M ファイルには、以下の順に並んだセクションがあります。最初にセクションを作成するときに、セクションに名前を付けるコメントを追加することによって、構成を保持するのに役立つことができます。

- 1 MATLAB の help コマンドに応答して表示されるコメント
- 2 日付生成などの初期化タスクとコンポーネントを作成するために必要な処理。詳細は、“GUI の初期化” (p.12-3)を参照してください。
- 3 Figure とコンポーネントの作成。詳細は、“GUI の Figure の作成” (p.11-6)と“GUI にコンポーネントを追加” (p.11-9)を参照してください。
- 4 存在するコンポーネントが必要とする初期化タスクと返す出力。詳細は、“GUI の初期化” (p.12-3)を参照してください。
- 5 コンポーネントに対するコールバック。コールバックは、マウス クリックやキー ストロークなど、ユーザーが起こすイベントに応答して実行するルーチンです。詳細は、章 12, “GUI のプログラミング”を参照してください。
- 6 ユーティリティ関数

ファイル テンプレート

これは、GUI M ファイルに対するテンプレートです。

```
function varargout = mygui(varargin)
```

```
% MYGUI Brief description of GUI.
%      Comments displayed at the command line in response
%      to the help command.

% (Leave a blank line following the help.)

% Initialization tasks

% Construct the components

% Initialization tasks

% Callbacks for MYGUI

% Utility functions for MYGUI

end
```

このドキュメンテーションは、入れ子関数を用いて GUI を作成するので、function ステートメントに対応する end ステートメントが必要です。章 12, “GUI のプログラミング” では、このトピックを説明します。

そのファイルをユーザーの現在のフォルダーあるいは MATLAB パス上に保存します。

GUI の起動

ユーザー GUI は、その M ファイルを実行することによって、いつでも表示できます。たとえば、GUI M ファイルが mygui.m の場合、コマンド ラインで次のようにタイプします。

```
mygui
```

必要に応じて、実行時引数を与えます。M ファイルは、ユーザー パスまたは現在のフォルダーになければなりません。

GUI M ファイルを実行する場合、GUI の完全な機能をもつコピーがスクリーン上に表示されます。これに含まれるコンポーネントを操作できますが、M ファイルが、コンポーネントを使用できるように、GUI とコールバックを初期化するコードを含まない限り、何も起こりません。章 12, “GUI のプログラミング” では、この方法を説明します。

GUI の Figure の作成

MATLABにおいて、GUIはFigureです。GUIにコンポーネントを追加する前に、Figureを明示的に作成し、そのハンドルを取得します。ユーザー ファイルの初期化において、次のようなステートメントを使用して、Figureを作成します。

```
fh = figure;
```

fhは、Figureハンドルです。

メモ Figureがない場合にコンポーネントを作成すると、MATLABはFigureを自動的に作成しますが、Figureハンドルを知りません。

Figureを作成すると、Figureに対するプロパティも指定できます。最も一般に使用されるfigureプロパティは、次の表に示されます。

プロパティ	値	説明
MenuBar	figure、none。既定値はfigureです。	MATLAB標準メニュー バーメニューを表示または非表示にします。noneで、ユーザー作成するメニューがない場合、メニュー バー自身が削除されます。
Name	文字列	Figure ウィンドウに表示されるタイトルNumberTitleがonの場合、この文字列はFigureの番号に付加されます。
NumberTitle	on、off。既定値はonです。	文字列'Figure n'(nは、Figureの番号)が、Nameによって指定されるFigure ウィンドウ タイトルの前に置かれるかどうかを決めます。
Position	4要素ベクトル。[左端からの距離、下端からの距離、幅、高さ]です。	GUIのFigureのサイズと、スクリーンの左下隅に対する位置。

プロパティ	値	説明
Resize	on、off。既定値は on です。	ユーザーがマウスを使って Figure ウィンドウをサイズ変更できるかどうか決めます。
Toolbar	auto、none、figure。既定値は auto です。	既定の Figure ツールバーを表示または非表示にします。 <ul style="list-style-type: none"> · none — figure ツールバーを表示しない。 · auto — figure ツールバーを表示しますが、ユーザーインターフェイス コントロール (uicontrol) が figure に追加されると、削除されます。 · figure — figure ツールバーを表示します。
Units	pixels、centimeters、characters、inches、normalized、points。既定値は、pixels です。	Position ベクトルを処理するために用いられる測定の単位
Visible	on、off。既定値は on です。	Figure がスクリーン上に表示されるかどうかを決めます。

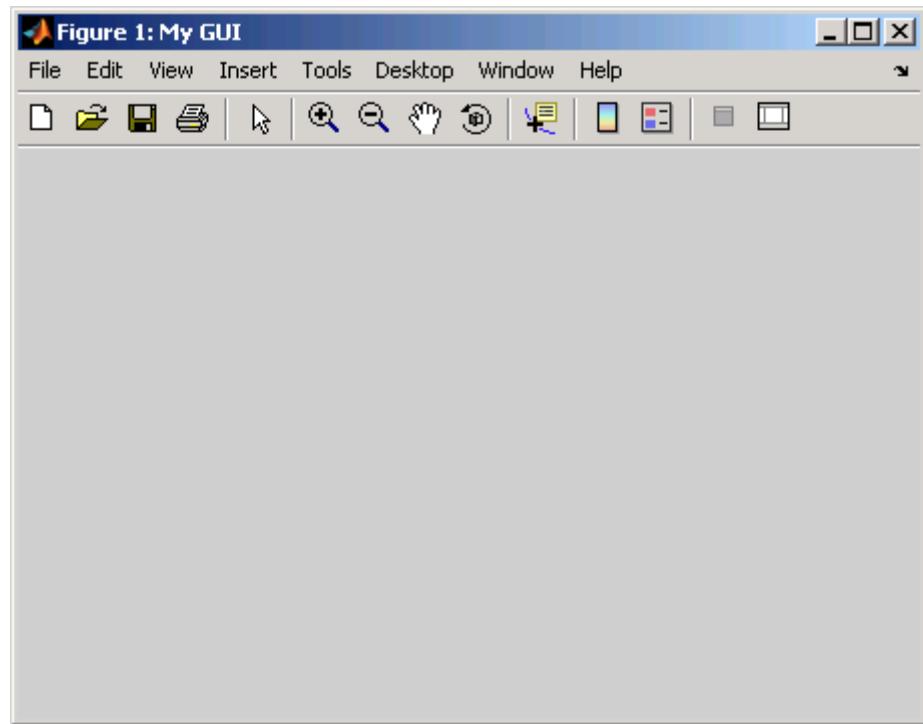
プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB リファレンス ドキュメンテーションの「Figure プロパティ」リファレンス ページを参照してください。

次のステートメントは、Figure に My GUI と名前を付けます。スクリーン上で Figure の位置を決め、コンポーネントが追加されたり初期化されるときに、ユーザーがコンポーネントを見ることがないように GUI を非表示にします。他のプロパティはすべて既定値であるとします。

```
f = figure('Visible','off','Name','My GUI',...
    'Position',[360,500,450,285]);
```

Position プロパティは、スクリーン上の GUI の位置とサイズを指定する 4 要素ベクトル [左端からの距離、下端からの距離、幅、高さ] です。既定の単位はピクセルです。

Figure が表示可能であれば、次のように見えます。



次のトピック、“GUI にコンポーネントを追加”(p.11-9) は、GUI にプッシュ ボタン、座標軸、他のコンポーネントを追加する方法を示します。“メニューの作成”(p.11-74) は、ツールバーとコンテキスト メニューの作成方法を示します。“ツールバーの作成”(p.11-87) は、GUI にユーザーのツールバーを追加する方法を示します。

GUI にコンポーネントを追加

このセクションの内容…

- “利用可能なコンポーネント” (p.11-9)
- “ユーザー インターフェイス コントロールの追加” (p.11-13)
- “パネルとボタン グループを追加する” (p.11-31)
- “座標軸の追加” (p.11-37)
- “ActiveX コントロールの追加” (p.11-40)

利用可能なコンポーネント

コンポーネントは、プッシュ ボタンやスライダーなどのユーザー インターフェイス コントロール、パネルやボタン グループなどのコンテナー、座標軸、ActiveX コントロールを含みます。以下のトピックでは、ユーザー GUI にこれらのコンポーネントを配置する方法を説明します。

メモ MATLAB では、1度の関数呼び出しで作成できる標準的なダイアログ ボックスを選択できます。これらのダイアログ ボックスと作成に使用する関数についての詳細は、MATLAB 関数リファレンス ドキュメンテーションの “Predefined Dialog Boxes” を参照してください。

以下の表は、利用可能なコンポーネントと各コンポーネントの作成に使用する関数を説明します。続くトピックスは、コンポーネントの追加についての固有の情報と手順を説明します。

コンポーネント	関数	説明
ActiveX	actxcontrol	ActiveX コンポーネントにより、GUI に ActiveX コントロールを表示することができます。これらは、Microsoft Windows プラットフォーム上でのみ利用できます。
“座標軸” (p.11-39)	axes	座標軸により、ユーザー GUI がグラフィックス (たとえば、グラフィックスとイメージ) を表示することが可能になります。

コンポーネント	関数	説明
“ボタン グループ” (p.11-35)	uibuttongroup	ボタン グループはパネルに似ていますが、ラジオ ボタンやトグル ボタンに対する排他的な選択を管理するために使用されます。
“チェック ボックス” (p.11-15)	uicontrol	チェック ボックスは、チェックされたときに動作を起こし、状態が、チェックされている、あるいはチェックされていないかを表示します。チェック ボックスは、たとえば、ツールバーを表示するなど、複数の独立な設定を提供するときに役立ちます。
“エディット テキスト” (p.11-16)	uicontrol	エディット テキスト コントロールは、テキスト文字列を入力したり、修正することができます。テキストを入力したい場合、エディット テキストを使用してください。ユーザーは数字を入力できますが、等価な数値に変換する必要があります。
“リスト ボックス” (p.11-19)	uicontrol	リスト ボックスは、項目のリストを表示し、1つあるいは複数の項目を選択することを可能にします。
“パネル” (p.11-33)	uipanel	パネルは、GUI コンポーネントをグループ化します。パネルは、関連するコントロールを視覚的にグループ化して、ユーザー インターフェイスの理解を容易にします。パネルはタイトルと各境界をもつことができます。 パネルの子は、座標軸やユーザー インターフェイス コントロールと同様、パネルやボタン グループもなることができます。パネル内の各コンポーネントの位置は、パネルに相対的に解釈されます。パネルを移動する場合、その子もパネルとともに移動し、パネル上での位置を保ちます。

コンポーネント	関数	説明
“ポップアップ メニュー”(p.11-21)	uicontrol	ポップアップ メニューは、三角形の印を押すと、選択のリストを表示して開きます。
“プッシュ ボタン”(p.11-25)	uicontrol	プッシュ ボタンがクリックされると、アクションを実行します。たとえば、[OK] ボタンは、設定を適用し、ダイアログ ボックスを閉じます。プッシュ ボタンをクリックすると、押された状態に見え、マウスをはなすと、プッシュ ボタンの外見が押されていない状態に戻ります。
“ラジオ ボタン”(p.11-26)	uicontrol	ラジオ ボタンは、チェック ボックスに似ていますが、一般的に、関連するラジオ ボタンのグループ内で、互いに排他的になります。つまり、1 つのボタンを選択すると、以前に選択されたボタンは非選択になります。ラジオ ボタンをアクティブにするために、オブジェクト上でマウス ボタンをクリックしてください。表示は、ボタンの状態を示します。ボタン グループを用いて、ラジオ ボタンの排他的な選択を管理できます。
“スライダー”(p.11-27)	uicontrol	スライダーは、設定した範囲内の数値を、ユーザーが、スライダーまたはサム(thumb)と呼ばれるスライディングバーを移動することにより、設定可能にするものです。ユーザーは、スライダーをクリックしてドラッグしたり、溝または矢印をクリックすることによって、スライダーを移動します。スライダーの位置は、指定した範囲のパーセンテージで表示します。

コンポーネント	関数	説明
“スタティック テキスト”(p.11-29)	uicontrol	スタティック テキストは、テキストの表示をコントロールします。スタティック テキストは、一般に、他のコントロールのラベル付けに使用したり、ユーザーに対する指示を与えたり、あるいはスライダーに関連する値を表示するために使用されます。ユーザーはスタティック テキストを対話で変更できません。
“テーブル”(p.11-23)	uitable	テーブルには、数からなる行、テキスト文字列、列でグループ化される選択が含まれます。テーブルに含まれるデータに応じて、テーブルの大きさが自動的に決まります。行と列には、名前または番号を付けることができます。テーブル セルが選択されたり、または編集されると同時に、コールバックが実行されます。テーブル全体または選択された列は、ユーザー編集可能にすることができます。
“トグル ボタン”(p.11-30)	uicontrol	トグル ボタンは、on あるいは off の状態を示し動作します。トグル ボタンをクリックすると、押された状態に見え、on であることを示します。マウス ボタンをはなすと、トグル ボタンを 2 度目にクリックするまでトグル ボタンは押された状態になります。2 度目にクリックすると、ボタンは off であることを示す上がった状態になります。ボタン グループを用いて、ラジオ ボタンの排他的な選択を管理できます。
ツールバー ボタン	uitoolbar、 uitoggletool、 uipushtool	ノンモーダルの GUI はツールバーを表示し、ツールバーにカスタム アイコンとツールチップで識別できる、プッシュ ボタンとトグル ボタンをもつことができます。

コンポーネントは、コンポーネントを作成するために使用された関数の名前によって参照することができます。たとえば、プッシュ ボタンは関数 `uicontrol` を用いて作成

され、uicontrol と呼ばれることがあります。関数 uipanel を用いて作成されたパネルは、uipanel と呼ばれることがあります。

ユーザー インターフェイス コントロールの追加

ユーザー インターフェイス コントロールを作成するには、関数 uicontrol を使用します。ユーザー インターフェイス コントロールは、プッシュ ボタン、トグル ボタン、スライダー、ラジオ ボタン、エディット テキスト コントロール、スタティック テキスト コントロール、ポップアップ メニュー、チェック ボックス、リスト ボックスを含みます。

メモ これらのコンポーネントの詳細は、“利用可能なコンポーネント”(p.11-9)を参照してください。これらのコンポーネントのプログラミングの基本的な例として、“ユーザー インターフェイス コントロールのプログラミング”(p.12-20)を参照してください。

関数 uicontrol に対する構文は、次のとおりです。

```
uich = uicontrol(parent, 'PropertyName', PropertyValue, ...)
```

uich は、結果のユーザー インターフェイス コントロールのハンドルです。parent を指定しないと、ルートの CurrentFigure プロパティで指定されるように、コンポーネントの親は現在の Figure になります。他の有効な構文は、uicontrol のリファレンス ページを参照してください。

以下のトピックスでは、ユーザー インターフェイス コントロールの一般に利用されるプロパティについて述べ、各コントロールに対する簡単な例を提供します。

- ・ “一般に利用するプロパティ”(p.11-14)
- ・ “チェック ボックス”(p.11-15)
- ・ “エディット テキスト”(p.11-16)
- ・ “リスト ボックス”(p.11-19)
- ・ “ポップアップ メニュー”(p.11-21)
- ・ “テーブル”(p.11-23)
- ・ “プッシュ ボタン”(p.11-25)
- ・ “ラジオ ボタン”(p.11-26)

- ・ “スライダー” (p.11-27)
- ・ “スタティック テキスト” (p.11-29)
- ・ “トグル ボタン” (p.11-30)

一般に利用するプロパティ

以下の表では、ユーザー インターフェイス コントロールを記述するために必要となる、最も一般に利用されるプロパティを示します。

プロパティ	値	説明
Max	スカラー。既定値は 1 です。	最大値。説明は、Style プロパティに依存します。
Min	スカラー。既定値は 0 です。	最小値。説明は、Style プロパティに依存します。
Position	4 要素ベクトル。[左端からの距離、下端からの距離、幅、高さ]。既定値は [20, 20, 60, 20] です。	コンポーネントのサイズとその親に対する相対的な位置。
String	文字列。文字列のセル配列または文字配列になります。	コンポーネント ラベル。リストボックスとpopupmenu の場合は、項目のリストです。ラベルに文字 “&” を表示するには、文字列で 2 つの & 文字を使用します。remove、default、factory (大文字と小文字の区別あり) は予約語です。これらのいずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ (\) を付けます。たとえば、\\$remove は remove を与えます。
Style	pushbutton、togglebutton、radiobutton、checkbox、edit、text、slider、listbox、popupmenu。既定値は pushbutton です。	ユーザー インターフェイス コントロール オブジェクトのタイプ。

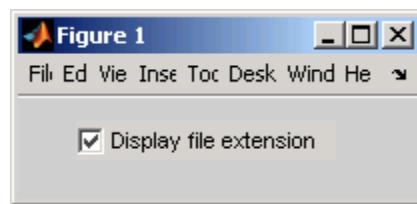
プロパティ	値	説明
Units	pixels、centimeters、characters、inches、normalized、points。既定値は、pixels です。	Position ベクトルを処理するために用いられる測定の単位
Value	スカラーまたはベクトル。	コンポーネントの値。説明は、Style プロパティに依存します。

プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB 関数リファレンスドキュメンテーションの「Uicontrol プロパティ」を参照してください。GUI の動作をコントロールするために必要なプロパティは、章 12, “GUI のプログラミング”に説明します。

チェック ボックス

次のステートメントは、ハンドル cbh をもつチェック ボックスを作成します。

```
cbh = uicontrol(fh, 'Style', 'checkbox', ...
    'String', 'Display file extension', ...
    'Value', 1, 'Position', [30 20 130 20]);
```



最初の引数 fh は、親の Figure のハンドルを指定します。親をパネルまたはボタン グループとして指定することもできます。詳細は、“パネル”(p.11-33)と“ボタン グループ”(p.11-35)を参照してください。

Style プロパティ checkbox は、ユーザー インターフェイス コントロールをチェック ボックスとして指定します。

String プロパティは、チェック ボックスを Display file extension としてラベルします。チェック ボックスのテキストは 1 行に限られます。指定された String を収められ

ないような幅の狭いコンポーネントを指定すると、MATLAB は省略記号を用いて文字列を短くします。



Value プロパティは、ボックスがチェックされたかどうかを指定します。Value に Max プロパティ(既定値は 1)の値を設定して、ボックスがチェックされたコンポーネントを作成します。チェック ボックスがチェックされていない状態にするには、Value を Min(既定値は 0)に設定します。これに対応して、ユーザーがチェック ボックスをクリックするときに、チェック ボックスをチェックすると、MATLAB は Value を Max に設定し、チェック ボックスをチェックしないと、Min に設定します。

Position プロパティは、リスト ボックスの位置とサイズを指定します。この例では、リスト ボックスは幅が 130 ピクセルで高さが 20 ピクセルです。これは、Figure の左端から 30 ピクセルで、下端から 20 ピクセルの位置にあります。このステートメントは、Units プロパティの既定値 pixels を仮定します。

メモ イメージをラベルとして使用することもできます。詳細は、“プッシュ ボタンへのイメージの追加”(p.11-26)を参照してください。

エディット テキスト

次のステートメントは、ハンドル eth をもつエディット テキスト コンポーネントを作成します。

```
eth = uicontrol(fh,'Style','edit',...
    'String','Enter your name here.',...
    'Position',[30 50 130 20]);
```



最初の引数 `fh` は、親の Figure のハンドルを指定します。親をパネルまたはボタングループとして指定することもできます。詳細は、“パネル”(p.11-33)と“ボタングループ”(p.11-35)を参照してください。

`Style` プロパティ `edit` は、ユーザー インターフェイス コントロールをエディット テキスト コンポーネントとして指定します。

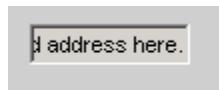
`String` プロパティは、コンポーネントに現われるテキストを定義します。

複数行の入力を可能にするために、`Max - Min` は、次のステートメントのように、1 より大きくなればなりません。MATLAB は、必要に応じて文字列を折り返します。

```
eth = uicontrol(fh, 'Style', 'edit', ...
    'String', 'Enter your name and address here.', ...
    'Max', 2, 'Min', 0, ...
    'Position', [30 20 130 80]);
```



`Max-Min` が 1 以下の場合、エディット テキスト コンポーネントは 1 行のみの入力ができます。指定された文字列を収められないような幅の狭いコンポーネントを指定すると、MATLAB は文字列の一部のみを表示します。ユーザーは矢印キーを使用して、文字列全体にカーソルを移動させることができます。

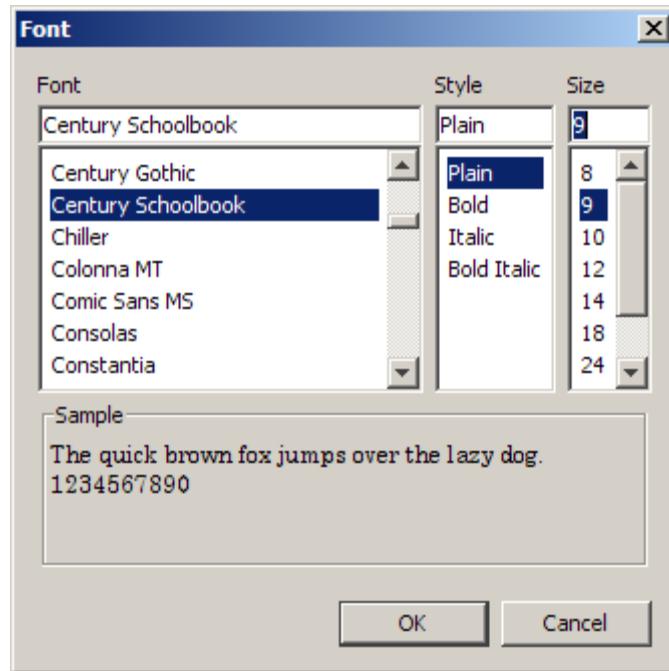


Position プロパティは、エディット テキスト コンポーネントの位置とサイズを指定します。この例では、エディット テキストは幅が 130 ピクセルで高さが 20 ピクセルです。これは、Figure の左端から 30 ピクセルで、下端から 50 ピクセルの位置にあります。このステートメントは、Units プロパティの既定値 pixels を使用します。

フォントの特性の設定. FontName プロパティを用いて、エディット ボックスに表示するテキスト フォントを指定します。Microsoft Windows プラットフォームでは、既定値は MS Sans Serif です。Macintosh と UNIX プラットフォームでは、既定値は Helvetica です。Symbol および Marlett 以外のシステム フォントを使用できます。

エディット ボックスに対してテキスト フォントを選択して、uisetfont GUI からの出力を用いてフォントの特性をすべて同時に設定できます。uisetfont GUI では、利用可能なフォントを一覧表示しプレビューできます。それらの 1 つを選択して [OK] をクリックすると、その名前と他の特性が MATLAB 構造体に返されます。この構造体は、エディット ボックスのフォントの特性を設定するために使用できます。たとえば、標準スタイルとサイズが 9 ポイントの Century Schoolbook を使用するには、以下を行います。

```
font = uisetfont
```



```

font =
    FontName: 'Century Schoolbook'
    FontWeight: 'normal'
    FontAngle: 'normal'
    FontSize: 9
    FontUnits: 'points'

```

メモ 一覧表示されたフォントには、ユーザーのシステムで GUI のユーザーが利用できないものもあります。

ユーザーは構造体のデータを必要な分だけ、M ファイルのステートメントに挿入できます。たとえば、

```
set(eth,'FontName','Century Schoolbook','FontSize',9)
```

フォントを指定するのではなく、uisetfont の GUI からフォントを選択できるように、GUI にプッシュ ボタンまたはコンテキスト メニューを与えることができます。そのためのコールバックは、次のようになります。

```

font = uisetfont;
set(eth,'font')

```

ここで、eth は、ユーザーが設定するフォントをもつ、エディット ボックスのハンドルです。Figure の AppData にハンドルを格納して、getappdata を用いてそのハンドルを取得できます。

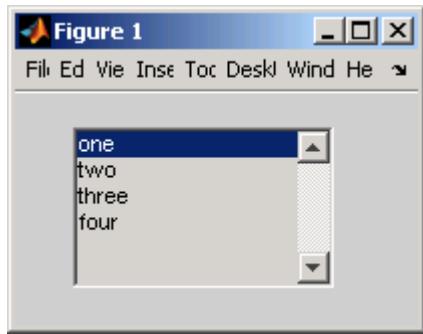
リスト ボックス

次のステートメントは、ハンドル lh をもつリスト ボックスを作成します。

```

lh = uicontrol(fh,'Style','listbox',...
    'String',{'one','two','three','four'},...
    'Value',1,'Position',[30 80 130 20]);

```



最初の引数 `fh` は、親の Figure のハンドルを指定します。親をパネルまたはボタングループとして指定することもできます。詳細は、“パネル”(p.11-33)と“ボタングループ”(p.11-35)を参照してください。

`Style` プロパティ `listbox` は、ユーザー インターフェイス コントロールをリスト ボックスとして指定します。

`String` プロパティは、リスト 項目を定義します。次の表に示す形式のいずれかで項目を指定できます。

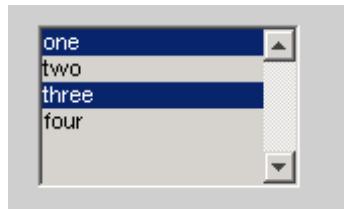
String プロパティの形式	例
文字列のセル配列	{' one' ' two' ' three' }
パディングされた文字列の行列	[' one ' ; ' two ' ; ' three']
垂直スラッシュ () で区切られた文字列のベクトル	[' one two three']

指定した文字列のうちの 1 つ以上が表示できなくなるような幅の狭いコンポーネントを指定すると、MATLAB は省略記号を用いて文字列を短くします。

`Value` プロパティは、コンポーネントが作成されたときに選択される項目（複数の場合もある）を指定します。1 つの項目を選択するには、`Value` を選択したリスト 項目のインデックスを示すスカラーに設定します。ここで 1 はリストの最初の項目に対応します。

複数の項目を選択するには、Value に選択された項目のインデックスのベクトルを設定します。複数の項目の選択を可能にするために、Max - Min は、次のステートメントのように、1 より大きくなればなりません。

```
Ibh = uicontrol(fh,'Style','listbox',...
    'String',{'one','two','three','four'},...
    'Max',2,'Min',0,'Value',[1 3],...
    'Position',[30 20 130 80]);
```



イニシャル セクションが必要ない場合、

- 1 複数の選択が可能になるように、Max と Min プロパティを設定します。
- 2 Value プロパティに空行列 [] を設定します。

リスト ボックスが小さく、すべてのリスト 項目を表示できない場合、ListBoxTop プロパティを、コンポーネントが作成されたときにトップに表示したい項目のインデックスに設定できます。

Position プロパティは、リスト ボックスの位置とサイズを指定します。この例では、リスト ボックスは幅が 130 ピクセルで高さが 80 ピクセルです。これは、Figure の左端から 30 ピクセルで、下端から 20 ピクセルの位置にあります。このステートメントは、Units プロパティの既定値 pixels を使用します。

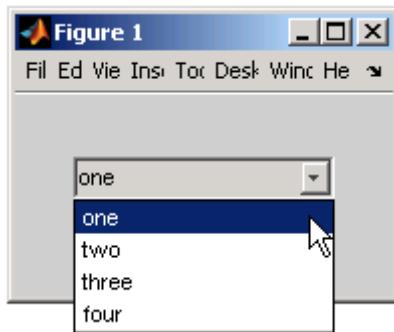
リスト ボックスは、ラベルを提供しません。リスト ボックスにラベリングするには、スタイルック テキストコンポーネントを使用します。

ポップアップ メニュー

次のステートメントは、ハンドル pmh をもつポップアップ メニュー（ドロップダウン メニューまたはコンボ ボックスとも呼ばれる）を作成します。

```
pmh = uicontrol(fh,'Style','popupmenu',...
```

```
'String', {'one', 'two', 'three', 'four'}, ...
'Value', 1, 'Position', [30 80 130 20]);
```



最初の引数 `fh` は、親の `Figure` のハンドルを指定します。親をパネルまたはボタングループとして指定することもできます。詳細は、“パネル”(p.11-33)と“ボタン グループ”(p.11-35)を参照してください。

`Style` プロパティ `popupmenu` は、ユーザー インターフェイス コントロールをポップアップメニューとして指定します。

`String` プロパティは、メニュー項目を定義します。次の表に示す形式のいずれかで項目を指定できます。

String プロパティの形式	例
文字列のセル配列	{'one' 'two' 'three'}
パディングされた文字列の行列	['one' ; 'two' ; 'three']
垂直スラッシュ () で区切られた文字列のベクトル	['one two three']

指定した文字列のうちの 1 つ以上が表示できなくなるような幅の狭いコンポーネントを指定すると、MATLAB は省略記号を用いて文字列を短くします。

`Value` プロパティは、コンポーネントが作成されたときに選択される項目のインデックスを指定します。`Value` に、選択されたメニュー項目のインデックスを示すスカラーを

設定します。1 は、リストの最初の項目に相当します。ステートメントで Value が 2 の場合、メニューは作成されたときに次のように見えます。



Position プロパティは、ポップアップ メニューの位置とサイズを指定します。この例では、ポップアップ メニューは幅が 130 ピクセルです。これは、Figure の左端から 30 ピクセルで、下端から 80 ピクセルの位置にあります。ポップアップ メニューの高さは、フォント サイズで決まります。位置ベクトルに設定する高さは無視されます。このステートメントは、Units プロパティの既定値 pixels を使用します。

ポップアップ メニューは、ラベルを提供しません。ポップアップ メニューをラベリングするためには、スタティック テキスト コンポーネントを使用します。

テーブル

次のコードは、ハンドル th をもつテーブルを作成します。テーブルに行列 magic(5) が与えられたら、Position プロパティの width と height を、その Extent プロパティのものに設定することで、そのサイズを調整します。

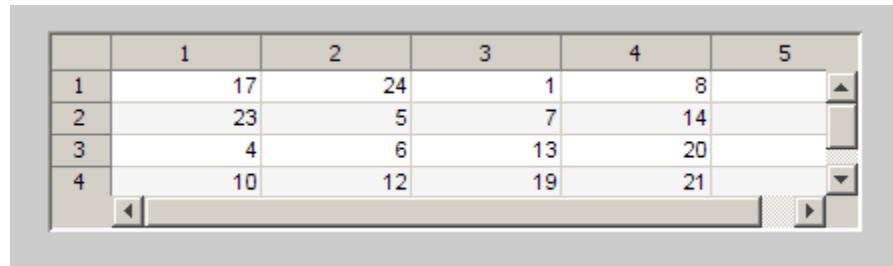
```
th      = uitable(fh,'Data',magic(5));
tpos= get(th,'Position')

tpos =
    20    20    300    300

texn= get(th,'Extent')

texn =
    0    0    407    100

tpos(3) = texn(3);
tpos(4) = texn(4);
set(th, 'Position', tpos)
```



既定では、uitable の大きさは 300×300 ピクセルであり、uitable の Position と Extent の Units の既定値は pixels です。テーブルの Extent はスクロールバーを含むように計算されます。このため、テーブルの Position を上のように設定するときに、データの最後の行と列がわかりにくくなります。

テーブルのセルは、ColumnEditable プロパティにより編集可能となっている場合、ユーザーが編集できます。CellEditCallback は、テーブルのセルが編集されると実行します。既定では、セルは編集可能ではありません。

uitable は Style プロパティをもちませんが、以下を設定することで、いくつかの方法でその外見を変更できます。

- ・ 前景色と背景色
- ・ 行のストライピング
- ・ 行のラベル/番号。
- ・ 列のラベル/番号。
- ・ 列の形式と幅
- ・ フォントの特性

さらに、uitable はプログラムできる 6 つのコールバック関数をもちます。

uitable プロパティの多くは、set コマンドを用いるのではなく、プロパティインスペクターから開くテーブル プロパティエディターを用いて設定できます。この GUI により、テーブルの行、列、色、データに対してプロパティを設定します。

詳細は、uitable と「uitable プロパティ」を参照してください。uitable を含む GUI の例は、“テーブルのデータを表示してグラフを作成する GUI”(p.15-17) を参照してください。

プッシュ ボタン

次のステートメントは、ハンドル pbh をもつプッシュ ボタンを作成します。

```
pbh = uicontrol(fh,'Style','pushbutton','String','Button 1',...
    'Position',[50 20 60 40]);
```



最初の引数 fh は、親の Figure のハンドルを指定します。親をパネルまたはボタン グループとして指定することもできます。詳細は、“パネル”(p.11-33)と“ボタン グループ”(p.11-35)を参照してください。

Style プロパティ pushbutton は、ユーザー インターフェイス コントロールをプッシュ ボタンとして指定します。pushbutton は既定のスタイルなので、' Style' プロパティをステートメントから省略できます。

String プロパティは、プッシュ ボタンを Button 1 とラベルします。プッシュ ボタンのテキストは 1 行に限られます。2 行以上指定すると、最初の行のみが表示されます。指定された String を収められないような幅の狭いコンポーネントを指定すると、MATLAB は省略記号を用いて文字列を短くします。



Position プロパティは、プッシュ ボタンの位置とサイズを指定します。この例では、プッシュ ボタンは幅が 60 ピクセルで高さが 40 ピクセルです。これは、Figure の左端から 50 ピクセルで、下端から 20 ピクセルの位置にあります。このステートメントは、Units プロパティの既定値 pixels を使用します。

プッシュ ボタンへのイメージの追加。 プッシュ ボタンにイメージを追加するには、ボタンの CData プロパティに、トゥルーカラー イメージを定義する RGB 値の $m \times n \times 3$ 配列を割り当てます。たとえば、配列 img は (rand で生成される) 0 から 1 までの乱数の値を使用して、 16×64 のトゥルーカラー イメージを定義します。

```
img(:,:,1) = rand(16,64);
img(:,:,2) = rand(16,64);
img(:,:,3) = rand(16,64);
pbh = uicontrol(fh,'Style','pushbutton',...
    'Position',[50 20 100 45],...
    'CData',img);
```

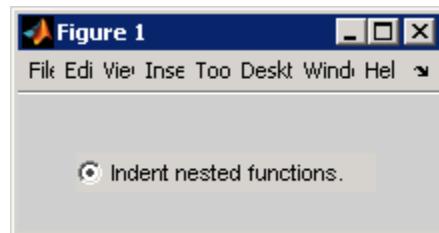


メモ “アイコン エディター” (p.15-60)に述べるアイコン エディターを用いてユーザー独自のアイコンを作成します。行列 X の変換と対応するカラーマップ、すなわち、(X, MAP) イメージから RGB (トゥルーカラー) 形式への変換の詳細は、関数 ind2rgb を参照してください。

ラジオ ボタン

次のステートメントは、ハンドル rbh をもつラジオ ボタンを作成します。

```
rbh = uicontrol(fh,'Style','radiobutton',...
    'String','Indent nested functions.',...
    'Value',1,'Position',[30 20 150 20]);
```



最初の引数 `fh` は、親の `Figure` のハンドルを指定します。親をパネルまたはボタングループとして指定することもできます。ラジオ ボタンとトグル ボタンの排他的な選択を管理するためには、ボタン グループを使用します。詳細は、“パネル”(p.11-33)と“ボタン グループ”(p.11-35)を参照してください。

`Style` プロパティ `radiobutton` は、ユーザー インターフェイス コントロールをラジオ ボタンとして指定します。

`String` プロパティは、ラジオ ボタンを `Indent nested functions` とラベルします。ラジオ ボタンのテキストは 1 行に限られます。2 行以上指定すると、最初の行のみが表示されます。指定された `String` を収められないような幅の狭いコンポーネントを指定すると、MATLAB は省略記号を用いて文字列を短くします。



`Value` プロパティは、コンポーネントが作成されるときに、ラジオ ボタンが選択されるかどうかを指定します。`Value` に、`Max` プロパティの値(既定値は 1)を設定して、ラジオ ボタンが選択された状態のコンポーネントを作成します。ラジオ ボタンを選択されていない状態にするには、`Value` を `Min`(既定値は 0)に設定します。

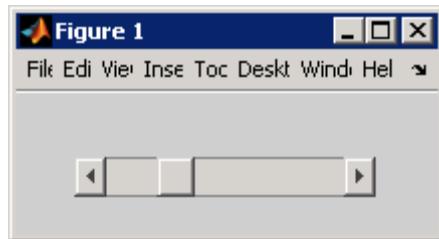
`Position` プロパティは、ラジオ ボタンの位置とサイズを指定します。この例では、ラジオ ボタンは幅が 150 ピクセルで高さが 20 ピクセルです。これは、`Figure` の左端から 30 ピクセルで、下端から 20 ピクセルの位置にあります。このステートメントは、`Units` プロパティの既定値 `pixels` を使用します。

メモ イメージをラベルとして使用することもできます。詳細は、“プッシュ ボタンへのイメージの追加”(p.11-26)を参照してください。

スライダー

次のステートメントは、ハンドル `sh` をもつスライダーを作成します。

```
sh = uicontrol(fh,'Style','slider',...
    'Max',100,'Min',0,'Value',25,...
    'SliderStep',[0.05 0.2],...
    'Position',[30 20 150 30]);
```



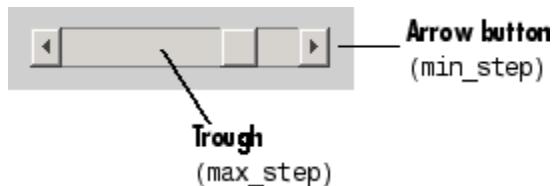
最初の引数 `fh` は、親の Figure のハンドルを指定します。親をパネルまたはボタン グループとして指定することもできます。詳細は、“パネル”(p.11-33)と“ボタン グループ”(p.11-35)を参照してください。

`Style` プロパティ `slider` は、ユーザー インターフェイス コントロールをスライダーとして指定します。

`Max` プロパティは、スライダーの最大値です。`Min` プロパティはスライダーの最小値で、`Max` より小さくなければなりません。

`Value` プロパティは、スライダーが作成されたときにスライダーに表示される値を指定します。`Value` を `Min` 以上 `Max` 以下の数に設定します。指定した範囲外の値を指定すると、スライダーは表示されません。

ユーザーが最小ステップのために矢印ボタンをクリックしたり、最大ステップのためにスライダーの溝をクリックするときに、`SliderStep` プロパティは、スライダーの `Value` が変化する量をコントロールします。`SliderStep` を、各値の範囲が [0, 1] である 2 要素ベクトル `[min_step, max_step]` として指定します。



例では、5 パーセントの最小ステップと 20 パーセントの最大ステップを与えます。既定値 `[0.01 0.10]` は、1 パーセントの最小ステップと 10 パーセントの最大ステップを与えます。

`Position` プロパティは、スライダーの位置とサイズを指定します。この例では、スライダーは幅が 150 ピクセルで高さが 30 ピクセルです。これは、Figure の左端から 30

ピクセルで、下端から 20 ピクセルの位置にあります。このステートメントは、`Units` プロパティの既定値 `pixels` を使用します。

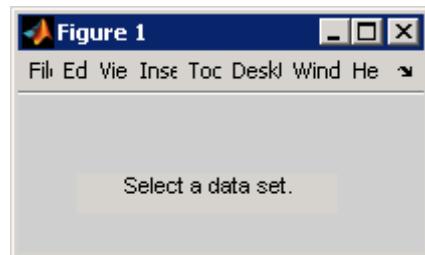
メモ Mac プラットフォームでは、水平方向のスライダーの高さは少なくとも 15 ピクセルでなければなりません。`position` ベクトルに設定する `height` がこれよりも小さいと、スライダーは適切に表示されません。

スライダー コンポーネントには、テキスト記述がありません。スライダーにラベリングするには、スタティック テキストコンポーネントを使用します。

スタティック テキスト

次のステートメントは、ハンドル `sth` をもつスタティック テキストコンポーネントを作成します。

```
sth = uicontrol(fh,'Style','text',...
    'String','Select a data set.',...
    'Position',[30 50 130 20]);
```



最初の引数 `fh` は、親の `Figure` のハンドルを指定します。親をパネルまたはボタン グループとして指定することもできます。詳細は、“パネル”(p.11-33)と“ボタン グループ”(p.11-35)を参照してください。

`Style` プロパティ `sth` は、ユーザー インターフェイス コントロールをスタティック テキストコンポーネントとして指定します。

`String` プロパティは、コンポーネントに現われるテキストを定義します。指定された `String` を収められないような幅の狭いコンポーネントを指定すると、MATLAB は省略記号を用いて文字列を短くします。

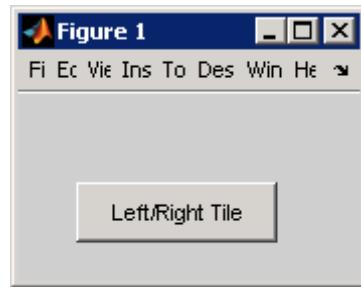


Position プロパティは、スタティック テキストコンポーネントの位置とサイズを指定します。この例では、スタティック テキストは幅が 130 ピクセルで高さが 20 ピクセルです。これは、Figure の左端から 30 ピクセルで、下端から 50 ピクセルの位置にあります。このステートメントは、Units プロパティの既定値 pixels を使用します。

トグル ボタン

次のステートメントは、ハンドル tbh をもつトグル ボタンを作成します。

```
tbh = uicontrol(fh,'Style','togglebutton',...
    'String','Left/Right Tile',...
    'Value',0,'Position',[30 20 100 30]);
```



最初の引数 fh は、親の Figure のハンドルを指定します。親をパネルまたはボタン グループとして指定することもできます。ラジオ ボタンとトグル ボタンの排他的な選択を管理するためには、ボタン グループを使用します。詳細は、“パネル”(p.11-33)と“ボタン グループ”(p.11-35)を参照してください。

Style プロパティ togglebutton は、ユーザー インターフェイス コントロールをトグル ボタンとして指定します。

String プロパティは、トグル ボタンを Left/Right Tile とラベルします。トグル ボタンのテキストは 1 行に限られます。2 行以上指定すると、最初の行のみが表示されます。指定された String を収められないような幅の狭いコンポーネントを指定すると、MATLAB は省略記号を用いて文字列を短くします。



Value プロパティは、コンポーネントが作成されるときに、トグル ボタンが選択されるかどうかを指定します。Value に、Max プロパティの値（既定値は 1）を設定して、トグル ボタンが選択された（押された）状態のコンポーネントを作成します。トグル ボタンを選択されていない（上がった）状態にするには、Value を Min（既定値は 0）に設定します。次の図は、押された状態のトグル ボタンを示します。



Position プロパティは、トグル ボタンの位置とサイズを指定します。この例では、トグル ボタンは幅が 100 ピクセルで高さが 30 ピクセルです。これは、Figure の左端から 30 ピクセルで、下端から 20 ピクセルの位置にあります。このステートメントは、Units プロパティの既定値 pixels を使用します。

メモ イメージをラベルとして使用することもできます。詳細は、“プッシュ ボタンへのイメージの追加”(p.11-26)を参照してください。

パネルとボタン グループを追加する

パネルとボタン グループは、GUI コンポーネントをグループにまとめて配置するコンテナです。パネルやボタン グループを移動すると、その子も共に移動し、パネルやボタン グループ上での相対的な位置を保ちます。

メモ これらのコンポーネントの詳細は、“利用可能なコンポーネント”(p.11-9)を参照してください。

これらのコンポーネントを作成するには、関数 `uipanel` と関数 `uibuttongroup` を使用します。

パネルに対する構文は、次のようにになります。

```
ph = uipanel(fh, 'PropertyName', PropertyValue, ...)
```

ph は、結果のパネルのハンドルです。最初の引数 fh は、親の Figure のハンドルを指定します。親をパネルまたはボタン グループとして指定することもできます。他の有効な構文は、uipanel のリファレンス ページを参照してください。

ボタン グループに対する構文は、次のとおりです。

```
bgh = uibuttongroup('PropertyName', PropertyValue, ...)
```

bgh は、結果のボタン グループのハンドルです。ボタン グループに対して、Parent プロパティを使用してコンポーネントの親を指定する必要があります。他の有効な構文は、uibuttongroup のリファレンス ページを参照してください。

パネルとボタン グループの両方に対して親を指定しないと、コンポーネントの親は、ルートの CurrentFigure プロパティで指定される現在の Figure になります。

以下のトピックスでは、パネルやボタン グループの一般に利用されるプロパティについて述べ、各コンポーネントに対する簡単な例を提供します。

- ・ “一般に利用するプロパティ” (p.11-32)
- ・ “パネル” (p.11-33)
- ・ “ボタン グループ” (p.11-35)

一般に利用するプロパティ

次の表では、パネルまたはボタン グループを記述するために必要となる、最も一般に利用されるプロパティを示します。

プロパティ	値	説明
Parent	ハンドル	コンポーネントの親の Figure、パネル、またはボタン グループの、ハンドル。
Position	4 要素ベクトル。[左端からの距離、下端からの距離、幅、高さ] です。既定値は [0, 0, 1, 1] です。	コンポーネントのサイズとその親に対する相対的な位置。

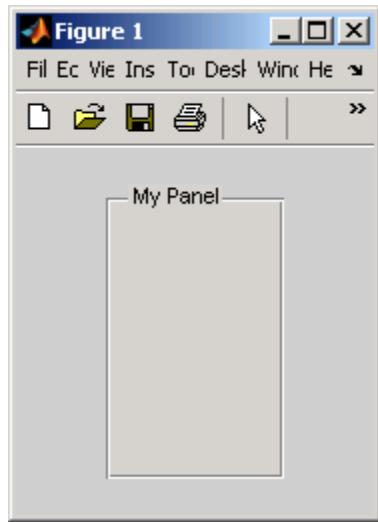
プロパティ	値	説明
Title	文字列	コンポーネント ラベル。ラベルに文字 “&” を表示するには、文字列で 2 つの & 文字を使用します。remove、default、factory (大文字と小文字の区別あり) は予約語です。これらのいずれかをラベルとして使用するには、文字列の先頭にバックスラッシュ (\) を付けます。たとえば、\\$remove は remove を与えます。
TitlePosition	lefttop、centertop、righttop、leftbottom、centerbottom、rightbottom。既定値は lefttop です。	パネルまたはボタン グループについて、タイトルの文字列の位置。
Units	normalized、centimeters、characters、inches、pixels、points。既定値は normalized です。	Position ベクトルを処理するために用いられる測定の単位。

プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB 関数リファレンスドキュメンテーションの「Uipanel プロパティ」と「Uibuttongroup プロパティ」を参照してください。GUI の動作をコントロールするために必要なプロパティは、章 12, “GUI のプログラミング”に説明します。

パネル

次のステートメントは、ハンドル ph をもつパネルを作成します。GUI のコンポーネントをグループ化するためにパネルを使用します。

```
ph = uipanel('Parent', fh, 'Title', 'My Panel', ...
    'Position', [.25 .1 .5 .8]);
```



Parent プロパティは、親 Figure のハンドル `fh` を指定します。親をパネルまたはボタン グループとして指定することもできます。

Title プロパティは、パネルに `My Panel` とラベルします。

このステートメントは、`TitlePosition` プロパティの既定値 `lefttop` を使用します。

`Units` プロパティは、`Position` プロパティを解釈するために使用されます。このパネルは、`Units` プロパティの既定値 `normalized` を使用します。これによって、Figure がサイズ変更されるときにパネルが自動的にサイズ変更されます。サイズ変更動作の詳細は、Figure の `ResizeFcn` プロパティの説明を参照してください。

`Position` プロパティは、パネルの位置とサイズを指定します。この例では、パネルの幅は Figure の幅の 50 パーセントで、パネルの高さは Figure の高さの 80 パーセントです。座標軸は、Figure の左端から Figure の幅の 25 パーセントで、下端から Figure の高さの 10 パーセントの位置にあります。Figure がサイズ変更されるとき、これらの比率は保たれます。

次のステートメントは、ハンドル `ph` をもつパネルに 2 つのプッシュ ボタンを追加します。パネル内の各コンポーネントの `Position` プロパティは、パネルに相対的に解釈されます。

```
pbh1 = uicontrol(ph,'Style','pushbutton','String','Button 1',...
    'Position',[10,10,100,50]);
```

```
' Units' , ' normalized' , ...  
' Position' , [ .1 .55 .8 .3]);  
pbh2 = uicontrol(ph,' Style' , ' pushbutton' , ' String' , ' Button 2' , ...  
' Units' , ' normalized' , ...  
' Position' , [ .1 .15 .8 .3]);
```

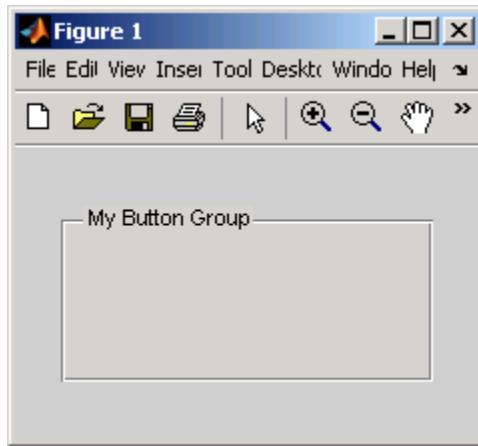
プッシュ ボタンの追加についての詳細は、“プッシュ ボタン”(p.11-25)を参照してください。



ボタン グループ

次のステートメントは、ハンドル bgh をもつボタン グループを作成します。ラジオ ボタンとトグル ボタンを排他的に管理するには、ボタン グループを使用します。

```
bgh = uibuttongroup(' Parent' , fh,' Title' , ' My Button Group' , ...  
' Position' , [ .1 .2 .8 .6]);
```



Parent プロパティは、親 Figure のハンドル `fh` を指定します。親をパネルまたはボタン グループとして指定することもできます。

`Title` プロパティは、ボタン グループに `My Button Group` とラベルします。

このステートメントは、`TitlePosition` プロパティの既定値 `lefttop` を使用します。

`Units` プロパティは、`Position` プロパティを解釈するために使用されます。このボタン グループは、`Units` プロパティの既定値 `normalized` を使用します。これによって、Figure がサイズ変更されるときにボタン グループが自動的にサイズ変更されます。サイズ変更動作の詳細は、Figure の `ResizeFcn` プロパティの説明を参照してください。

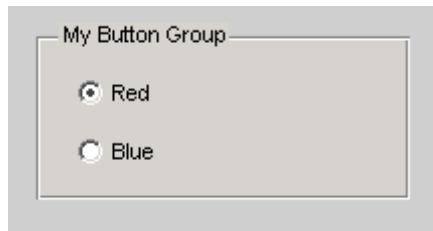
`Position` プロパティは、ボタン グループの位置とサイズを指定します。この例では、ボタン グループの幅は Figure の幅の 80 パーセントで、ボタン グループの高さは Figure の高さの 60 パーセントです。座標軸は、Figure の左端から Figure の幅の 10 パーセントで、下端から Figure の高さの 20 パーセントの位置にあります。Figure がサイズ変更されるとき、ボタン グループは、これらの比率を保ちます。

次のステートメントは、ハンドル `bgh` をもつボタン グループに 2 つのラジオ ボタンを追加します。

```
rbh1 = uicontrol(bgh,'Style','radiobutton','String','Red',...
    'Units','normalized',...
    'Position',[.1 .6 .3 .2]);
rbh2 = uicontrol(bgh,'Style','radiobutton','String','Blue',...
    'Units','normalized',...
    'Position',[.1 .6 .3 .2]);
```

```
' Units' , ' normalized' , ...
' Position' , [.1 .2 .3 .2]);
```

既定では、MATLAB は、ボタン グループに追加された最初のラジオ ボタンを自動的に選択します。ラジオ ボタンの `Value` プロパティを使用して、初期選択を明示的に指定できます。詳細は、“ラジオ ボタン”(p.11-26)を参照してください。



座標軸の追加

次のようなコマンド(`plot`、`surf`、`line`、`bar`、`polar`、`pie`、`contour`、`mesh`などのコマンド)を使用する場合、座標軸により、ユーザー GUI が、グラフィックス(たとえば、グラフィックスとイメージ)を表示することが可能になります。

メモ このコンポーネントの詳細は、“利用可能なコンポーネント”(p.11-9)を参照してください。

座標軸を作成するには、関数 `axes` を使用します。この関数に対する構文は、次のとおりです。

```
ah = axes('PropertyName', PropertyValue, ...)
```

`ah` は、結果の座標軸のハンドルです。`Parent` プロパティを使用して、座標軸の親を指定する必要があります。`Parent` を指定しない場合、その親は、ルートの `CurrentFigure` プロパティによって指定されるように、現在の `Figure` です。他の有効な構文は、`axes` のリファレンス ページを参照してください。

以下のトピックスでは、座標軸の一般的に使用されるプロパティについて述べ、簡単な例を提供します。

- ・ “一般に利用するプロパティ”(p.11-38)

- “座標軸”(p.11-39)

一般に利用するプロパティ

次の表では、座標軸を記述するために必要となる、最も一般に利用されるプロパティを示します。

プロパティ	値	説明
HandleVisibility	on、callback、off。既定値は on です。	オブジェクトのハンドルが、このオブジェクトの親に対する子のリストで表示されるかどうかを決めます。座標軸の場合、HandleVisibility を callback に設定し、コマンドライン操作から保護します。
NextPlot	add、replace、replacechildren。既定値は replace です。	プロットでグラフィックスを追加するかどうか、グラフィックスを置き換えて Axes のプロパティを既定値にリセットする、あるいはグラフィックスのみを置き換える、を指定します。
Parent	ハンドル	コンポーネントの親の Figure、パネル、またはボタン グループの、ハンドル。
Position	4要素ベクトル。[左端からの距離、下端からの距離、幅、高さ] です。	コンポーネントのサイズとその親に対する相対的な位置。
Units	normalized、centimeters、characters、inches、pixels、points。既定値は normalized です。	位置ベクトルを処理するために用いられる測定の単位。

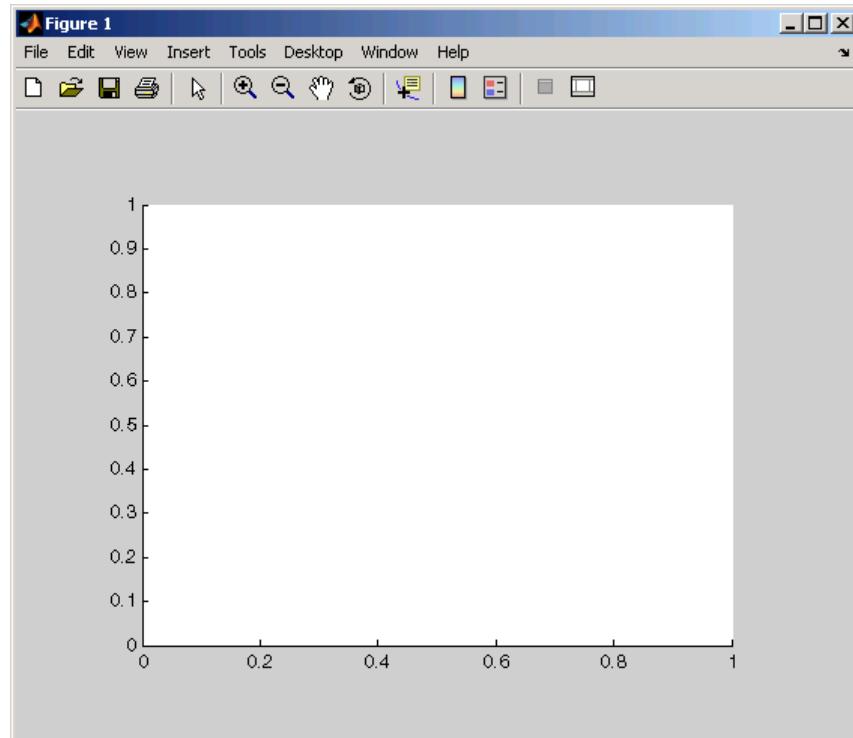
プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB 関数リファレンス ドキュメンテーションの「Axes プロパティ」を参照してください。GUI の動作をコントロールするために必要なプロパティは、章 12, “GUI のプログラミング”に説明します。

Axes オブジェクト、plot、surf、line、bar、polar、pie、contour、imagesc、mesh の詳細は、以下のようなコマンドを参照してください。完全なリストとして、MATLAB 関数リファレンス ドキュメンテーションの“Function Reference”を参照してください。

座標軸

次のステートメントは、ハンドル ah をもつ座標軸を作成します。

```
ah = axes('Parent', fh, 'Position', [.15 .15 .7 .7]);
```



Parent プロパティは、親 Figure のハンドル fh を指定します。親をパネルまたはボタン グループとして指定することもできます。

Units プロパティは、Position プロパティを解釈するため使用されます。この座標軸は、Units プロパティの既定値 normalized を使用します。これによって、Figure がサイズ変更されるときに座標軸が自動的にサイズ変更されます。サイズ変更動作の

詳細は、Figure プロパティのリファレンス ページで「`ResizeFcn`」プロパティの説明を参照してください。

`Position` プロパティは、座標軸の位置とサイズを指定します。この例では、座標軸の幅は Figure の幅の 70 パーセントで、座標軸の高さは Figure の高さの 70 パーセントです。座標軸は、Figure の左端から Figure の幅の 15 パーセントで、下端から Figure の高さの 15 パーセントの位置にあります。Figure がサイズ変更されると、座標軸はこれらの比率を保ちます。

MATLAB は、自動的に目盛を追加します。座標軸に描かれるほとんどの関数は、目盛を適切に更新します。

カスタマイズされた Axes のプロパティがリセットされないようにする。 `plot`、`image`、`scatter`、その他の、データをグラフ化する関数の多くは、Axes に描画する前に既定で Axes プロパティをリセットします。このことは、プロット間で Axes の範囲、目盛り、Axis の色、フォントの特性を同じものにする必要がある GUI では、問題になることがあります。

この動作は、Axes の `NextPlot` プロパティが既定値 '`replace`' であるときに行われます。この値の場合、グラフがプロットされるか再プロットされるときに Axes からすべてのコールバックを削除すると、GUI はプロットを作成できなくなります。GUI に対して適切な値は、多くの場合 '`replacechildren`' です。そのため、グラフを作成するコールバックには、グラフを描くことで Axes のコンテンツを変更する前に以下のようないくつかのコードを含む必要があります。

```
set(ah, 'NextPlot', 'replacechildren')
```

これは、GUI の色、フォント、コンテキストメニュー、または `ButtonDownFcn` など、GUI が必要とする、既存の Axes のプロパティ値をリセットせずに、グラフをプロットします。`NextPlot` がこのように設定される例として、GUIDE ドキュメンテーションの“Tablestat の拡張”(p.10-52) を参照してください。

ActiveX コントロールの追加

ActiveX コンポーネントにより、GUI に ActiveX コントロールを表示することができます。これらは、Microsoft Windows プラットフォーム上でのみ利用できます。

ActiveX コントロールは、Figure すなわち GUI 自身に対してのみ子になることができます。これは、パネルやボタン グループの子になることはできません。

ActiveX コントロールの Figure への追加についての詳細は、MATLAB External Interfaces ドキュメンテーションの “Creating an ActiveX Control” を参照してください。ActiveX コントロールの一般的な情報は、MATLAB External Interfaces ドキュメンテーションの “Creating COM Objects” を参照してください。

対話型ツールを使用した GUI の作成およびコーディング

このセクションの内容...
“コンポーネントの位置の対話的な設定” (p.11-43)
“コンポーネントの整列” (p.11-52)
“色の対話的な設定” (p.11-58)
“フォントの特性の対話的な設定” (p.11-59)
“コンポーネントプロパティを設定するためのコードの生成” (p.11-62)
“GUI 開発ツールのまとめ” (p.11-67)

プログラムによる GUI のレイアウトには時間がかかることがあります。また、多数の小さな手順が含まれます。たとえば、コンポーネントを希望通りの場所に正確に配置するには、多くの場合複数回、コンポーネントを手動で配置しなければなりません。他のプロパティの最終設定を行い、それらのステートメントをコーディングするのにも時間がかかります。こうした労力を削減するには、組み込みの MATLAB ツールおよび GUI を利用してコンポーネントプロパティの値を設定します。次の節では、いくつかのツールについて説明します。

モードまたはツール	使用目的	コマンド
プロット編集モード	対話的なプロットの編集や注釈の追加	plotedit
プロパティエディター	オブジェクトのグラフィカルなプロパティの編集	propedit、propertyeditor
プロパティインスペクター	大部分のオブジェクトプロパティの対話的な表示および編集	inspect
整列ツール	コンポーネントを互いに整列して配置	align
カラー セレクター	カラー パレットからの色の選択とその値の取得	uisetcolor
フォント セレクター	文字のフォント、スタイルおよびサイズのプレビューとそれらの値の選択	uisetfont

これらのツールの中には、プロパティ値を返すものもあれば、プロパティ値を返さずにプロパティを対話的に編集できるようにするものもあります。特に、プロパティインスペクターでは、ほとんどすべてのオブジェクトプロパティを対話的に設定できます。その後、プロパティ値をコピーし、コマンド ウィンドウまたはコード ファイルに貼り付けることができます。ただし、Color、Position など、ベクトル値のプロパティを取得する場合、プロパティ インスペクターでは、一度に 1 つしか値をコピーできません。

コンポーネントの位置の対話的な設定

GUI コンポーネントの初期位置や他のプロパティが気に入らない場合は、それらを手動で調整できます。GUI の Figure をプロット編集モードにすることにより、マウスを使用して様々なコンポーネントのプロパティを移動、サイズ変更、整列および変更できます。その後、変更したプロパティの値を読み取り、GUI コード ファイルにコピーしてコンポーネントを初期化できます。

プロット編集モードで位置を設定するには、次のようにします。

- 1 プロット編集モードになります。矢印ツール  をクリックするか、[ツール] メニューから [プロット編集] を選択します。Figure にメニューまたはツールバーがない場合は、コマンド ウィンドウで `plotedit on` と入力します。
- 2 コンポーネントを選択します。編集するコンポーネントの上で左マウス ボタンをクリックします。
- 3 コンポーネントを移動し、サイズ変更します。コンポーネント内でクリックし、ドラッグして新しい位置に移動します。黒い四角のハンドルをクリックし、ドラッグしてその形を変更します。矢印キーを使用して微調整します。
- 4 操作したコンポーネントのハンドルを確認してください。次のコードでは、ハンドルは、`object_handle` という名前の変数です。
- 5 プロパティ インスペクターからコンポーネント位置ベクトルを取得します。次のように入力します。

```
inspect
```

あるいは、次のような `get` ステートメントを入力します。

```
get(object_handle, 'Position')
ans =
    15.2500 333.0000 106.0000 20.0000
```

6 結果 (ans) をコピーし、コード ファイル内の set ステートメントに大かっこで囲んで挿入します。

```
set(object_handle, 'Position', [15.2500 333.0000 106.0000 20.0000])
```

コンポーネントを体系的に配置するには、そのプロセスを管理する関数を作成できます。editpos という関数の簡単な例を次に示します。

```
function rect = editpos(object_handle)
% Enter plot edit mode to reposition an object.
figure_handle = ancestor(object_handle, 'figure');
plotedit(figure_handle)
disp(['Select, move and resize object' inputname(1)])
disp('When you are done, press Enter to continue.')
pause
rect = get(object_handle, 'Position');
inspect(object_handle)
```

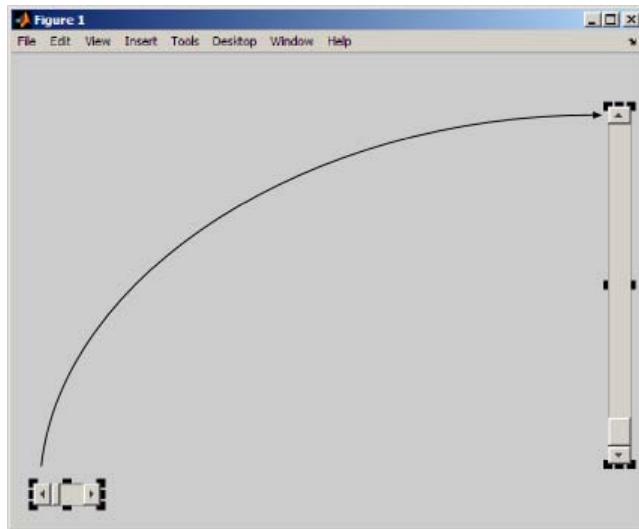
この関数を試してみるには、コマンド ウィンドウで次のコードを入力します。

```
hfig = figure;
hs1 = uicontrol('Style','slider')
editpos(hs1)
```

editpos を呼び出すと、次のプロンプトが表示されます。

```
== Select, move and resize the object. Use mouse and arrow keys.
== When you are finished, press Return to continue.
```

初めてプロット編集モードに入る場合は、Figure 自体が選択されます。スライダーをクリックして選択し、これを再配置します。たとえば、次の図に示すように、Figure の右側にこれを移動し、垂直に方向付けます。



プロット編集モードを使用してプロパティを変更

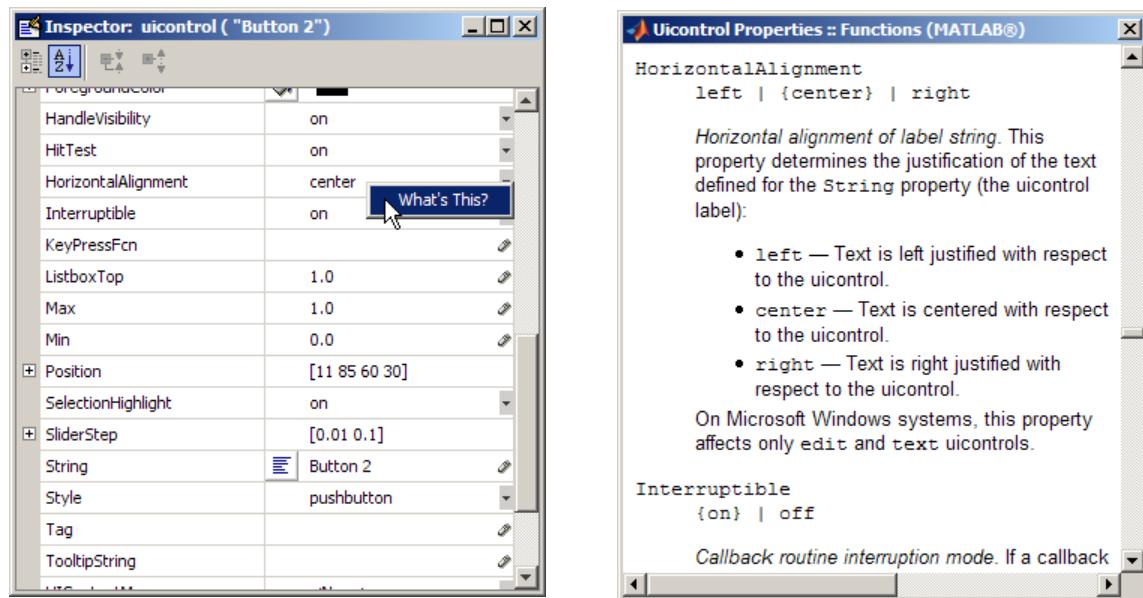
プロット編集モードでオブジェクトを選択した後、プロパティインスペクターを開いてそのプロパティのいずれかを表示および変更できます。オブジェクトが選択されているときに、コマンド ウィンドウで次のように入力します。

```
inspect
```

たとえば、関数形式を使用して、調べたいオブジェクトのハンドルを渡すこともできます。

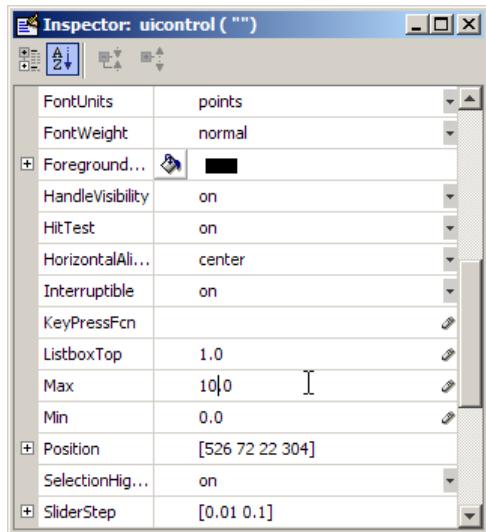
```
inspect(hsl)
```

プロパティインスペクターが開き、オブジェクトプロパティを表示します。プロパティ値を読み込むだけでなく、編集できます。また、コンポーネントは直ちに更新されます。任意のプロパティの定義を確認するには、プロパティインスペクターで名前または値を右クリックし、表示された [これはなに?] メニュー項目をクリックします。次の図に示すように、状況依存ヘルプのウィンドウが開き、そのプロパティの定義を表示します。



ヘルプ ウィンドウ内をスクロールして、他のプロパティの説明を表示します。[X] クローズ ボックスをクリックしてウィンドウを閉じます。

次のインスペクターの図は、インスペクターを使用して、スライダー uicontrol の Max プロパティを既定値 (1.0) から 10.0 に変更する方法を示しています。



プロパティ エディターでの編集

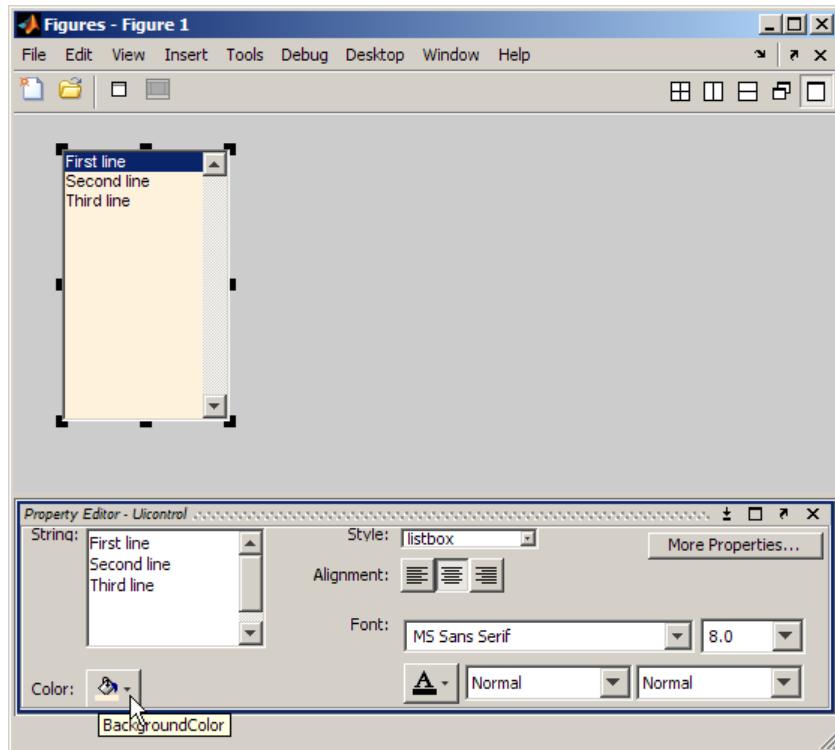
プロパティ エディターはプロパティインスペクターよりグラフィカルなインターフェイスをもちます。このインターフェイスは、コンポーネントの外観に影響するプロパティを設定するのに便利です。コンポーネントについてこれを聞くには、コマンド ウィンドウで次のように入力します。

```
propedit(object_handle)
```

あるいは、引数を省略し、次のように入力します。

```
plotedit on
```

Figure がプロット編集モードになります。編集したいオブジェクトを選択し、プロパティエディターが表示するプロパティを変更します。次の図は、プロパティエディターを使用して変更されたリスト ボックスの BackgroundColor プロパティと String プロパティを示しています。



プロパティエディターで設定できるプロパティの大部分は、外観のプロパティです。その他のプロパティの値にアクセスするには、[その他のプロパティ] をクリックします。プロパティインスペクターが開いて（または、すでに開いている場合は、フォーカスされ）、選択されたオブジェクトのプロパティを表示します。これを使用して、プロパティエディターで表示されないプロパティを変更します。

位置ベクトルのスケッチ

`rbbox` は、位置の設定に役立つ関数です。この関数を呼び出したら、Figure 内の任意の場所にボックス型をドラッグ アウトします。マウス ボタンを離すと、このボックスの位置ベクトルが表示されます。`rbbox` の実行時には、次のことに注意してください。

- ・ Figure ウィンドウがフォーカスされていなければなりません。
- ・ マウス カーソルが Figure ウィンドウ内になければなりません。

- 左マウス ボタンが押されなければなりません。

この動作のため、マウス ボタンが押されるまで待機する関数またはスクリプトから `rbbox` を呼び出さなければなりません。返される位置ベクトルは、Figure 単位で描画される長方形を指定します。次の関数は `rbbox` を使用してコンポーネントの位置を設定します。

```
function rect = setpos(object_handle)
% Use RBBOX to establish a position for a GUI component.
% object_handle is a handle to a uicomponent that uses
% any Units. Internally, figure Units are used.

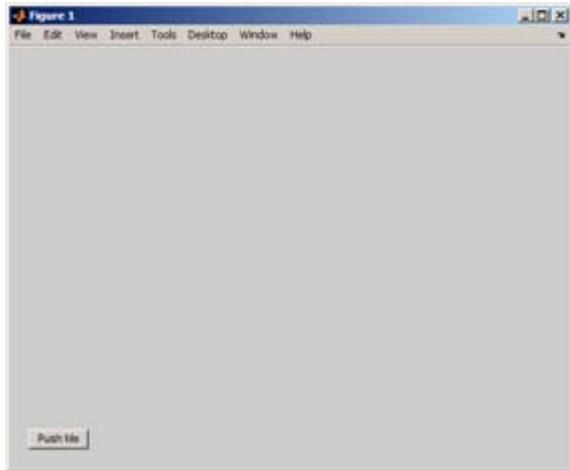
disp([' == Drag out a Position for object ' inputname(1)])
waitForbuttonpress % So that rbbox does not return immediately
rect = rbbox;      % User drags out a rectangle, releases button
% Pressing a key aborts rbbox, so check for null width & height
if rect(3) ~= 0 && rect(4) ~= 0
    % Save and restore original units for object
    myunits = get(object_handle,'Units');
    set(object_handle,'Units',get(gcf,'Units'))
    set(object_handle,'Position',rect)
    set(object_handle,'Units',myunits)
else
    rect = [];
end
```

関数 `setpos` は Figure 単位を使用してコンポーネントの `Position` プロパティを設定します。まず、`setpos` はコンポーネントの `Units` プロパティを取得および保存し、そのプロパティを Figure 単位に設定します。オブジェクト位置を設定した後、この関数はオブジェクトの元の単位に戻します。

次の図は、`setpos` を使用して既定の位置から離れた場所にボタンを再配置する方法を示しています。

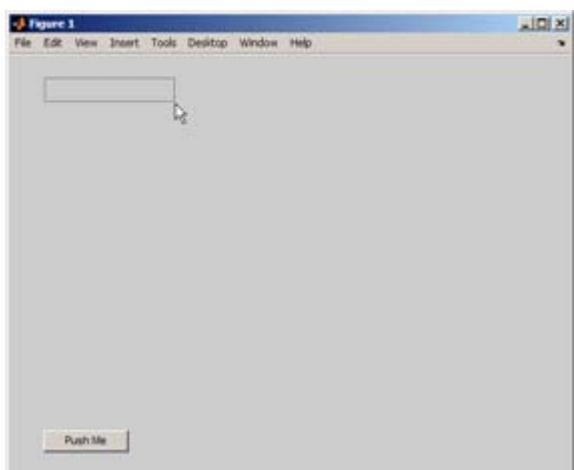
手順 1:

```
figure; btn1 = uicontrol('Style' ,...  
    'pushbutton' , 'String' , 'Push Me');
```



手順 2:

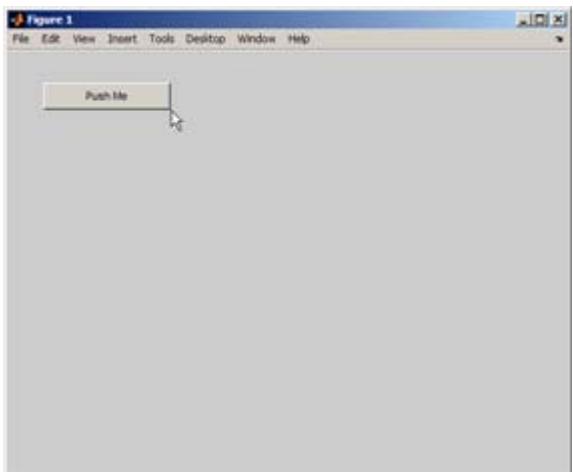
```
rect = setpos(btn1)  
==== Drag out a Position for object btn1
```



手順 3:マウス ボタンを離します。コントロールが移動します。

手順 4:ボタンの Position が設定され、返されます。

```
rect =
```



37 362 127 27

GUI コード ファイルに対して不完全な set コマンドを作成します。

```
set(btn1, 'Position', )
```

rect 位置ベクトルをコピーし、大かっこで囲んで貼り付けます。

```
set(btn1, 'Position', [37 362 127 27])
```

コンポーネントの作成時に setpos を呼び出すことはできません。これは、setpos では引数としてそのコンポーネントのハンドルが必要となるためです。ただし、コンポーネントの作成時にコンポーネントを対話的に配置できる小さな関数を作成できます。この関数は、マウス ボタンが押されるまで待機してから rbbox を呼び出し、マウス ボタンが離されると、長方形の位置を返します。

```
function rect = getrect
disp(' == Click and drag out a Position rectangle.' )
waitForButtonPress % So that rbbox does not return immediately
rect = rbbox; % User drags out a rectangle, releases button
```

getrect を使用するには、次のようにします。

1 uicontrol を呼び出すときに Position プロパティの値として getrect を指定します。

```
btn1 = uicontrol('Style', 'pushbutton', 'String', 'Push Me', ...
    'Position', getrect);
```

(ドラッグ中に文字を入力した場合、rbbox が中止し、エラーが表示され、Uicontrol は作成されません。)

ドラッグが終了すると、新しいコンポーネントによって、作成した長方形が占有されます。

2 次に、入力したステートメントをコピーし、GUI コード ファイル内に配置します。その後、ボタンの Position を取得します。データのベクトルをコピーします。

```
rect = get(btn1, 'Position')
rect =
    55 253 65 25
```

3 getrect の代わりに、GUI コード ファイル内にベクトルを貼り付けます。ベクトルを大かっこで囲みます。座標値の違いを考慮に入れると、ステートメントは次のようになります。

```
btn1 = uicontrol('Style','pushbutton','String','Push Me',...
    'Position',[55 253 65 25]);
```

rbbox は Figure 単位で座標を返すという点に注意してください(この例では、'pixels' です)。コンポーネントの既定の Units 値が Figure と異なる場合は、コンポーネントの作成時に同じになるように指定します。たとえば、Uipanel の既定の Units は 'normalized' です。Uipanel の位置をスケッチするには、次の例のように、Figure Units を使用するコードを使用します。

```
pnl1 = uipanel('Title','Inputs',...
    'Units',get(gcf,'Units'),...
    'Position',getrect)
```

GUI を作成するための 2 つの MATLAB ユーティリティが位置を指定するときに役立ちます。getpixelposition を使用して、コンポーネントの Units 設定に関係なく、コンポーネントの位置ベクトルをピクセル単位で取得します。位置の原点は、コンポーネントまたは囲んでいる Figure の親に関連しています。setpixelposition を使用して、新しいコンポーネント位置をピクセル単位で指定します。これらの関数のいずれかを呼び出した後も、コンポーネントの Units プロパティは変更されません。

コンポーネントの整列

- ・ “整列関数の利用” (p.11-53)
- ・ “整列ツールの使用” (p.11-55)

コンポーネントを配置した後も、コンポーネントがまだ完全に整列していない場合があります。最終調整を行うには、コマンド ウィンドウから関数 `align` を使用します。別の対話的な方法として、Figure メニューから利用できる GUI であるオブジェクト整列ツールを使用します。次の節では、両方のアプローチについて説明します。

整列関数の利用

ユーザー インターフェイス コントロールと座標軸を整列するには、関数 `align` を使用します。この関数によって、コンポーネントを水平方向と垂直方向に整列できます。コンポーネントを全長にわたって等間隔に配置したり、それらを固定した間に指定することもできます。

関数 `align` に対する構文は、次のとおりです。

```
align(HandleList, 'HorizontalAlignment', 'VerticalAlignment')
```

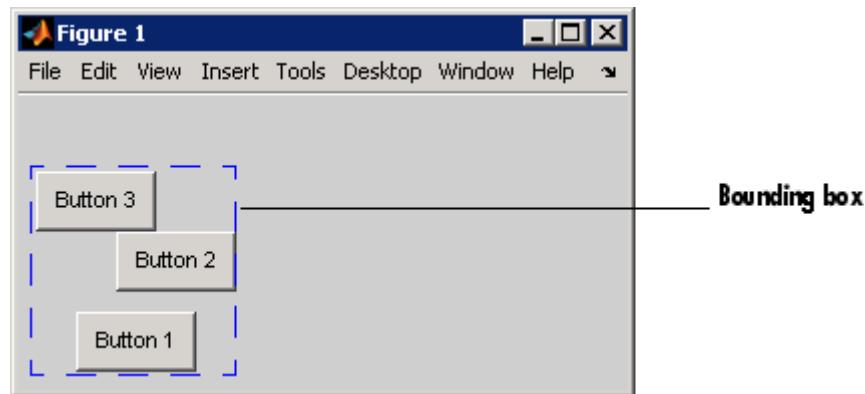
次の表は、これらのパラメーターの可能な値をまとめています。

<i>HorizontalAlignment</i>	<i>VerticalAlignment</i>
None、Left、Center、Right、Distribute、Fixed	None、Top、Middle、Bottom、Distribute、Fixed

`HandleList` のすべてのハンドルは、同じ親をもたなければなりません。他の構文の詳細は、`align` のリファレンス ページを参照してください。

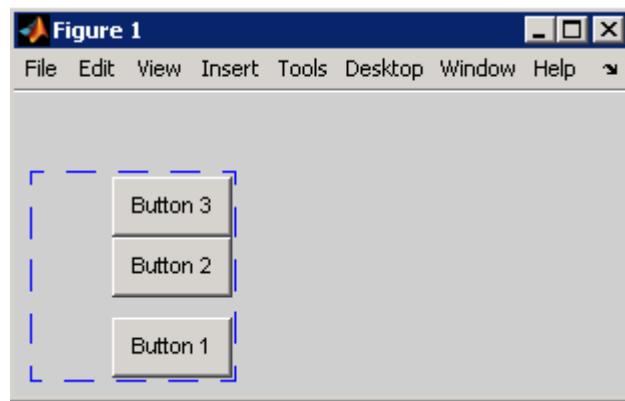
以下のコードは、任意の場所に 3 つのプッシュ ボタンを作成します。続く例はそれぞれ、同じ 3 つのプッシュ ボタンではじまり、これらを別の方法で整列します。Figure 内の青い点線で示すように、この関数はコンポーネントをその境界ボックスに合わせて配置します。

```
fh = figure('Position',[400 300 300 150])
b1 = uicontrol(fh,'Posit',[30 10 60 30],'String','Button 1');
b2 = uicontrol(fh,'Posit',[50 50 60 30],'String','Button 2');
b3 = uicontrol(fh,'Posit',[10 80 60 30],'String','Button 3');
```



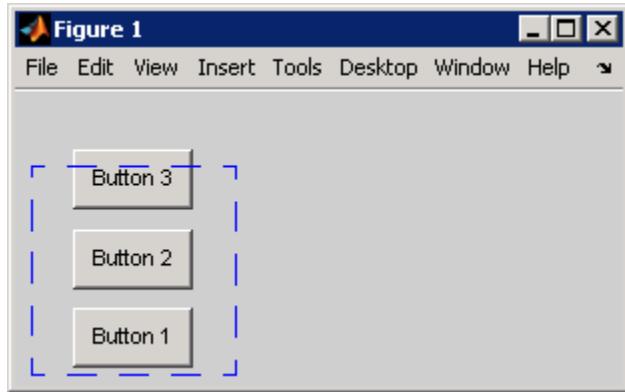
コンポーネントを水平に整列させる。次のステートメントは、プッシュボタンを境界の右に水平に移動させます。これは、その垂直位置を変更しません。この Figure は、オリジナルの境界を示します。

```
align([b1 b2 b3], 'Right', 'None');
```



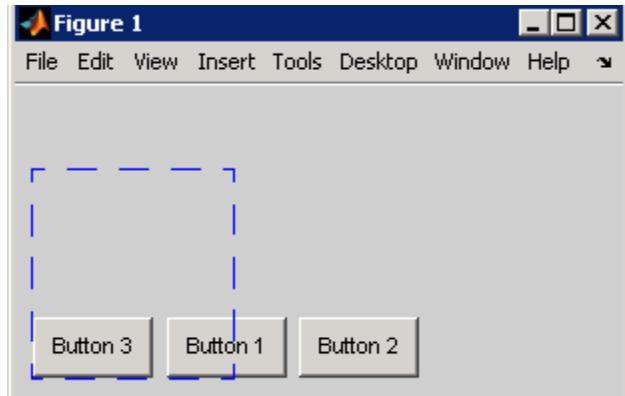
コンポーネントを垂直に配置してコンポーネントを水平方向に整列する。次のステートメントは、プッシュボタンを境界ボックスの中央まで水平方向に移動させ、垂直方向の位置を調整します。`'Fixed'` オプションは、ボックス間の距離を一律にします。距離をポイント単位 (1 ポイント = 1/72 インチ) で指定します。この例では、距離は 7 ポイントです。オリジナルの境界の中央に、プッシュボタンが現れます。下方のプッシュボタンは、オリジナルの境界のボックスの下部に留まります。

```
align([b1 b2 b3], 'Center', 'Fixed', 7);
```



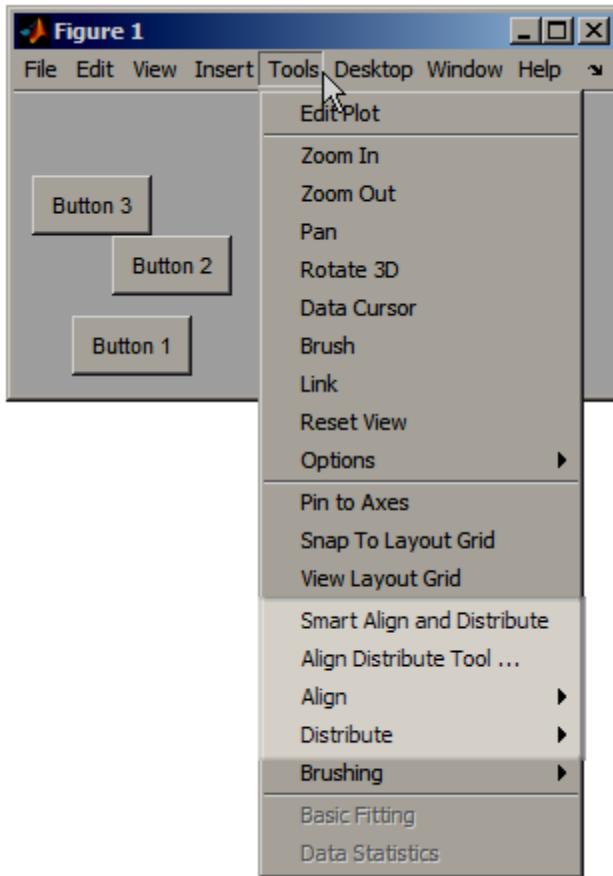
コンポーネントを水平に配置してコンポーネントを垂直方向に整列する。次のステートメントは、プッシュボタンを境界ボックスの下まで垂直方向に移動させます。また、水平方向の位置を調整して、ボックス間の距離を固定した 5 ポイントにします。オリジナルの境界の下に、プッシュボタンが現れます。

```
align([b1 b2 b3], 'Fixed', 5, 'Bottom');
```



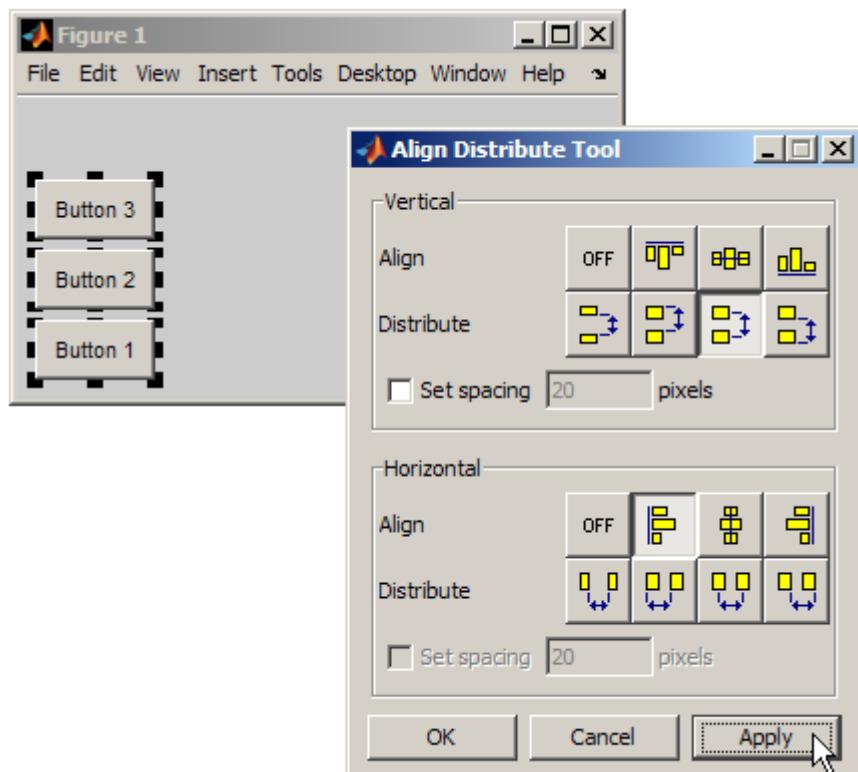
整列ツールの使用

Figure に標準のメニュー バーがある場合、プロット編集モードで直接選択したコンポーネントに対して整列/配置操作を実行できます。[ツール] メニューのいくつかのオプションを使用すると、align 関数コマンドを入力せずに済みます。次の図で、整列および配置メニュー項目は強調表示されています。



次の手順では、オブジェクト整列ツールを使用して GUI のコンポーネントを配置する方法を説明しています。“整列関数の利用”(p.11-53)で説明されているように、このツールは関数 `align` と同じオプションを提供します。

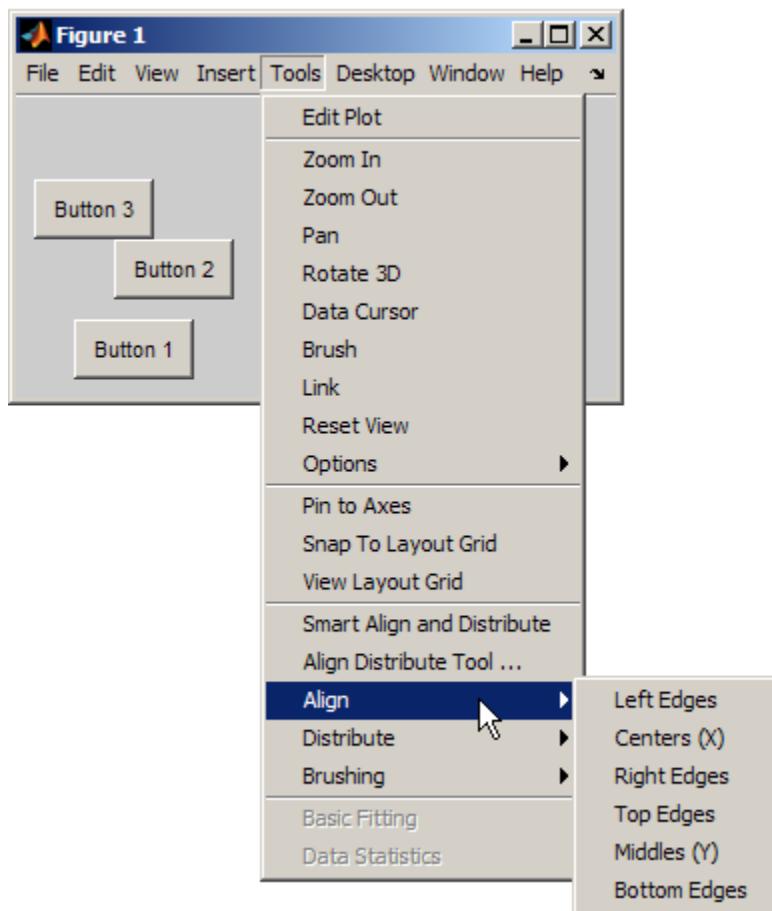
- 1 [ツール] メニューから [オブジェクト整列ツール] オプションを選択します。
- 2 [垂直] パネルで、3 番目の [配置] オプション (関数 `align` の `Middle VerticalAlignment` オプションと同じ) を選択します。[水平] パネルで、最初の [整列] オプション (関数 `align` の `Left HorizontalAlignment` オプションと同じ) を選択します。
- 3 [適用] をクリックします。



次に示すように、ボタンが整列します。

メモ コンポーネントを整列するときに 1 つ覚えておくべきことは、関数 align がポイント単位を使用するのに対して、オブジェクト整列 GUI はピクセル単位を使用するということです。ただし、どちらの方法も、整列するコンポーネントの Units プロパティを変更しません。

Figure の [ツール] メニューから [整列] または [配置] オプションを選択して、いずれかの操作を直ちに実行することもできます。たとえば、[整列] メニュー項目から利用できる 6 つのオプションは次のとおりです。



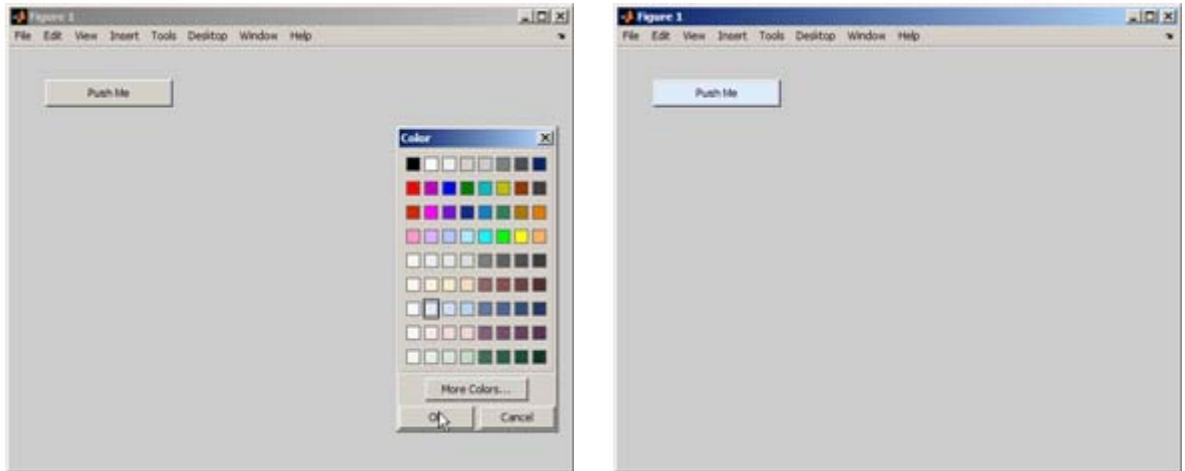
詳細は、“Align/Distribute Menu Options”を参照してください。

色の対話的な設定

色の例を確認せずに、Color、ForegroundColor、BackgroundColor、FontColor およびプロット オブジェクトのカラー プロパティに対して色を指定することは、難しいことがあります。関数 `uisetcolor` は、`set` を使用してコンポーネントの作成時または作成後にコンポーネントにプラグインできるカラー値を返す GUI を開きます。たとえば、次のステートメントは、色を選択できるカラー セレクター GUI を開きます。

```
set(object_handle, 'BackgroundColor', uisetcolor)
```

[OK] をクリックすると、set が直ちに割り当てる RGB 色ベクトルが返されます。オブジェクトが、指定された名前をもつプロパティをもたない場合や、指定されたプロパティが RGB 色値を受け入れない場合は、エラーが発生します。



位置と色の設定を 1 行のコードまたは 1 つの関数にまとめることができます。以下に例を示します。

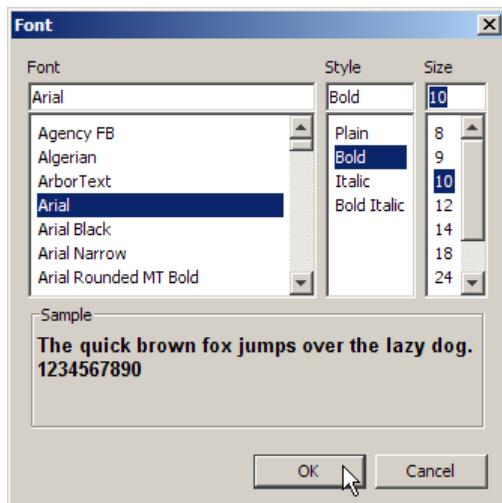
```
btn1 = uiicontrol('String', 'Button 1',...
    'Position', getrect,...
    'BackgroundColor', uisetcolor)
```

このステートメントを実行すると、最初に getrect が実行されて、rbbox を使用して位置を設定できます。マウス ボタンを離すと、uisetcolor GUI が開いて、背景色を指定できます。

フォントの特性の対話的な設定

uisetfont GUI では、システム上のすべてのフォントの特性にアクセスできます。これを用いて、テキストを表示する任意のコンポーネントについてフォントの特性を設定します。選択したプロパティ値を記述するデータを含む構造体が返されます。

```
FontData = uisetfont(object_handle)
```

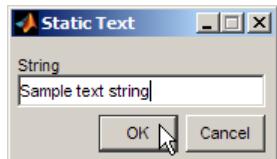


```
FontData =
  FontName: 'Arial'
  FontWeight: 'bold'
  FontAngle: 'normal'
  FontSize: 10
  FontUnits: 'points'
```

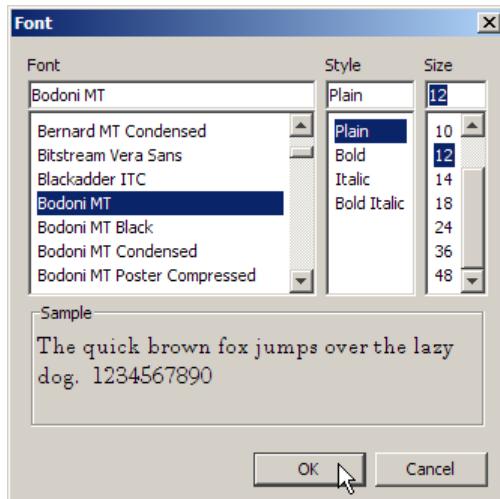
`uisetfont` は、一度にすべてのフォント特性を返します。構造体からフィールドを削除しない限り、それらのいずれかを省略できません。`String` プロパティをもつコンポーネントの作成時に、`uisetfont` を使用できます。テキスト文字列入力用の事前定義済み GUI である `inputdlg` を呼び出すことによって、同時に文字列自体を指定することもできます。スタティック テキストの作成、フォント プロパティの設定、フォントの配置を対話的に行う例を次に示します。

```
txt1 = uicontrol(...
  'Style', 'text',...
  'String', inputdlg('String', 'Static Text',...
  'uisetfont', 'Position', getrect)
```

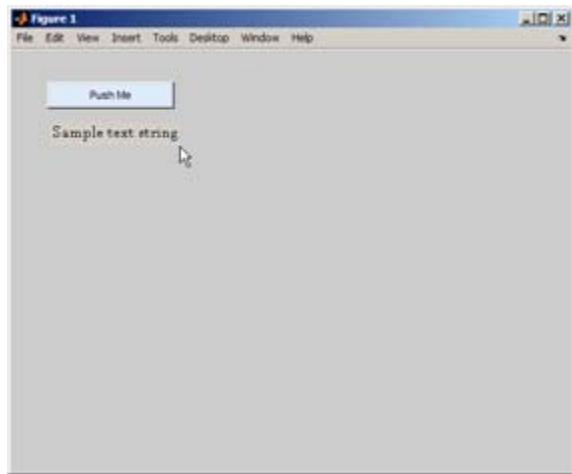
次に示すように、`inputdlg` ダイアログ ボックスが最初に表示されます。



文字列を入力し、[OK] をクリックすると、uisetfont ダイアログ ボックスが開いて、文字列を表示するためのフォントの特性を設定できます。



フォント、スタイルおよびサイズを指定し、[OK] をクリックすると、関数 getrect が実行されます（“位置ベクトルのスケッチ”（p.11-48）を参照）。テキストコンポーネントの長方形をドラッグ アウトし、マウス ボタンを離します。結果は、次の図のようになります。



コンポーネントプロパティを設定するためのコードの生成

前節で説明した手法は、コンポーネントのプロパティを対話的に設定するものです。しかし、まだプロパティ値を入力するためのコードを GUI コード ファイル内に作成する必要があります。プロパティ値を取得する通常の方法は、次のような `get` コマンドを入力することです。

```
get(object_handle, 'String')
ans =
    'Generate Data'
```

次に、`get` が返した文字列をコピーし、GUI コード ファイルに部分的に入力した `set` ステートメントに貼り付けます。

```
set(object_handle, 'String', 'Generate Data')
```

このプロセスを自動化するには、引数として指定したコンポーネント、プロパティおよび値に対して `set` コマンドを生成する補助関数を実行します。`setprop` という関数が、ドキュメンテーション例を含むヘルプ システム内のフォルダーにあります。“関数 `setprop` の表示と実行”(p.11-66)を参照してください。プロパティ値は文字列である必要はありません。関数 `setprop` は、次の MATLAB データ型を、生成した `set` コマンド内のテキストに正しく変換します。

- ・ 文字列
- ・ セル配列（他のセル配列を含むセル配列を除く）
- ・ 数値型（スカラー、ベクトル、行列）
- ・ オブジェクトハンドル
- ・ 関数ハンドル

これは、文字列、セル配列および関数ハンドルをエンコードするため、`setprop` は `ButtonDownFcn`、`Callback`、`CloseRequestFcn` などのコールバックを出力できます。この関数は、コンポーネントについて “set 文字列” を返します。コンポーネントハンドルとそのプロパティの 1 つの名前を指定します。そのプロパティの値も指定した場合、`setprop` は最初に set 文字列内に新しい値を設定してから返します。この関数のヘルプ テキストを次に示します。

```
function setstr = setprop(objhandle, property, value)
```

Generates a SET statement for a property of an object.

Copy the statement and paste it into your GUI code.

`setstr = setprop(objhandle)` opens a GUI listing all properties of `objhandle`, the handle of an HG object. When you choose one, a SET command is returned for the current value of the property you selected.

`setstr = setprop(objhandle,property)` returns a SET command for the current value of the property, or '' for an invalid property name.

`setstr = setprop(objhandle,property,value)` returns a SET command for the specified value of the property, or '' for an invalid property name or value. If property is empty, the listbox GUI opens to let you choose a property name.

The value can be a string, scalar, vector, cell array, HG handle or function handle. Properties consisting of structs or objects are disregarded and return '' (empty).

String プロパティに対する `setprop` の出力を次に示します。

```
setprop(object_handle,'String')
```

```
ans =
set(object_handle, 'String' , 'Generate Data' )
```

setprop を呼び出すときに文字列を割り当てることができます。複数行の文字列を作成するには、セル配列を使用します。以下に例を示します。

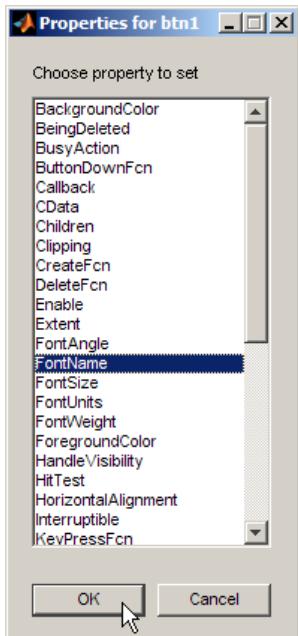
```
setstr = setprop(object_handle, 'string' , {'eggs' , 'milk' , 'flour' , 'sugar' })
setstr =
set(object_handle, 'String' , {'eggs' , 'milk' , 'flour' , 'sugar' })
```

プロパティ名を一意にしさえすれば、プロパティ名を短縮できます。たとえば、次の Position プロパティ set 文字列を取得してください。

```
setprop(object_handle, 'pos' )
ans =
set(object_handle, 'Position' , [20 20 60 20])
```

プロパティ名を省略した場合、オブジェクトのすべてのプロパティをリストする GUI が表示されます。

```
setprop(object_handle)
```



リストからプロパティ名を選択し、[OK] をクリックすると、`setprop` はそのプロパティの `set` 文字列とその現在の値を返します。

```
ans =  
set(btn1, 'FontName', 'MS Sans Serif')
```

`ans` について出力された値（または、値を割り当てた変数）をコマンド ウィンドウからコピーし、GUI コード ファイルに直接貼り付けます。

ヒント `setprop` を呼び出すときに空のプロパティ名に続けて値を指定できます。次に、この関数はその値を、GUI で選択されたプロパティに割り当てようします。割り当てたい値はわかっているが、プロパティ名のつづりがわからないときに、この方法で `setprop` を呼び出します。

一部のプロパティ値（たとえば、行列、セル配列など）は長いステートメントを生成することができます。長いコマンドを継続行（...）に分割して読みやすくすることができますが、分割は必須ではありません。

関数 `setprop` と関数 `setpos` を結合して、コンポーネントの位置を対話的に設定し、新しい位置について `set` コマンドを生成します。`object_handle` を実際のコンポーネントハンドルに置き換えて、次のコードを試してみてください。

```
setstr = setprop(object_handle, ' pos' , setpos(object_handle))
== Drag out a Position for object_handle
```

Figure 内に長方形をドラッグ アウトすると、次のステートメントのような結果が得られます(ただし、ユーザーの長方形を指定しています)。

```
setstr = set(hs2, ' Position' , [38 62 146 239])
```

関数 `setprop` の表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、`setprop.m` ドキュメンテーション例ファイルにアクセスできます。Web 上で、あるいは PDF でお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の関数のコードを変更しようとする場合には、最初に現在のフォルダーにそのコード ファイルのコピーを保存する必要があります。ファイルをそこに保存するには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下のリンクをクリックします。

- 1 ここをクリックして、`setprop` ファイルを現在のフォルダーにコピーしてください。
- 2 MATLAB エディター内にこのファイルを開くには、`edit setprop` と入力するか、または ここをクリックします。

関数を実行するか、コードを調べようとする場合、以下の手順に従います。

- 1 MATLAB パスにすべての GUI 例ファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 MATLAB エディター内にこのファイルを開くには、`edit setprop` と入力するか、または、ここをクリックします。(読み取り専用)

注意 変更した例を既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。これらは、現在のフォルダーあるいは別の作業フォルダーに保存してください。

GUI 開発ツールのまとめ

前節で説明したヒントおよびツールは、より短い時間で、タイプミスなどのエラーの可能性を減らして、GUI を設定するのに役立ちます。説明した手法は次のとおりです。

- ・ コンポーネントのレイアウトを調整するための、プロット編集モード。
- ・ プロパティ値の変更および読み込みのための、プロパティ エディターとプロパティ インスペクター。
- ・ コンテナー内にスクリーンピクセル単位でコンポーネントを配置するための、ユーティリティ関数 `getpixelposition` および `setpixelposition`。
- ・ 関数 `align` とその GUI である、プロット編集モードで機能するオブジェクト整列ツール。
- ・ 文字列、色およびフォントの特性プロパティをそれぞれ指定するための `inputdlg`、`uisetcolor`、`uisetfont` などの事前定義済みダイアログ ボックス。
- ・ コンポーネントを対話的に配置するための補助関数 `getrect`、`setpos` および `editpos`。これらの小さな関数を取得するには、“コンポーネントの位置の対話的な設定”(p.11-43)および“位置ベクトルのスケッチ”(p.11-48)でそれらのリストをコピーし、個々のコード ファイルに貼り付けます。これら 3 つの関数は例ファイルとして提供されていません。
- ・ `set` ステートメントとしてコピーし、GUI コードに貼り付けることができる `set` コマンド 文字列としてプロパティ名と値を取得するための補助関数 `setprop`。“コンポーネント プロパティを設定するためのコードの生成”(p.11-62)を参照してください。

これらのツールに慣れると、GUI 開発ワークフローを強化し、より少ない労力でより良い結果を得ることができます。

対話型ツールの使用方法の詳細は、次のドキュメンテーションを参照してください。

- ・ “The Property Editor”
- ・ “Accessing Object Properties with the Property Inspector”

- “Working in Plot Edit Mode”
- “Alignment Tool — Aligning and Distributing Objects”

タブの順序の設定

このセクションの内容…

“タブの動作仕様” (p.11-69)

“既定のタブ順序” (p.11-69)

“タブ順序の変更” (p.11-72)

タブの動作仕様

GUI のタブの順序は、ユーザーがキーボードの Tab キーを押す際に、GUI のコンポーネントがフォーカスされる順序です。フォーカスは、通常、境界または点線の境界で表されます。

タブ順序は、各親の子に対して別々に決定されます。たとえば、GUI の Figure の子のコンポーネントはそれ自身のタブ順序をもちます。各パネルまたはボタン グループの子のコンポーネントは、それ自身のタブ順序ももちます。

ある 1 つのレベルにあるコンポーネントにタブの順序を付ける場合、ユーザーはパネルまたはボタン グループに順序を付け、続いて、パネルまたはボタン グループに到達するレベルに戻る前に、パネルまたはボタン グループ内のコンポーネントに順序を付けます。たとえば、GUI の Figure に 3 つのプッシュボタンを含むパネルがあり、ユーザーがこのパネルにタブ順序を付ける場合、タブは、Figure に返る前に 3 つのプッシュボタンを一巡します。

メモ 座標軸とスタティック テキストコンポーネントにタブを付けることはできません。
どのコンポーネントがフォーカスをもっているかをプログラミングで決定できません。

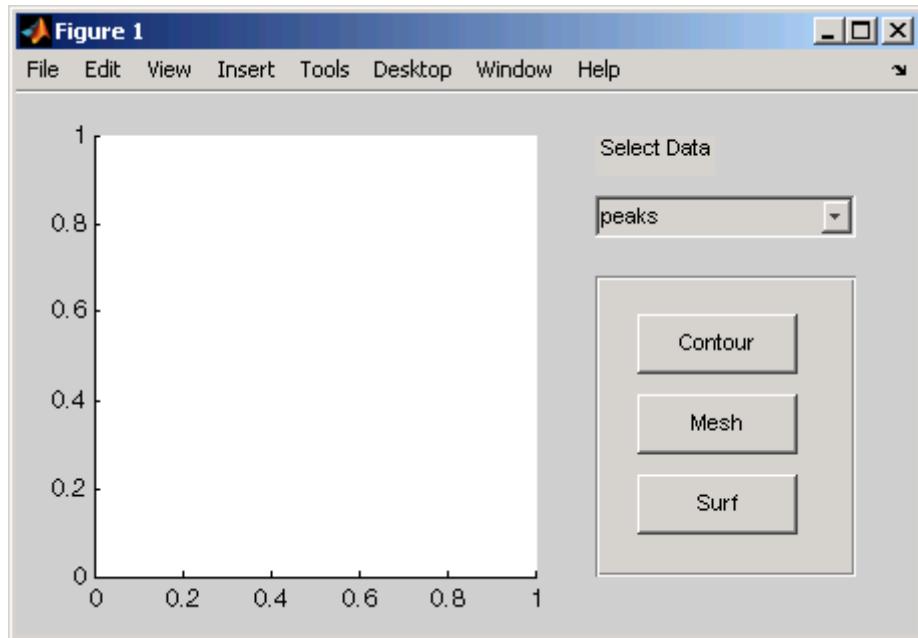
既定のタブ順序

各レベルに対する既定のタブの順序は、そのレベルでコンポーネントを作成する順序です。

以下のコードは、スタティック テキストラベルをもつポップアップメニュー、3 つのプッシュボタンをもつパネル、座標軸を含む GUI を作成します。

```
fh = figure('Position', [200 200 450 270]);
```

```
pmh = uicontrol(fh,'Style','popupmenu',...
    'String',{'peaks','membrane','sinc'},...
    'Position',[290 200 130 20]);
sth = uicontrol(fh,'Style','text','String','Select Data',...
    'Position',[290 230 60 20]);
ph = uipanel('Parent',fh,'Units','pixels',...
    'Position',[290 30 130 150]);
ah = axes('Parent',fh,'Units','pixels',...
    'Position',[40 30 220 220]);
bh1 = uicontrol(ph,'Style','pushbutton',...
    'String','Contour','Position',[20 20 80 30]);
bh2 = uicontrol(ph,'Style','pushbutton',...
    'String','Mesh','Position',[20 60 80 30]);
bh3 = uicontrol(ph,'Style','pushbutton',...
    'String','Surf','Position',[20 100 80 30]);
```



Children プロパティを取得することによって、Figure、パネル、またはボタン グループに対する、既定のタブ順序を取得できます。たとえば、次のステートメントのようになります。

```
ch = get(ph, 'Children')
```

ph は、パネルのハンドルです。このステートメントは、子と 3 つのプッシュボタンのハンドルを含むベクトルを返します。

```
ch =
4.0076
3.0076
2.0076
```

これらのハンドルは、次の表に示すようにプッシュボタンに相当します。

ハンドル	ハンドル変数	プッシュボタン
4.0076	bh3	Surf
3.0076	bh2	Mesh
2.0076	bh1	Contour

プッシュボタンの既定のタブの順序は、子のベクトルの順序の逆、[Contour] > [Mesh] > [Surf] です。

メモ 関数 `get` は、HandleVisibility プロパティを `on` に設定された、ハンドルが表示可能である子のみを返します。`allchild` を使用して、ハンドルが表示可能であるかどうかに関わらず、子を取得します。

例の GUI の `Figure`において、既定の順序は、パネルの `Contour`、`Mesh`、`Surf` プッシュボタンが（この順で）続く、ポップアップメニューです。座標軸コンポーネントやスタイルテキストコンポーネントにタブを付けることはできません。

`Mesh` プッシュボタンの前に、`Contour` プッシュボタンの作成に続きポップアップメニューを作成するようにコードを修正します。ここで、コードを実行して GUI と各コンポーネントのタブを作成します。このコードの変更は、既定のタブ順序を変更しません。これは、ポップアップメニューがプッシュボタンと同じ親をもたないためです。この `Figure` は、パネルとポップアップメニューの親です。

タブ順序の変更

関数 `uistack` を使用して、同じ親をもつコンポーネントのタブ順序を変更します。`uistack` の使いやすい構文は、次のとおりです。

```
uistack(h, stackopt, step)
```

`h` は、タブ順序を変更する必要があるコンポーネントのハンドルのベクトルです。

`stackopt` は、移動の方向を表します。これは、文字列 `up`、`down`、`top`、`bottom` のうちの 1 つでなければならず、ステートメントによって返される列ベクトルに関連して解釈されます。

```
ch = get(ph, 'Children')
```

```
ch =  
4. 0076  
3. 0076  
2. 0076
```

タブ順序が現在 `[Contour] > [Mesh] > [Surf]` である場合、次のステートメント

```
uistack(bh2, 'up', 1)
```

は、`bh2` (`Mesh`) を、子のベクトル内で 1 つ上の場所に配置し、タブ順序を `[Contour] > [Surf] > [Mesh]` に変更します。

```
ch = get(ph, 'Children')
```

は、つぎの出力をします。

```
ch =  
3. 0076  
4. 0076  
2. 0076
```

`step` は、変更されたレベル数です。既定値は 1 です。

メモ タブの順序は、コンポーネントを積み重ねる順序にも影響します。コンポーネントが重なる場合、子の順序が下位に現れるコンポーネントは、より上位に現れるコンポーネントの上に描かれます。例のプッシュ ボタンが重なった場合、[Contour] プッシュ ボタンが上になります。

メニューの作成

このセクションの内容...

“メニュー バーにメニューを追加する” (p.11-74)

“コンテキストメニューの追加” (p.11-80)

メニュー バーにメニューを追加する

関数 `uimenu` を使用して、ユーザー GUI にメニュー バー メニューを追加します。`uimenu` に対する構文は、次のようにになります。

```
mh = uimenu(parent, 'PropertyName', PropertyValue, ...)
```

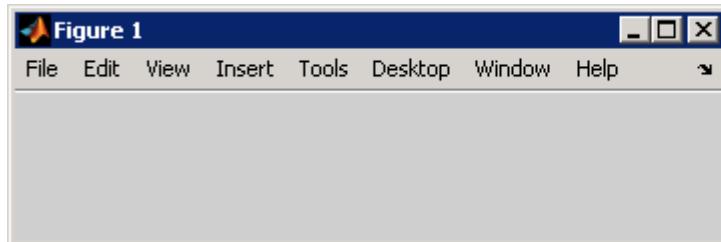
`mh` は、結果のメニューまたはメニュー項目のハンドルです。他の有効な構文は、`uimenu` のリファレンス ページを参照してください。

以下のトピックスは、MATLAB 標準メニュー バーのメニュー、一般に使用されるメニュー プロパティについて述べ、いくつかの簡単な例を提供します。

- ・ “標準のメニュー バー メニューの表示” (p.11-74)
- ・ “一般に利用するプロパティ” (p.11-75)
- ・ “メニューが Figure のドックにどのように影響するか” (p.11-76)
- ・ “メニュー バー メニュー” (p.11-78)

標準のメニュー バー メニューの表示

標準のメニュー バー メニューの表示は、オプションです。



Standard menu bar menus

標準メニュー バーのメニューを使用すると、ユーザーが作成するメニューが追加されます。標準メニュー バーのメニューを表示しないように選択すると、メニュー バーはユーザーが作成するメニューのみを表示します。標準のメニューを表示せず、メニューを作成しないと、メニュー バーは表示されません。

Figure のMenuBar プロパティを使用して、前図に示した MATLAB 標準メニュー バーを表示または非表示にします。MenuBar を figure (既定) に設定して、標準のメニューを表示します。それらを表示しないためには、MenuBar を none と設定します。

```
set(fh,'MenuBar','figure'); % Display standard menu bar menus.  
set(fh,'MenuBar','none'); % Hide standard menu bar menus.
```

これらのステートメントにおいて、fh は Figure のハンドルです。

一般に利用するプロパティ

以下の表では、メニュー バー メニューを記述するために必要となる、最も一般に利用されるプロパティを示します。

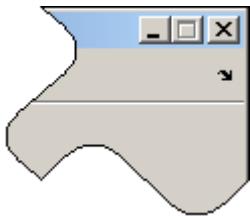
プロパティ	値	説明
Accelerator	アルファベット文字	キーボードと同じ。サブメニューをもたないメニュー項目に対して利用可能
Checked	off、on。既定値は off です。	メニュー チェック インジケータ
Enable	on、off。既定値は on です。	メニュー項目が選択できるかどうかをコントロールします。off に設定されると、メニュー ラベルはグレー表示になります。
HandleVisibility	on、off。既定値は on です。	オブジェクトのハンドルが、このオブジェクトの親に対する子のリストで表示されるかどうかを決めます。メニューの場合、HandleVisibility を off に設定して、メニューに意図されていない操作からメニューを保護します。

プロパティ	値	説明
Label	文字列	メニュー ラベル。 ラベルに文字 & を表示するには、文字列で 2 つの & 文字を使用します。 remove、default、factory (大文字と小文字の区別あり) は予約語です。これらのいずれかをラベルとして使用するには、文字列の先頭に バックスラッシュ (\) を付けます。たとえば、\\$remove は remove を与えます。
Position	スカラー。既定値は 1 です。	メニュー内のメニュー項目の位置
Separator	off、on。既定値は off です。	区切りのラインモード

プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB ドキュメンテーションの「Uimenu プロパティ」のリファレンス ページを参照してください。GUI の動作をコントロールするために必要なプロパティの詳細は、章 12, “GUI のプログラミング” を参照してください。

メニューが Figure のドックにどのように影響するか

メニュー バーまたはツールバーをカスタマイズすると、Figure の WindowStyle がその設定と衝突しない限り、DockControls を適切に設定することで、GUI のドックのコントロールを表示または非表示とすることができます。GUI に対してメニューは必要ありませんが、ユーザーが GUI をドックしたりアンドックしたりできるようにするには、メニュー バーまたはツールバーを含む必要があります。これは、以下の図が示すように、ドックがドックのためのアイコンでコントロールされているためです。このアイコンは、メニュー バーまたはツールバーの右上隅付近の小さな曲がった矢印です。



標準メニュー バーをもつ Figure ウィンドウにも、[デスクトップ] メニューがあります。ユーザーはこのメニューから Figure ウィンドウをドックしたりアンドックすることができます。

ドックの矢印と [デスクトップ] > [Figure をドック] メニュー項目を表示するには、Figure のプロパティ DockControls は 'on' に設定しなければなりません。これは、プロパティ インスペクターで設定できます。さらに、ドックのコントロールを表示するには、Figure のMenuBar および/またはToolBar プロパティを 'on' に設定する必要があります。

Figure プロパティ WindowStyle もドックの動作に影響します。既定値は 'normal' ですが、「docked」に変更した場合、以下を適用します。

- ・ GUI は、それが実行しているときデスクトップにドックされた状態で開きます。
- ・ DockControls プロパティは [on] に設定されますが、WindowStyle が [docked] に設定されるまでは、オフにすることはできません。
- ・ WindowStyle 'docked' で作成された GUI をアンドックすると、Figure がメニュー バーまたはツールバー（標準またはカスタマイズされた）表示しない限り、ドックのための矢印は表示されません。ドックのための矢印がないと、ユーザーはデスクトップからアンドックできますが、再びドックし直すことはできません。

要約のために、DockControls プロパティが Figure の WindowStyle プロパティと衝突しない限り、DockControls プロパティを用いてドッキング制御を表示できます。

メモ モーダルなダイアログである GUI (WindowStyle を 'modal' に設定した Figure) は、メニュー バー、ツールバー、ドックのコントロールをもつことができません。

詳細は、「Figure のプロパティ」のリファレンス ページで DockControls、MenuBar、ToolBar、WindowStyle プロパティの説明を参照してください。

メニュー バー メニュー

次のステートメントは、2つのメニュー項目をもつメニュー バー メニューを作成します。

```
mh = uimenu(fh, 'Label', 'My menu');
eh1 = uimenu(mh, 'Label', 'Item 1');
eh2 = uimenu(mh, 'Label', 'Item 2', 'Checked', 'on');
```

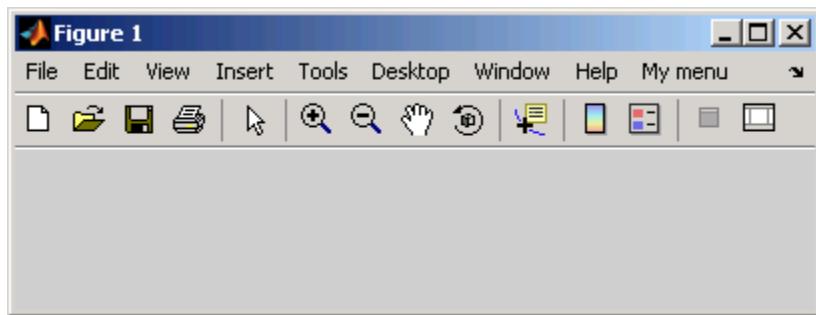
`fh` は、親の `Figure` のハンドルです。

`mh` は、親のメニューのハンドルです。

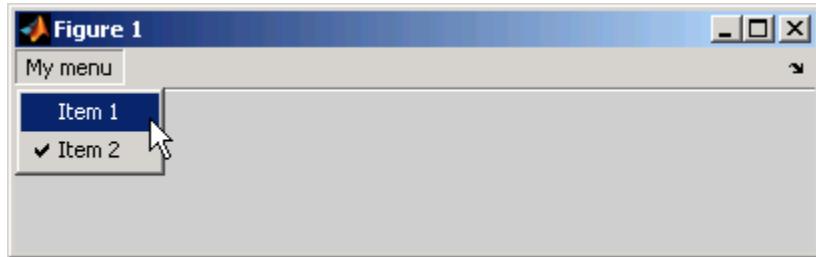
`Label` プロパティは、メニューに現われるテキストを指定します。

`Checked` プロパティは、メニューが作成されるときに、この項目の隣にチェックが表示されるように指定します。

ユーザー GUI が標準のメニュー バーを表示すると、これに新しいメニューが追加されます。

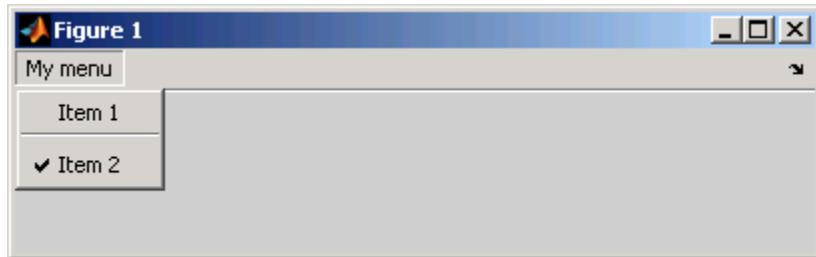


ユーザー GUI が標準のメニュー バーを表示しない場合、メニュー バーが存在しないときには、MATLAB はメニュー バーを作成し、メニュー バーにメニューを追加します。



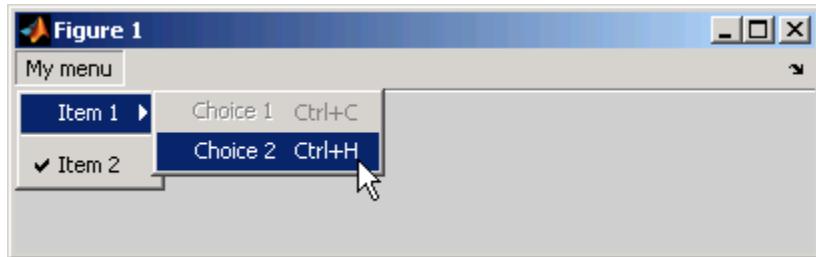
次のステートメントは、2番目のメニュー項目の前に境界線を追加します。

```
set(eh2,'Separator','on');
```



次のステートメントは、Item 1 に 2つのメニュー サブ項目を追加し、各サブ項目にキーボード アクセラレータを割り当て、最初のサブ項目を無効にします。

```
seh1 = uimenu(eh1,'Label','Choice 1','Accelerator','C',...
    'Enable','off');
seh2 = uimenu(eh1,'Label','Choice 2','Accelerator','H');
```



Accelerator プロパティは、メニュー項目にキーボード アクセラレータを追加します。アクセラレータはシステム上で他の目的で使用されるものもあり、結果として他のアクションが起こることもあります。

Enable プロパティは最初のサブ項目 Choice 1 を無効にするので、ユーザーはメニューが最初に作成されるときに、メニューを選択できません。項目は、グレー表示になります。

メモ ユーザーがすべてのメニュー項目を作成した後、次のステートメントを実行して、その HandleVisibility プロパティを off に設定します。

```
menuhandles = findall(figurehandle, ' type' , ' uimenu' );
set(menuhandles, ' HandleVisibility' , ' off' )
```

メニュー項目のプログラミングについての詳細は、“メニュー項目のプログラミング”(p.12-34)を参照してください。

コンテキストメニューの追加

ユーザーが Figure または GUI コンポーネントにおいて右クリックすると、コンテキストメニューが現われます。以下の手順に従い、ユーザー GUI にコンテキストメニューを追加します。

- 1 関数 `uicontextmenu` を使用して、コンテキストメニュー オブジェクトを作成します。
- 2 関数 `uimenu` を使用して、コンテキストメニューにメニュー項目を追加します。
- 3 オブジェクトの `UIContextMenu` プロパティを使用して、コンテキストメニューとグラフィックス オブジェクトを関連付けます。

以下のトピックスでは、一般に使用されるコンテキストメニュー プロパティを述べ、次の各手順を説明します。

- ・ “一般に利用するプロパティ”(p.11-81)
- ・ “コンテキストメニュー オブジェクトの作成”(p.11-82)
- ・ “コンテキストメニューへのメニュー項目の追加”(p.11-83)

- “コンテキストメニューとグラフィックス オブジェクトとの関連付け” (p.11-84)
- “コンテキストメニューの強制表示” (p.11-85)

一般に利用するプロパティ

以下の表では、コンテキストメニュー オブジェクトを記述するために必要となる、最も一般に利用されるプロパティを示します。以下のプロパティは、メニュー オブジェクトに対してのみ適用され、個々のメニュー項目には適用されません。

プロパティ	値	説明
HandleVisibility	on、off。既定値は on です。	オブジェクトのハンドルが、このオブジェクトの親に対する子のリストで表示されるかどうかを決めます。メニューの場合、HandleVisibility を off に設定して、メニューに意図されていない操作からメニューを保護します。
Parent	Figure ハンドル	コンテキストメニューの親の Figure のハンドル。
Position	2 要素ベクトル。[左端からの距離、下端からの距離]。既定値は [0 0] です。	親 Figure の左下隅から、コンテキストメニューの左上隅までの距離。このプロパティは、コンテキストメニューの Visible プロパティを、プログラミングで on に設定するときに限り、使用されます。
Visible	off、on。既定値は off です。	<ul style="list-style-type: none"> コンテキストメニューが現在表示されているかどうかを示します。コンテキストメニューが表示されているとき、プロパティ値は on です。コンテキストメニューが表示されていないとき、その値は off です。 値を on に設定すると、コンテキストメニューの位置をオーカスします。off に設定すると、削除すべきコンテキストメニューをオーカスします。Position プロパティは、コンテキストメニューが表示される位置を決めます。

プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB 関数リファレンス ドキュメンテーションの「Uicontextmenu プロパティ」のリファレンス ページを参照してください。GUI の動作をコントロールするために必要なプロパティは、章 12、“GUI のプログラミング”に説明します。

コンテキストメニュー オブジェクトの作成

関数 `uicontextmenu` を使用して、コンテキストメニュー オブジェクトを作成します。その構文は次のとおりです。

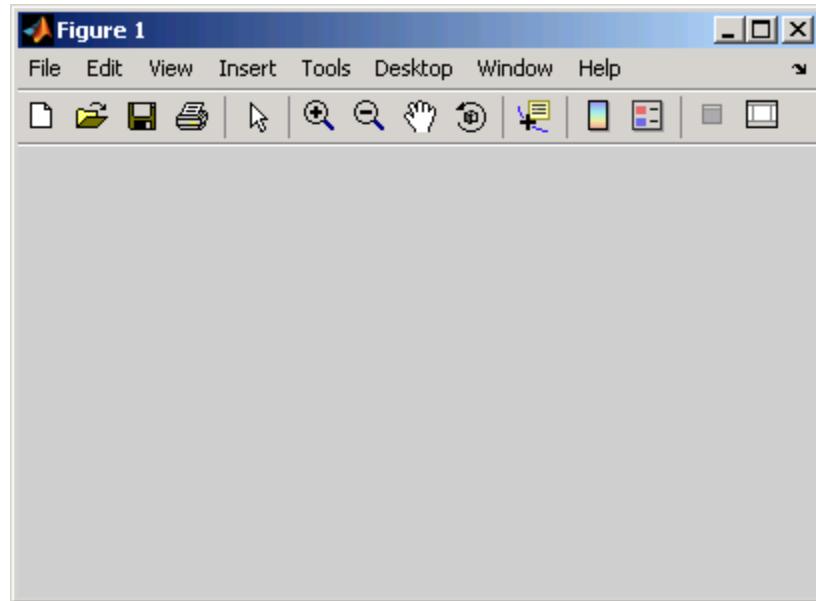
```
handle = uicontextmenu('PropertyName', PropertyValue, ...)
```

コンテキストメニューの親は、常に `Figure` でなければなりません。コンテキストメニュー `Parent` プロパティを使用して、その親を指定します。`Parent` を指定しない場合、その親は、ルートの `CurrentFigure` プロパティによって指定されるように、現在の `Figure` です。

以下のコードは、`Figure` を作成し、また親が `Figure` であるコンテキストメニューを作成します。

```
fh = figure('Position', [300 300 400 225]);
cmenu = uicontextmenu('Parent', fh, 'Position', [10 215]);
```

この時点で、`Figure` は表示できますが、メニューは表示できません。



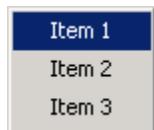
メモ “コンテキストメニューの強制表示” (p.11-85) は、Position プロパティの使用を説明します。

コンテキストメニューへのメニュー項目の追加

関数 `uimenu` を使用して、コンテキストメニューに項目を追加します。項目を追加した順序で、メニュー上に表示されます。以下のコードは、上で作成されたコンテキストメニューに 3 つの項目を追加します。

```
mh1 = uimenu(cmenu, 'Label', 'Item 1');
mh2 = uimenu(cmenu, 'Label', 'Item 2');
mh3 = uimenu(cmenu, 'Label', 'Item 3');
```

コンテキストメニューを見ると、次のように見えます。



コンテキストメニュー項目を定義する際、`Checked`、または `Separator` などの「Uimenu プロパティ」を使用できます。メニュー項目の作成のために `uimenu` の利用についての詳細は、`uimenu` リファレンス ページと“メニュー バーにメニューを追加する”(p.11-74)を参照してください。コンテキストメニューは、`Accelerator` プロパティをもたないことに注意してください。

メモ コンテキストメニューとそのすべてのメニュー項目を作成した後、次のステートメントを実行して、その `HandleVisibility` プロパティを `off` に設定します。

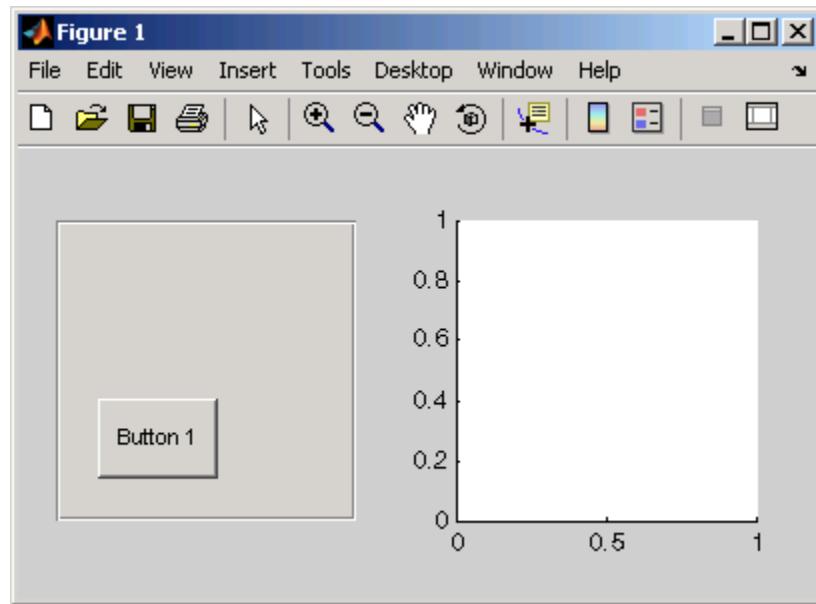
```
cmenuhandles = findall(figurehandle, 'type', 'uicontextmenu');
set(cmenuhandles, 'HandleVisibility', 'off')
menuitemhandles = findall(cmenuhandles, 'type', 'uimenu');
set(menuitemhandles, 'HandleVisibility', 'off')
```

コンテキストメニューとグラフィックス オブジェクトとの関連付け

コンテキストメニューを Figure 自身と UIContextMenu プロパティをもつすべてのコンポーネントと関連させることができます。これは、座標軸、パネル、ボタン グループ、すべてのユーザー インターフェイス コントロール (uicontrol) を含みます。

次のコードは、Figure にパネルと座標軸を追加します。パネルは、1 つのプッシュ ボタンを含みます。

```
ph = uipanel('Parent', fh, 'Units', 'pixels', ...
    'Position', [20 40 150 150]);
bh1 = uicontrol(ph, 'String', 'Button 1', ...
    'Position', [20 20 60 40]);
ah = axes('Parent', fh, 'Units', 'pixels', ...
    'Position', [220 40 150 150]);
```

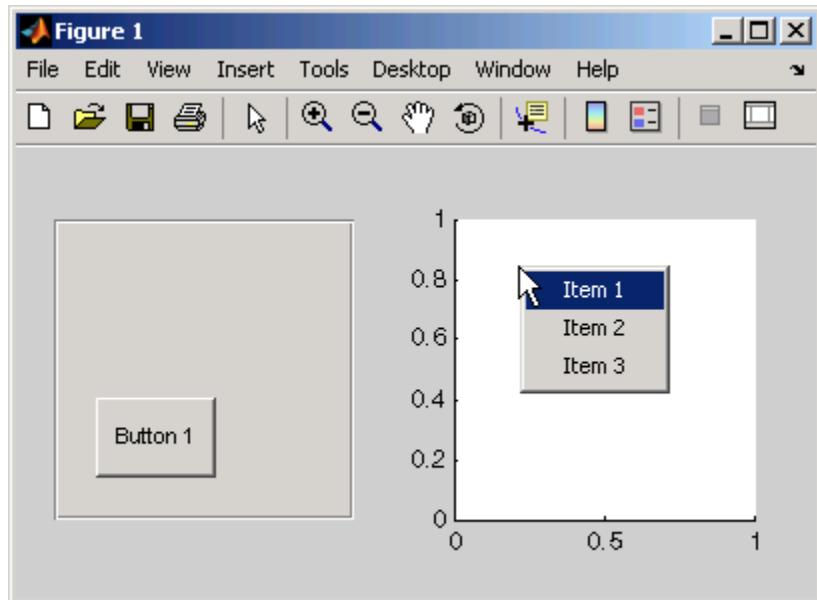


次のコードは、Figure の UIContextMenu プロパティと、コンテキストメニューのハンドル cmenu に座標軸を設定することによって、コンテキストメニューを Figure と座標軸に関連させます。

```
set(fh, 'UIContextMenu', cmenu); % Figure
```

```
set(ah,'UIContextMenu',cmenu); % Axes
```

Figure または座標軸上を右クリックします。コンテキストメニューが表示され、ユーザーがクリックした位置が左上隅になります。パネルまたはそのプッシュボタンを右クリックします。コンテキストメニューは表示されません。

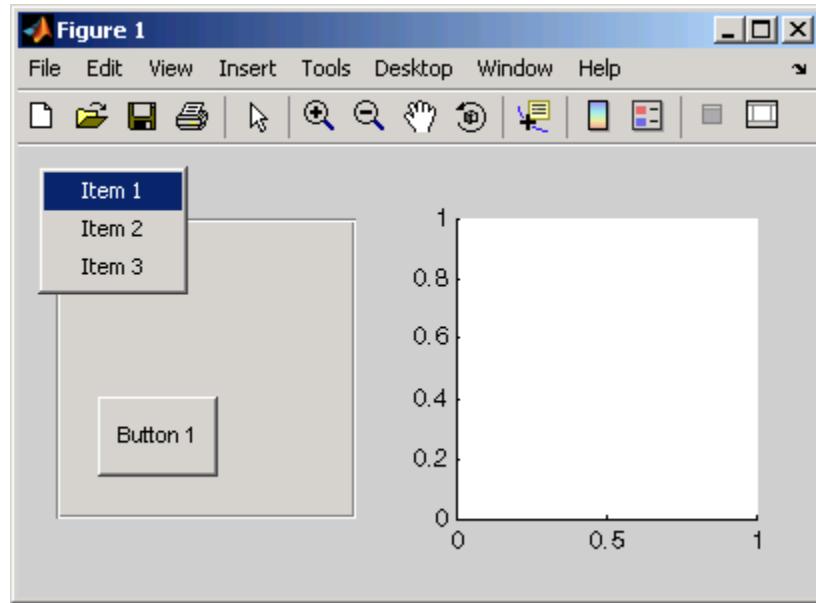


コンテキストメニューの強制表示

コンテキストメニューの Visible プロパティを on に設定すると、ユーザー アクションなしで、コンテキストメニューは、Position プロパティで指定される位置に表示されます。この例で、コンテキストメニューの Position プロパティは [10 215] です。

```
set(cm, 'Visible', 'on');
```

このとき、コンテキストメニューは、Figure の左端から 10 ピクセルで、下端から 215 ピクセルの位置に表示されます。



コンテキストメニューの `Visible` プロパティを `off` に設定したり、あるいはユーザーがコンテキストメニューの外で GUI をクリックすると、コンテキストメニューが消えます。

ツールバーの作成

このセクションの内容…

“関数 uitable の利用” (p.11-87)

“一般に利用するプロパティ” (p.11-87)

“ツールバー” (p.11-88)

“標準ツールバーの表示と修正” (p.11-91)

関数 uitable の利用

関数 `uitable` を使用して、GUI にカスタム ツールバーを追加します。関数 `uipushtool` と関数 `uitoggletool` を使用して、プッシュ ツールとトグル ツールをツールバーに追加します。プッシュ ツールは、プッシュ ボタンとして機能します。トグル ツールは、トグル ボタンとして機能します。標準のツールバーまたはカスタム ツールバーに、プッシュ ツールとトグル ツールを追加できます。

関数 `uitable`、`uipushtool`、`uitoggletool` に対する構文は、以下を含みます。

```
tbh = uitable(h, 'PropertyName', PropertyValue, ...)  
pth = uipushtool(h, 'PropertyName', PropertyValue, ...)  
tth = uitoggletool(h, 'PropertyName', PropertyValue, ...)
```

`tbh`、`pth`、`tth` は、それぞれツールバー、プッシュ ツール、トグル ツールのハンドルです。他の有効な構文は、`uitable`、`uipushtool`、`uitoggletool` のリファレンス ページを参照してください。

以下のトピックスは、ツールバーとツールバー ツールの一般に使用されるプロパティを述べ、簡単な例を提供して MATLAB 標準ツールバーの利用について説明します。

一般に利用するプロパティ

以下の表では、ツールバーとそのツールを記述するために必要となる最も一般的に利用するプロパティを示します。

プロパティ	値	説明
CData	0.0 と 1.0 の間の値をもつ 3 次元配列	プッシュ ボタンまたはトグル ボタンに表示されるトゥルーカラーアイメージを定義する RGB 値の $n \times m \times 3$ 配列。
HandleVisibility	on、off。既定値は on です。	オブジェクトのハンドルが、このオブジェクトの親に対する子のリストで表示されるかどうかを決めます。ツールバーとそれらのツールに対して、HandleVisibility を off に設定して、意図されていない操作から保護します。
Separator	off、on。既定値は off です。	プッシュ ツールまたはトグル ツールの左に境界線を描きます。
State	off、on。既定値は off です。	トグル ツールの状態。on は、下つまり押された位置です。off は、上つまり上がった位置です。
TooltipString	文字列	プッシュ ツールまたはトグル ツールに関連するツールチップのテキスト。

プロパティの完全なリスト、および表にリストされたプロパティの詳細は、MATLAB 関数リファレンスドキュメンテーションの「Uitoolbar プロパティ」、「Uipushtool プロパティ」、「Uitoggletool プロパティ」のリファレンス ページを参照してください。GUI の動作をコントロールするために必要なプロパティは、章 12, “GUI のプログラミング”に説明します。

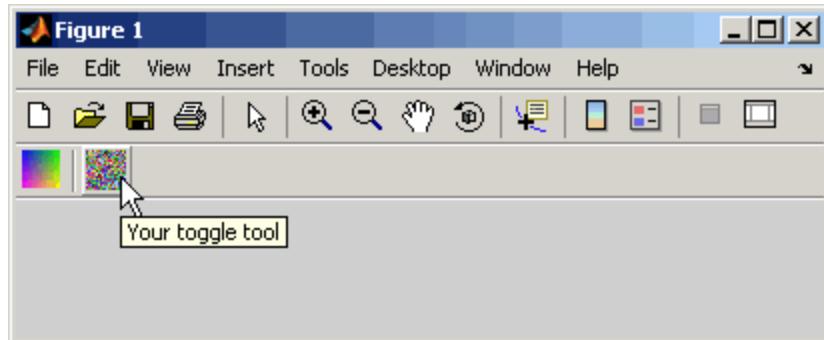
ツールバー

次のステートメントは、Figure にツールバーを追加し、ツールバーにプッシュ ツールとトグル ツールを追加します。既定では、ツールが作成される順序で左から右へ、ツールバーに追加されます。

```
% Create the toolbar
th = uiboolbar(fh);

% Add a push tool to the toolbar
```

```
a = [.20:.05:.95]
img1(:,:,1) = repmat(a,16,1)'
img1(:,:,2) = repmat(a,16,1);
img1(:,:,3) = repmat(flipdim(a,2),16,1);
pth = uipushtool(th,'CData',img1,...
    'TooltipString','My push tool',...
    'HandleVisibility','off')
% Add a toggle tool to the toolbar
img2 = rand(16,16,3);
tth = uitoggletool(th,'CData',img2,'Separator','on',...
    'TooltipString','Your toggle tool',...
    'HandleVisibility','off')
```



fh は、親の Figure のハンドルです。

th は、親のツールバーのハンドルです。

CData は、0 と 1 の間の値の $16 \times 16 \times 3$ 配列です。これは、ツール上に表示されるツールーカラー イメージを定義します。ユーザーのイメージにおいて、いずれかの次元が 16 ピクセルより大きいと、イメージが切り取られたり、あるいは他の望ましくない効果の原因になります。配列が切り取られると、配列の中央の 16×16 の部分のみが使用されます。

メモ“アイコン エディター”(p.15-60)に述べるアイコン エディターを用いてユーザー独自のアイコンを作成します。行列 X と対応するカラーマップ、すなわち、(X , MAP) イメージから RGB (トゥルーカラー) 形式への変換の詳細は、ind2rgb のリファレンス ページを参照してください。

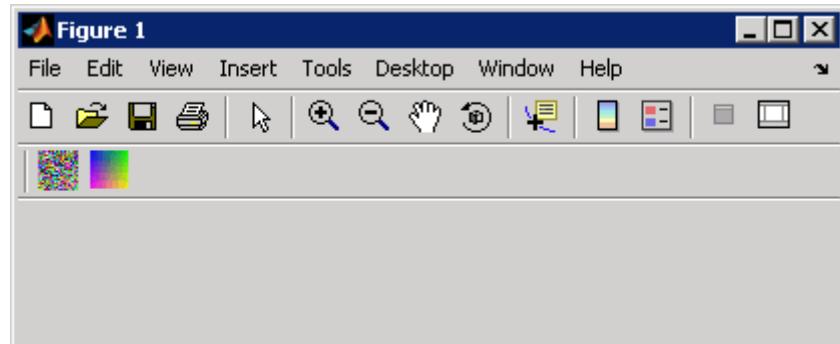
`TooltipString` は、プッシュ ツールとトグル ツールに対して、それぞれ、`My push tool` と `Your toggle tool` としてツールチップを指定します。

この例では、トグル ツールの `Separator` プロパティを `on` に設定すると、トグル ツールの左に境界線を作成します。

親ツールバーの子ベクトルを修正することによって、ツールの順序を変更できます。次のコードにより、ツールの順序を逆にすることができます。

```
oldOrder = allchild(th);
newOrder = flipud(oldOrder);
set(th,'Children',newOrder);
```

このコードは、`Children` プロパティは列ベクトルなので、`flipud` を使用します。



関数 `delete` を使用して、ツールバーからツールを削除します。次のステートメントは、ツールバーから トグル ツールを削除します。トグル ツールのハンドルは `tth` です。

```
delete(tth)
```

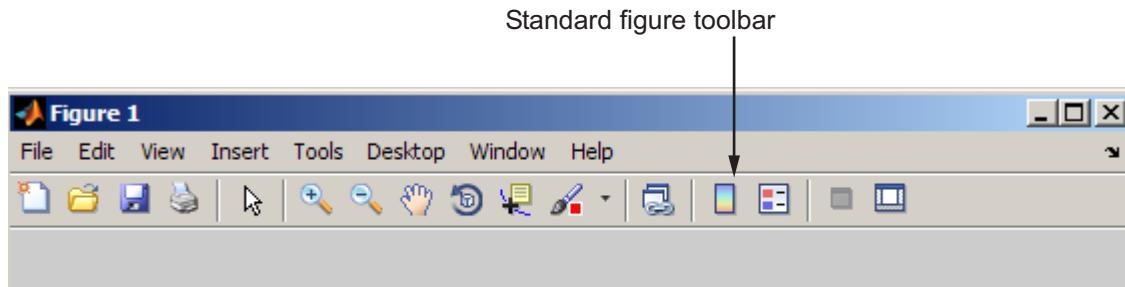
必要に応じて、関数 `findall` を用いることにより、特定のツールバー上のツールのハンドルを決めることができます。

メモ ユーザーがツールバーとそのツールを作成した後、次のようなステートメントを実行して、その HandleVisibility プロパティを off に設定します。

```
set(toolbarhandle, 'HandleVisibility', 'off')
toolhandles = get(toolbarhandle, 'Children');
set(toolhandles, 'HandleVisibility', 'off')
```

標準ツールバーの表示と修正

GUI に MATLAB 標準ツールバーを表示するかどうかを選択できます。標準ツールバーにツールを追加したり、削除することもできます。



標準ツールバーの表示

Figure の Toolbar プロパティを使用して、MATLAB 標準ツールバーを表示または非表示にできます。Toolbar を figure に設定して、標準ツールバーを表示します。非表示にするには、Toolbar を none と設定します。

```
set(fh, 'Toolbar', 'figure'); % Display the standard toolbar
set(fh, 'Toolbar', 'none'); % Hide the standard toolbar
```

これらのステートメントにおいて、fh は Figure のハンドルです。

Figure の Toolbar の既定値は、auto です。この設定は、Figure ツールバーを表示しますが、Figure にユーザー インターフェイス コントロール (uicontrol) を追加すると Figure ツールバーを削除します。

標準ツールバーの修正

標準ツールバーのハンドルがあると、ツールの追加、削除、順序変更ができます。

カスタムツールバーにツールを追加するのと同じようにツールを追加します。次のコードは、MATLAB 標準ツールバーのハンドルを取得し、“ツールバー”(p.11-88)に定義されたものと似たトグルツールをツールバーに追加します。fh は、Figure のハンドルです。

```
tbh = findall(fh, 'Type', 'uitoolbar');
tth = uitoggletool(tbh, 'CData', rand(20, 20, 3), ...
    'Separator', 'on', ...
    'HandleVisibility', 'off');
```



標準ツールバーからツールを削除するには、削除するツールのハンドルを決定し、関数 `delete` を使用して削除します。次のコードは、上記の標準ツールバーに追加されたトグルツールを削除します。

```
delete(tth)
```

必要に応じて、関数 `findall` を用いることにより、標準ツールバー上のツールのハンドルを決めることができます。

クロスプラットフォーム互換性のための設計

このセクションの内容…

“既定のシステムフォント” (p.11-93)

“標準背景色” (p.11-94)

“クロスプラットフォーム互換の Units” (p.11-95)

既定のシステムフォント

uicontrol はデフォルトで、使用しているプラットフォームの既定フォントを使用します。たとえば、PC 上でユーザー GUI を表示するとき、uicontrol は MS San Serif を使用します。ユーザー GUI が、異なるプラットフォーム上で動作するとき、コンピューターの既定フォントを使用します。このため、同じプラットフォーム上でユーザー GUI と他のアプリケーション GUI に関係して、同じ外見を与えます。

FontName プロパティを名前の付いたフォントに設定した後、既定値に戻したい場合、プロパティを文字列 default に設定できます。そうすると、MATLAB は実行時にシステム既定値を使用します。

このプロパティを設定するために、set コマンドを使用できます。たとえば、ユーザー GUI に ハンドル pbh1 をもつプッシュ ボタンがあると、次のステートメント

```
set(pbh1, 'FontName', 'default')
```

は、システム既定値を使用するように FontName プロパティを設定します。

固定幅フォントの指定

uicontrol に固定幅フォントを使用したい場合、その FontName プロパティを文字列 fixedwidth に設定してください。この特別な識別子により、ターゲット プラットフォームに対して、ユーザー GUI が標準の固定幅を使用することを保証します。

ルートの 「FixedWidthFontName」 プロパティを調べることにより、与えられたプラットフォーム上使用される固定幅フォント名を検出することができます。

```
get(0, 'FixedWidthFontName')
```

特定のフォント名の使用

FontName プロパティに対して、(Times あるいは Courierのような) 実際のフォント名を指定することができます。しかし、そのように指定すると、ユーザー GUI が異なるコンピューター上で動くとき、期待通りの外見にならない可能性があります。ターゲットコンピューターが指定のフォントをもたない場合、ユーザー GUI が良く見えなかつたり、そのシステム上の GUI に使用される標準フォントではない、他のフォントに置き換えられることもあります。また、同じ名前でフォントの異なるバージョンのものは、与えられた文字の組に対して、サイズ要求が異なることもあります。

標準背景色

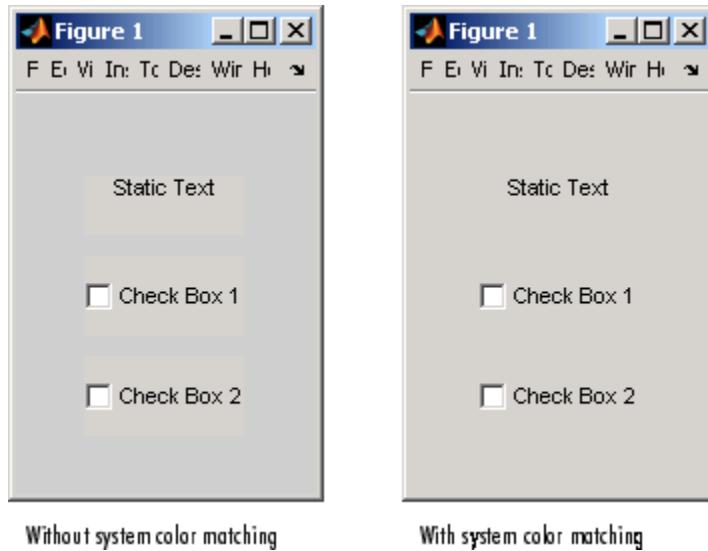
MATLAB は、GUI が実行しているシステムの標準のシステム背景色を、既定のコンポーネント背景色として使用します。この色はコンピューター システムが異なると変わります。たとえば、PC の標準的なグレーの影は UNIX システムのものとは異なり、既定の GUI 背景色とは一致しないことがあります。

GUI の背景色を、既定のコンポーネント背景色とマッチさせることができます。次のステートメントは、既定のコンポーネント背景色を取得し、それを Figure に割り当てます。

```
defaultBackground = get(0, 'defaultUicontrolBackgroundColor');
set(figureHandle, 'Color', defaultBackground)
```

Figure の Color プロパティは、Figure の背景色を指定します。

次の図は、システム カラー マッチングがある場合とない場合の結果を示します。



クロスプラットフォーム互換の Units

クロスプラットフォーム互換の GUI は、スクリーン サイズと解像度が異なるコンピューター上で、正しく見える必要があります。ピクセルのサイズは異なるコンピューター上で変わることがあるので、Figure の Units の既定値である pixels を使用すると、すべてのプラットフォーム上で同じに見える GUI を生成することはできません。Figure とコンポーネントの Units プロパティを適切に設定すると、GUI を異なるプラットフォームにどれほどよく移植できるかがわかります。

Units とサイズ変更の動作

単位の選択は、GUI のサイズ変更動作にも関連します。Figure の Resize と ResizeFcn プロパティは、ユーザー GUI のサイズ変更動作をコントロールします。

Resize は、マウスで Figure ウィンドウをサイズ変更できるかどうかを決定します。on と設定すると、ウィンドウをサイズ変更できることを意味し、off はサイズ変更できないことを意味します。Resize を off に設定すると、Figure ウィンドウは、Figure ウィンドウがサイズ変更できないことを示すサイズ変更コントロールを表示しません。

ResizeFcn は、GUI のサイズ変更動作をカスタマイズでき、Resize を on に設定する場合に限り、有効です。ResizeFcn は、ユーザーが GUI をサイズ変更するときに実行される、ユーザーが記述するコールバックのハンドルです。これは、GUI 内のすべてのコンポーネントのサイズ変更をコントロールします。サイズ変更動作の例は、Figure の ResizeFcn プロパティの説明を参照してください。

次の表は、ユーザー GUI のサイズ変更動作に基づき、適切な Units の設定を示します。これらの設定によって、GUI が異なるコンピューターに表示し、GUI がサイズ変更されるときに、ユーザー GUI のサイズとコンポーネントの相対的な間隔を自動的に調整できます。

コンポーネント	既定の単位	Resize = on ResizeFcn = []	Resize = off
Figure	pixels	characters	characters
プッシュ ボタン、スライダー、エディット テキスト コンポーネントなどのユーザー インターフェイス コントロール (uicontrol)	pixels	normalized	characters
座標軸	normalized	normalized	characters
パネル	normalized	normalized	characters
ボタン グループ	normalized	normalized	characters

メモ 上記の表に示される既定値は、GUIDE の既定値とは同じではありません。GUIDE の既定値は、GUIDE [リサイズ アクション] オプションに依存し、表の最後の 2 列に示すものと同じです。

いくつかの Units 設定について

Characters. character units は、既定のシステム フォントからの文字によって定義されています。character unit の幅は、システム フォントにおいて文字 x の幅に等しくなります。character unit の高さは、テキストの 2 行のベースライン間の距離です。character units が正方形でないことに注意してください。

Normalized. normalized units は、親のサイズのパーセンテージを表します。normalized units の値は、0 と 1 の間にあります。たとえば、パネルがプッシュボタンを含み、ボタンの units の設定が normalized の場合、プッシュボタンの Position の設定 [2 .2 .6 .25] は、プッシュボタンの左端が、パネルの左端からパネル幅の 20 パーセントにあることを意味します。ボタンの下部は、パネルの下端から、パネルの高さの 20 パーセントです。ボタン自身は、パネルの幅の 60 パーセントで、パネルの高さの 25 パーセントです。

一般的な単位の使用. GUI をレイアウトする場合、より一般的な単位、たとえば、インチまたはセンチメートルを使用すると便利な場合があります。しかし、異なるコンピューター上で GUI の外見を保つには、Figure の Units プロパティを characters に変更し、M ファイルを保存する前に、コンポーネントの Units プロパティを characters (サイズ変更不可の GUI) あるいは normalized (サイズ変更可能な GUI) に変更します。

GUI のプログラミング

- ・ “はじめに” (p.12-2)
- ・ “GUI の初期化” (p.12-3)
- ・ “コールバック:概要” (p.12-7)
- ・ “例:GUI コンポーネントのプログラミング” (p.12-20)

はじめに

GUI をレイアウトした後、その動作をプログラムします。この章は、プログラムにより作成される GUI のプログラミングについて扱います。特に、データ作成、GUI の初期化、コールバックを用いた GUI の動作のコントロールについて説明します。

以下のリストは、標準の GUI M ファイル構成内の次のトピックスを示します。

- 1 MATLAB の `help` コマンドに応答して表示されるコメント
- 2 データ生成などの初期化タスクとコンポーネントを作成するための処理。詳細は、“GUI の初期化”(p.12-3)を参照してください。
- 3 Figure とコンポーネントの作成。詳細は、章 11, “GUI のレイアウト”を参照してください。
- 4 存在するコンポーネントが必要とする初期化タスクと返す出力。詳細は、“GUI の初期化”(p.12-3)を参照してください。
- 5 コンポーネントに対するコールバック。コールバックは、マウスクリックやキーストロークなど、ユーザーが起こすイベントに応答して実行するルーチンです。詳細は、“コールバック:概要”(p.12-7)と“例:GUI コンポーネントのプログラミング”(p.12-20)を参照してください。
- 6 ユーティリティ関数

この章の議論は、入れ子関数の利用を仮定しています。入れ子関数の使用する詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”について参照してください。

GUI を作成するために用意されている関数の一覧は、MATLAB 関数リファレンス ドキュメンテーションの“Function Reference”を参照してください。

メモ MATLAB では、1度の関数呼び出しで作成できる標準的なダイアログ ボックスを選択できます。これらのダイアログ ボックスと作成に使用する関数についての詳細は、MATLAB 関数リファレンス ドキュメンテーションの“Predefined Dialog Boxes”を参照してください。

GUI の初期化

GUI を開くと、通常、特定のデータ構造と変数値が初期化されます。これらのアクションに以下を含めることができます。

- ・ 入力引数と出力引数をサポートする変数の定義。“入力引数と出力引数に対する変数の宣言”(p.12-4)を参照してください。
- ・ 入力引数と出力引数に対する既定値の定義。
- ・ コンポーネントを構成するために用いられるカスタム プロパティ値の定義。“カスタムのプロパティと値の組を定義”(p.12-4)を参照してください。
- ・ コマンド ライン入力引数の処理。
- ・ M ファイルの初期化セクションの下に入れ子にされている関数で使用される変数の作成。MATLAB Programming ドキュメンテーションの“入れ子関数”を参照してください。
- ・ GUI 間でデータを共有するための変数の定義。
- ・ 要求されたときにユーザーに出力を返す。
- ・ コンポーネントの更新または初期化。
- ・ GUI の外観の変更または改良。
- ・ プラットフォームにまたがって機能するように GUI を適合させる。“クロスプラットフォーム互換性のための設計”(p.11-93)を参照してください。
- ・ GUI のすべてのコンポーネントの使用準備が整うまで GUI を非表示にする。“Figure の非表示化”(p.12-5)を参照してください。
- ・ ユーザーに GUI を見せる用意ができたら、GUI を表示。

コード中にタスクを分散させるのではなく、これらのタスクをグループ化します。初期化タスクが長かったりあるいは複雑な場合には、この作業を行うユーティリティ関数の作成を考えてください。

通常は、いくつかの初期化タスクは、コンポーネントが構成される前に M ファイルに現れます。その他は、コンポーネントが作成された後に表示されます。コンポーネントを必要とする初期化タスクは、コンポーネントの構成に続く必要があります。

例

以下は、章 15, “プログラミングで作成する GUI の例”に述べた例からのいくつかの初期化の例です。ユーザーシステム上で MATLAB が実行している場合、これらのリンクを使用して完全な M ファイルを参照できます。

- ・ カラー パレット
- ・ アイコン エディター

入力引数と出力引数に対する変数の宣言

これらは、入力引数と出力引数に対する標準的な宣言です。これらは、例“アイコン エディター”(p.15-60)のものです。

```
mInputArgs = varargin; % Command line arguments when invoking
% the GUI
mOutputArgs = {} ; % Variable for storing output when GUI
% returns
```

詳細は、varargin のリファレンス ページとIcon Editor M ファイルを参照してください。

カスタムのプロパティと値の組を定義

例 “アイコン エディター”(p.15-60) は、入力引数として使用されるプロパティ値の組を定義します。

この例は、セル配列 mPropertyDefs のプロパティを定義し、プロパティを初期化します。

```
mPropertyDefs = [...
    'iconwidth' , @localValidateInput, 'mIconWidth';
    'iconheight' , @localValidateInput, 'mIconHeight';
    'iconfile' , @localValidateInput, 'mIconFile' );
mIconWidth = 16; % Use input property 'iconwidth' to initialize
mIconHeight = 16; % Use input property 'iconheight' to initialize
mIconFile = fullfile(matlabroot,'toolbox/matlab/icons/');
    % Use input property 'iconfile' to initialize
```

セル配列の各行は、1つのプロパティを定義します。これは、順に、プロパティの名前、入力を有効化するために呼び出されるルーチン、プロパティ値を保持する変数の名前を指定します。

関数 `fullfile` は、一部分からフルファイル名を構成します。

次の各ステートメントは、アイコン エディターを起動します。最初のものは、新しいアイコンを作成するために使用できます。第 2 のものは、既存のアイコンファイルを編集するために使用できます。

```
cdata = iconEditor('iconwidth', 16, 'iconheight', 25)
cdata = iconEditor('iconfile', 'eraser.gif');
```

`iconEditor` は、以下の初期化の際に、ルーチン `processUserInputs` を呼び出します。

- ・ セル配列の 1 番目の列に各プロパティをマッチングさせて各プロパティを識別する
- ・ 入力を有効化するために、第 2 列の名前が付けられたルーチンを呼び出す
- ・ 第 3 列の名前が付けられた変数に値を割り当てる

詳細は、完全な Icon Editor M ファイルを参照してください。

Figure の非表示化

GUI の Figure を作成するときには非表示にしておき、完成したら表示されるようにしてください。Figure 作成時に非表示化しておくと、パフォーマンスも向上します。

GUI を非表示にするには、Figure の `Visible` プロパティを `off` にします。これによって、Figure ウィンドウ全体が非表示になります。Figure を作成するステートメントは、次のようになります。

```
hMainFigure = figure(
    'Units', 'characters', ...
    'MenuBar', 'none', ...
    'Toolbar', 'none', ...
    'Position', [71.8 34.7 106 36.15], ...
    'Visible', 'off');
```

コーラーに戻る直前に、次のようなステートメントを使用して Figure を表示できます。

```
set(hMainFigure, 'Visible', 'on')
```

多くのコンポーネントは、`Visible` プロパティをもちます。これらのプロパティを使用して、個々のコンポーネントを非表示にできます。

ユーザーに出力を返す

ユーザー GUI 関数が等号の左辺に引数を提供していて、ユーザーがそのような引数を指定するときには、予想した出力が返されることを期待します。この出力を与えるコードは、通常、GUI が返る直前に表示されます。

Icon Editor の例の M ファイルからの、ここに示す例では、

1 `uiresume` への呼び出しは、`uiresume` が呼び出されたり、あるいは現在の Figure が削除されるまで、実行がブロックされます。

2 実行がブロックされている間、GUI ユーザーは、目的のアイコンを作成します。

3 ユーザーが OK をクリックしてアイコンの完成を知らせると、OK プッシュ ボタンを提供するルーチンは `uiresume` を呼び出し、`uiwait` への呼び出しに続くステートメントにコントロールが返ります。

4 すると、GUI は完成したアイコンを、GUI の出力としてユーザーに返します。

```
% Make the GUI blocking.  
uiwait(hMainFigure);  
  
% Return the edited icon CData if it is requested.  
mOutputArgs{1} = mIconCData;  
if nargout>0  
    [varargout{1:nargout}] = mOutputArgs{:};  
end
```

`mIconData` は、ユーザーが作成または編集したアイコンを含みます。`mOutputArgs` は、出力引数を保持するために定義されたセル配列です。`nargout` は、ユーザーが与えた出力引数の数を示します。`varargout` は、GUI によって返されるオプションの出力引数を含みます。詳細は、完全な Icon Editor M ファイルを参照してください。

コールバック:概要

このセクションの内容…

“コールバックとは” (p.12-7)

“コールバックの種類” (p.12-8)

“コンポーネントに対するコールバックを与える” (p.12-11)

コールバックとは

コールバックは、ユーザーが記述し、GUI の特定のコンポーネントあるいは GUI の Figure と関連させる機能です。コールバックは、コンポーネントに対するイベントに応答して何らかのアクションを行うことで、そのコンポーネントの動作をコントロールします。イベントは、プッシュボタンのマウスクリック、メニュー選択、キープレスなどになります。このようなプログラミングは、イベントドリブン プログラミングと呼ばれることがあります。

ユーザーが提供するコールバック関数は、GUI がボタンクリック、スライダー移動、メニュー項目の選択、あるいはコンポーネントの作成と削除などのイベントにどのように応答するかをコントロールします。各コンポーネントと GUI の Figure 自身に対するコールバックのセットがあります。

コールバックルーチンは、通常、初期化コードとコンポーネントの作成に続いて、M ファイルに現れます。詳細は、“ファイル編成” (p.11-4) を参照してください。

コンポーネントに対してイベントが起こると、そのイベントに関連するコンポーネントのコールバックが MATLAB により呼び出されます。例として、あるデータのプロットをトリガーするプッシュボタンをもつ GUI を考えます。ユーザーがボタンをクリックすると、MATLAB は、そのボタンをクリックすることに関するコールバックを呼び出します。すると、ユーザーがプログラムしたコールバックは、データを取得してプロットします。

コンポーネントは、座標軸、プッシュボタン、リストボックス、スライダーなどのコントロール デバイスです。プログラミングでは、コンポーネントは、パネルやボタン グループなどのコンテナ、ツールバー ツール、あるいはメニューにもなります。コンポーネントのリストや説明として、“利用可能なコンポーネント” (p.11-9) を参照してください。

コールバックの種類

GUI の Figure とコンポーネントの各タイプは、関連できるある種のコールバックをもちます。各コンポーネントに対して利用できるコールバックは、そのコンポーネントのプロパティとして定義されます。たとえば、プッシュボタンは 5 つのコールバックプロパティ ButtonDownFcn, Callback, CreateFcn, DeleteFcn, KeyPressFcn をもちます。パネルは 4 つのコールバックプロパティ ButtonDownFcn, CreateFcn, DeleteFcn, ResizeFcn をもちます。これらのプロパティのそれぞれに対してコールバック関数を作成できますが、作成は必須ではありません。GUI 自身は Figure ですが、関連できるある種のコールバックも、もちます。

各コールバックには、トリガー メカニズム、すなわち、そのコールバックを呼び出すイベントがあります。次の表は、GUIDE が利用するコールバックプロパティ、イベントのトリガー、それらが適用するコンポーネントをリストします。最初の列のリンクは、それぞれのタイプのコールバックのドキュメンテーションの検索結果にリンクします。これらのリンクは、MATLAB ヘルプ ブラウザーを使用しているときに限り機能します。

コールバックプロパティ	イベント トリガー	コンポーネント
ButtonDownFcn	ポイントがコンポーネント上にあつたり、または Figure のコンポーネントから 5 ピクセル以内のときに、ユーザーがマウスボタンを押すと実行します。	座標軸、Figure、ボタン グループ、パネル、ユーザー インターフェイス コントロール
コールバック	アクションのコントロール。たとえば、ユーザーがプッシュボタンをクリックしたり、メニュー項目を選択すると、実行します。	コンテキスト ミュージュー、メニュー、ユーザー インターフェイス コントロール
CellEditCallback	編集可能なセルに対して、表の値の編集についてレポートします。イベント データを使用します。	uitable
CellSelectionCallback	表にあるマウスの動作で選択された、セルのインデックスをレポートします。イベント データを使用します。	uitable

コールバックプロパティ	イベントトリガー	コンポーネント
ClickedCallback	アクションのコントロール。プッシュツールまたはトグルツールがクリックされると、実行します。トグルツールでは、その状態に依存しません。	プッシュツール、トグルツール
CloseRequestFcn	Figure が閉じるとき実行します。	図
CreateFcn	コンポーネントが作成されると、コンポーネントを初期化します。Figure やコンポーネントが作成された後、表示されるまでに実行します。	座標軸、ボタン グループ、コンテキストメニュー、Figure、メニュー、パネル、プッシュツール、トグルツール、ユーザーインターフェイス コントロール
DeleteFcn	コンポーネントまたは Figure が削除される直前にクリーンアップ作業を実行します。	座標軸、ボタン グループ、コンテキストメニュー、Figure、メニュー、パネル、プッシュツール、トグルツール、ユーザーインターフェイス コントロール
KeyPressFcn	コールバックのコンポーネントまたは Figure が選択されていてユーザーがキーボードのキーを押すと、実行します。	Figure、ユーザーインターフェイス コントロール
KeyReleaseFcn	ユーザーがキーボードキーをはなし、Figure がフォーカスされると実行します。	図
OffCallback	アクションのコントロール。トグルツールの State が off に変わると、実行します。	トグルツール

コールバックプロパティ	イベントトリガー	コンポーネント
OnCallback	アクションのコントロール。トグルツールの State が on になると、実行します。	トグルツール
ResizeFcn	ユーザーがパネル、ボタン グループ、あるいは Resize プロパティが On に設定されている Figure をサイズ変更するときに実行します。	Figure、ボタン グループ、パネル
SelectionChangeFcn	ボタン グループ コンポーネントで、異なるラジオ ボタンやトグル ボタンを選択すると、実行します。	ボタン グループ
WindowButtonDownFcn	ポインタが Figure ウィンドウ内にある場合にユーザーがマウスボタンをクリックすると、実行します。	図
WindowButtonMotionFcn	ポインターが Figure ウィンドウ上を動くと実行します。	図
WindowButtonUpFcn	マウスボタンをはなすと、実行します。	図
WindowKeyPressFcn	Figure またはその子オブジェクトのいずれかが選択されているときに、キーを押すと、実行します。	図
WindowKeyReleaseFcn	Figure またはその子オブジェクトのいずれかが選択されているときに、キーをはなすと、実行します。	図
WindowScrollWheelFcn	マウスがスクロールされ Figure がフォーカスされると、実行します。	図

メモ ユーザー インターフェイス コントロールは、プッシュ ボタン、スライダー、ラジオ ボタン、チェック ボックス、エディット テキスト ボックスやスタティック テキスト ボックス、リスト ボックス、トグル ボタンを含みます。これらは、*uicontrol* オブジェクトとして参照されることがあります。

特定のコールバックが使用される方法を知るには、前述の表のリンクに従います。与えられたコールバック プロパティに固有の情報を取得するには、ユーザー コンポーネント、たとえば、「Figure プロパティ」、「Uicontrol プロパティ」、「Uibuttongroup プロパティ」、または「Uitable プロパティ」のリファレンス ページを確認してください。

コンポーネントに対するコールバックを与える

GUI は、多数のコンポーネントをもつことができます。各コンポーネントのプロパティは、そのコンポーネントに対する特定のイベントに応答して、どのコールバックが実行するかを指定する方法を提供します。ユーザーが Yes ボタンをクリックする際に実行するコールバックは、No ボタンに対して実行するものとは通常異なります。各メニュー項目は、別の関数も実行し、それ自身のコールバックが必要です。

コンポーネントの *Callback* プロパティの（前の表に述べた）値を、プロパティ/値の組としてコールバックに設定することで、特定のコンポーネントにコールバックを付加します。プロパティはコールバックのタイプを特定し、値はそれに実行する関数を特定します。これは、コンポーネントを定義するとき、あるいは後で他の初期化コードにおいて、行うことができます。GUI が使用されているときに、コードでコールバックを変更することもできます。

コンポーネントのコールバックのプロパティ値を、以下の 1 つとして指定します。

- ・ 評価する 1 つ以上の MATLAB またはツールボックスのコマンドを含む文字列
- ・ GUI が実行しているときにスコープ内にある関数へのハンドル
- ・ 引数として、文字列の関数名、または関数ハンドル、さらにオプションの文字列、定数、または変数名を含むセル配列

コールバックのプロパティ名と値（その呼び出し列）を与えることでコンポーネントを作成する場合、コールバックを付加できます。後で、*set* コマンドを用いて、コールバックの追加や置換もできます。以下の例はすべて、コールバックが指定するいくつかのパラメーターは、コンポーネントが作成された時点で存在しないか必要な値をもたないので、推奨の方法として *set* を使用します。

文字列コールバックの利用

String コールバックは、独立しているので最も作成しやすいタイプです。それらは、M ファイルではなく、GUI の Figure 自身の中にもあります。目的が単純であるときには文字列のコールバックを使用できますが、コールバックが 2 つ以上の動作を行ったり、2 つ以上のパラメーターを必要とすると、複雑になります。コールバックに対して使われる文字列は、組み込み、または M ファイル関数を含む、有効な MATLAB 表現またはコマンドであり、関数に対する引数を含むことができます。以下に例を示します。

```
hb = uicontrol('Style','pushbutton',...
    'String','Plot line')
set(hb,'Callback','plot(rand(20,3))')
```

有効な MATLAB コマンドである、コールバック文字列 'plot(rand(20,3))' は、ボタンがクリックされると評価されます。たとえば、以下のように変数をプロットするコールバックを変更すると、

```
set(hb,'Callback','plot(myvar)')
```

コールバックがトリガーされるか、またはコールバックがエラーを起こす時点で、変数 myvar はベースワークスペースに存在しなければなりません。この変数は、コールバックがトリガーされるときには、コールバックがコンポーネントに与えられた時点で存在する必要はありません。コールバックを使用する前に、コードでこれを宣言できます。

```
myvar = rand(20,1);
```

文字列コールバックは、定義時に引数が変数として存在する必要がない唯一のコールバックタイプです。関数ハンドルコールバックの引数は、定義時に評価されるため、その時点では存在しなければなりません。

ワークスペースの詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションと関数 evalin のリファレンス ページの“Scope of a Variable”を参照してください。

文字列のコールバックでコマンドを連結することができます。たとえば、これは作成するプロットにタイトルを追加します。

```
set(hb,'Callback',...
    'plot(myvar,'--m'); title('String Callback'))
```

メモ 文字列内に存在する文字を囲むために、單一引用符が 2 つ必要になります。

関数ハンドル コールバックの使用

関数ハンドル（記号 @ が前に付いた関数名。たとえば、@my_function など）の使用について留意すべき最も重要なことは、次のとおりです。

- ・ 関数ハンドルは、M ファイルのファイル名ではなく、M ファイル内の関数の名前です。
- ・ MATLAB スクリプト内に関数を配置することはできません。
- ・ 関数を使用するコールバックの宣言時にその関数が存在する必要はありません。
- ・ コールバックの実行時に、その関数を含む M ファイルがユーザー パス上になければなりません。
- ・ セル配列内のすべてのものを改行しない限り、Callback プロパティ定義内で関数ハンドルの後に引数を続けることはできません。
- ・ コールバック関数宣言には、Handle Graphics により自動的に与えられ、一般に (hObject, eventdata) と呼ばれる 2 つの初期引数を含めなければなりません。
- ・ これら 2 つの引数（コールバックと、このコールバックがオプションで提供するイベント データを発行するオブジェクトのハンドル）は、Callback プロパティ定義に現れてはなりません。

uicontrol の定義時にコールバックを宣言する例を次に示します。

```
figure
uicontrol('Style','slider','Callback',@display_slider_value)
```

GUI M ファイル内の関数の定義を次に示します。スライダーの値を調整すると、コールバックはその値を出力します。

```
function display_slider_value(hObject, eventdata)
disp(['Slider moved to ' num2str(get(hObject,'Value'))]);
```

スライダーの矢印をクリックすると、関数の出力は次のようにになります。

```
Slider moved to 0.01
Slider moved to 0.02
...
```

どちらのコード セクションも、同じ GUI M ファイル内に存在しなければなりません。uicontrol を定義するものを、GUI を設定する関数、通常メイン関数に含めます。サブ関数または入れ子関数としてコールバック関数を追加します。詳細は、MATLAB「P

「プログラミングの基礎」ドキュメンテーションの“Subfunctions”および“Nested Functions”を参照してください。

セル配列のコールバックの利用

コールバックに対して引数を指定する必要がある場合、セル配列内の関数名文字列または関数ハンドルと引数を改行できます。

- ・ コールバックを文字列として識別して、その名前、たとえば、pushbutton_callback.mなどをもつ M ファイルを実行します。
- ・ コールバックを関数ハンドルとして識別して、たとえば、@pushbutton_callback などの現在実行中の M ファイル内のサブ関数または入れ子関数を実行します。

次の 2 節では、これら 2 つの方法の違いについて説明します。

文字列を含むセル配列の使用. 次のセル配列のコールバックは、引用符で囲まれた文字列 'pushbutton_callback' として M 関数名と、変数名と文字列の 2 つの引数を定義します。

```
myvar = rand(20,1);
set(hObject,'Callback',[ 'pushbutton_callback' , myvar, '--m' ])
```

セル配列の最初に関数名を置き、文字列として指定します。この形のコールバックが実行されると、MATLAB はセル配列の最初の要素の名前をもつ M ファイルを実行します。コールバックに、2 つの標準的な引数を渡し、続いてユーザーが指定するセル配列の要素を追加して渡します。関数名とリテラル文字列引数を一重引用符で囲みます。ただし、ワークスペース変数名引数は囲みません。関数は、MATLAB パス上に存在する必要があります、少なくとも 2 つの引数をもたなければなりません。最初の 2 つ (MATLAB が自動的に挿入) は、以下のものです。

- ・ 現在コールバックが呼び出されているコンポーネントのハンドル
- ・ イベントデータ (いくつかの Figure および GUI コンポーネント コールバックが提供するがほとんどは空の行列を渡す MATLAB 構造体) 詳細は、“イベントデータを渡すコールバック” (p.12-18) を参照してください。

コンポーネントのコールバックを宣言するときにハンドルおよびイベントデータ引数を含めないようにしてください (たとえば、
set(hObject,'Callback',['pushbutton_callback' , myvar, '--m']) など)。ただし、次の段落で説明するように、コールバックの定義には含めてください。

コンポーネントのコールバックを指定する場合、これら 2 つの引数にユーザーが含める引数が続きます。'pushbutton_callback' を実行するコードは、以下のようになります。

```
function pushbutton_callback(hObject, eventdata, var1, var2)
plot(var1, var2)
```

定義する引数は、変数、定数、または文字列になります。コールバックが引数として使用する変数は、コールバック プロパティを定義する時点で現在のワークスペースに存在しなければなりません。上記の例において、最初の引数の値（変数 `myvar`）は、設定されるときコールバックにコピーされます。その結果、`myvar` が現在存在しないと、エラーとなります。

```
??? Undefined function or variable 'myvar'.
```

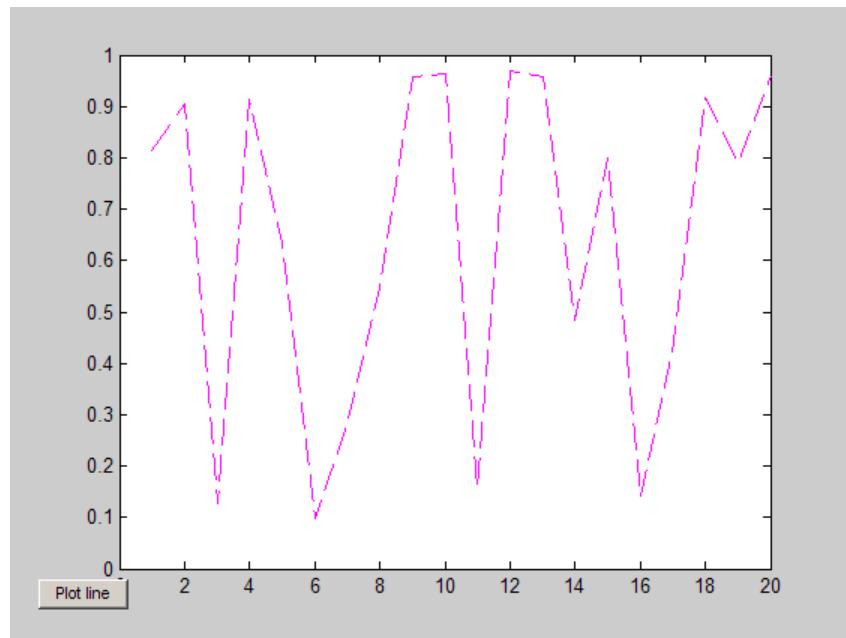
コールバックを定義した後に `myvar` が変更されたり削除されても、元の値が使用されます。

2 つめの引数（'—m'）は、文字列リテラル `LineSpec` です。これは、変数を参照しないので、コールバックを指定するときにエラーを起こすことはありません。ただし、関数の引数リストにこれが含まれていない場合を除きます。

この GUI を使用するには、以下のコードを含む `pushbutton_callback.m` という M ファイルを作成します。

```
function pushbutton_callback(hObject, eventdata, var1, var2)
plot(var1, var2)
```

プッシュ ボタンを押すことでこの GUI を実行すると、以下のような、マジンダの点線として現れる `myvar` の線グラフが表示されます。データを生成するために関数 `rand` を使用しているのでグラフは異なることがあります。



`myvar` の値は設定されたときにコールバックにコピーされたので、ボタンをクリックすると、`myvar` の値がベースワークスペースで変化していても、常に同じプロットを生成します。

詳細は、MATLAB Graphics ドキュメンテーションの “文字列のセル配列としてのコールバック定義 – 特殊な場合” を参照してください。

関数ハンドルのセル配列の利用. 関数名を使用する代わりに、関数ハンドルを使用してコールバック関数を指定することができます。関数ハンドルを利用する主な利点は、オンザフライで関数を定義できる機能です。つまり、たとえば、無名関数など、関数のスコープ内で定義された関数のハンドルに、コンポーネントのコールバックを設定する M-コードを実行するときです。コールバックの動的な割り当てにより、コールバックが機能するコンテキストまたは処理するデータに応じてコールバックの動作を変化させることができます。コールバックの宣言時に関数ハンドルを引用符で囲まないでください。

以下の変化は、ユーザーが [Plot line] をクリックするときに実行されるコールバックルーチンとして、`pushbutton_callback` を指定する関数ハンドルを使用します。

```

figure;
hb = uicontrol('Style','pushbutton',...
    'String','Plot line')
set(hb,'Callback',{@pushbutton_callback,myvar,'--m'})

```

Callback は、コールバックプロパティの名前です。セル配列の最初の要素は、コールバックルーチンのハンドルであり、続く複数の要素は、コールバックに対する入力引数です。関数ハンドルは文字列やファイル名ではないため、一重引用符で囲まないでください。前述の例の `'linespec' '--m'` など、リテラル文字列であるコールバック引数に対してのみ引用符を使用してください。セル配列の 2 つめと 3 つめの要素、変数 `myvar` と文字列 `'--m'` は、コールバックの 3 つめと 4 つめの引数になります。これらは、`hObject` と `eventdata` の後にあります。

上記のように、コールバックは、以下のようなコードを含む `pushbutton_callback.m` という名前の M ファイルにあります。

```

function pushbutton_callback(hObject, eventdata, var1, var2)
plot(var1,var2)

```

前の例からわかるように、セル配列を用いて、コールバック内で関数名（一重引用符で囲む）または関数ハンドル（一重引用符なし）を指定できます。結果は同じになります。関数ハンドルを利用すると、ユーザー・アプリケーションが動的に動作しなければならないときに、さらにフレキシビリティがあります。

メモ (“文字列コールバックの利用” (p.12-12)で説明されているように、必要な引数を使用して) 文字列として宣言しない限り、通常の関数をコールバックとして使用しないでください。そのようにすると、その関数はエラーになり、予期しない動作をする可能性があります。MATLAB GUI コンポーネントのコールバックは、自動生成された引数を含むので、通常の MATLAB または toolbox の関数名、あるいは関数ハンドル（例、`plot` または `@plot`）をコールバックとして単純に指定することはできません。

さらに、GUIDE が生成したコールバック関数シグネチャは、3 つめの自動生成された引数 `handles` を含みます。GUIDE でのコールバックの取り扱いの詳細は、GUIDE ドキュメンテーションの “コールバック構文と引数” (p.8-15) を参照してください。

関数ハンドルの利用の詳細は、MATLAB Graphics ドキュメンテーションの “Function Handle Callbacks”を参照してください。利用可能なコールバックのまとめは、“コールバックの種類” (p.12-8)を参照してください。それぞれの種類のコンポーネントが

サポートする、特定のタイプのコールバックの詳細については、コンポーネントのプロパティのリファレンス ページを参照してください。

イベント データを渡すコールバック

Figure と GUI コンポーネントのコールバックには、 eventdata 引数にユーザーが起こすイベントを記述するデータを与えるものがあります。このデータが存在する場合、コールバックの 2 番目の引数を占有します。コールバックのイベント データが存在しない場合、引数は空行列になります。たとえば、プッシュ ボタンの KeyPressFcn コールバックは、次のようにイベント データを受け取ります。

```
function pushbutton1_KeyPressFcn(hObject, eventdata)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata   structure with the following fields (see UICONTROL)
%   Key:        name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier:   name(s) of the modifier key(s) (i.e., control, shift) pressed
```

次の表は、イベント データを提供するコールバックとそれらが適用されるコンポーネントをまとめています。詳細は、適切なプロパティのリファレンス ページへのリンクをクリックしてください。

GUI コンポーネント	イベント データをもつコールバック	プロパティのリファレンス ページ
図	KeyPressFcn, KeyReleaseFcn, WindowKeyPressFcn, WindowKeyReleaseFcn, WindowScrollWheel	Figure プロパティ
ユーザー インターフェイス コントロール (uicontrol)	KeyPressFcn	Uicontrol プロパティ
ボタン グループ (uibuttongroup)	SelectionChangeFcn	Uibuttongroup プロパティ
表 (uitable)	CellEditCallback, CellSelectionCallback	Uitable プロパティ

コンポーネント間でのコールバックの共有

GUIDE を使用せずにプログラミングにより GUI を設計している場合、2つ以上のコンポーネントに同じコールバックを与えることができます。これは、コントロールのグループが、少し異なる似たような動作を行う場合、または異なるデータに同じように動作する場合に使用すると良い手法です。そのような場合、それぞれのケースを取り扱うために別々のコードパスを与える、1つのコールバック関数を設計できます。コールバックは、それを呼び出すオブジェクトの特性や種類、あるいは、それに渡されたパラメーターに基づき、どのコードパスをとるかを決めることができます。

表形式データの3つの異なる列をプロットする3つのチェックボックスが共有するコールバックの例は、“テーブルのデータを表示してグラフを作成する GUI”(p.15-17)を参照してください。3つのコンポーネントはすべて同じことを行います。それらが共有するコールバックの最後の引数は、プロットしているときから、データを取得するための列数を提供します。

例: GUI コンポーネントのプログラミング

このセクションの内容…

- “ユーザー インターフェイス コントロールのプログラミング” (p.12-20)
- “パネルとボタン グループのプログラミング” (p.12-28)
- “座標軸のプログラミング” (p.12-30)
- “ActiveX コントロールのプログラミング” (p.12-34)
- “メニュー項目のプログラミング” (p.12-34)
- “ツールバー ツールのプログラミング” (p.12-36)

ユーザー インターフェイス コントロールのプログラミング

例は、コールバック プロパティが関数ハンドルを用いて指定されることを仮定しています。MATLAB が、引数として、イベントがトリガーされたコンポーネントのハンドルである `hObject` と `eventdata` を渡すことを可能にします。詳細は、“コンポーネントに対するコールバックを与える” (p.12-11) を参照してください。

- ・ “チェック ボックス” (p.12-21)
- ・ “テキストの編集” (p.12-21)
- ・ “リスト ボックス” (p.12-23)
- ・ “ポップアップメニュー” (p.12-24)
- ・ “プッシュ ボタン” (p.12-25)
- ・ “ラジオ ボタン” (p.12-26)
- ・ “スライダー” (p.12-26)
- ・ “トグル ボタン” (p.12-27)

メモ これらのコンポーネントの詳細は、“利用可能なコンポーネント” (p.11-9) を参照してください。ユーザー GUI へのこれらのコンポーネントの追加についての詳細は、“ユーザー インターフェイス コントロールの追加” (p.11-13) を参照してください。

チェック ボックス

次の例に示されるように、Value プロパティの状態を確認することによりコールバック内から、チェック ボックスの現在の状態を決定することができます。

```
function checkbox1_Callback(hObject, eventdata)
if (get(hObject,'Value') == get(hObject,'Max'))
    % Checkbox is checked-take appropriate action
else
    % Checkbox is not checked-take appropriate action
end
```

`hObject` はイベントがトリガーされたコンポーネントのハンドルです。

チェック ボックスの Value プロパティに、チェック ボックスの Max または Min プロパティの値を設定することによって、チェック ボックスの状態をプログラムによっても変更できます。たとえば、

```
set(cbh,'Value', 'Max')
```

は、ハンドル `cbh` をもつチェック ボックスをチェックされた状態にします。

テキストの編集

ユーザーがエディットボックスにタイプする文字列を取得するには、エディットボックスのコールバックのいずれかを使用して、String プロパティの値を取得します。この例は、Callback コールバックを使用します。

```
function edittext1_Callback(hObject, eventdata)
user_string = get(hObject,'String');

% Proceed with callback
```

エディット テキストの Max プロパティと Min プロパティの値を $\text{Max} - \text{Min} > 1$ であるように設定すると、ユーザーは複数行を入力できます。たとえば、Min を既定値 0 として Max を 2 に設定すると、ユーザーは複数行の入力が可能です。ユーザーが最初に String をキャラクター文字列として指定する場合、各行に 1 行を含み、複数行のユーザー入力が 2 次元文字配列として返されます。最初に String をセル配列として指定すると、複数行のユーザー入力が文字列の 2 次元セル配列として返されます。

`hObject` はイベントがトリガーされたコンポーネントのハンドルです。

エディット テキスト コンポーネントからの数値データの取得. MATLAB は、キャラクター文字列として、エディット テキストの「String」プロパティの値を返します。数値の入力を希望する場合は、文字を数字に変換する必要があります。これは、コマンド `str2double` を使用して行うことができ、文字列を倍精度値に変換します。数字でない文字列を入力した場合、`str2double` は `NaN` を返します。

エディット テキスト コールバックにおいて、次のようなコードを使用できます。これは `String` プロパティの値を取得し、倍精度値に変換します。変換された値が、数字でない文字の入力を示す `NaN` (`isnan`) であるかどうかをチェックし、エラー ダイアログ ボックス (`errordlg`) を表示します。

```
function edittext1_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg(' You must enter a numeric value' , 'Bad Input' , 'modal')
    uicontrol(hObject)
    return
end
% Proceed with callback...
```

エディット テキスト コントロールは、(Return を入力するか、あるいはクリックすることで) ユーザーが送信して変更するときに、フォーカスがなくなります。行 `uicontrol(hObject)` は、エディット テキスト ボックスへのフォーカスを元の状態に戻します。これを行うことは、そのコールバックが動作するために必要ありませんが、ユーザー入力が検証できないイベントにおいては役立ちます。このコマンドは、エディット テキスト ボックスでテキストすべてを選択する効果があります。

コールバック実行のトリガー. エディット テキスト コンポーネントの内容が変更された場合、GUI 内でエディット テキストの外部をクリックすると、エディット テキスト コールバックが実行されます。ユーザーは、1行のみのエディット テキストに対して Enter を押すか、あるいは複数行のエディット テキストに対して Ctrl+Enter を使用することもできます。

利用可能なキーボードアクセラレータ. GUI のユーザーは以下のキーボードアクセラレータを用いて、エディット テキストのコンテンツを修正できます。これらのアクセラレータは、変更可能ではありません。

- `Ctrl+X` – カット
- `Ctrl+C` – コピー

- Ctrl+V – 貼り付け
- Ctrl+H – 最後の文字の削除
- Ctrl+A – すべて選択

リスト ボックス

リスト ボックスの `Callback` コールバックがトリガーされると、リスト ボックスの `Value` プロパティは選択された項目のインデックスを含みます。ここで 1 はリストの最初の項目に対応します。`String` プロパティは、文字列のセル配列としてリストを含みます。

次の例では、選択された文字列を取得します。セル配列から `String` プロパティにより返される値を文字列に変換する必要ですので、注意してください。

```
function listbox1_Callback(hObject, eventdata)
index_selected = get(hObject,'Value');
list = get(hObject,'String');
item_selected = list{index_selected}; % Convert from cell array
% to string
```

`hObject` はイベントがトリガーされたコンポーネントのハンドルです。

リスト ボックスの `Value` プロパティを、表示したい項目のインデックスに設定することによって、リスト項目をプログラムでも選択できます。たとえば、

```
set(lbh,'Value',2)
```

は、ハンドル `lbh` をもつリスト ボックスの 2 番目の項目を選択します。

コールバック実行のトリガー. MATLAB は、マウスボタンがはなされたり、あるキーが押された後、リスト ボックスの `Callback` プロパティを用いたコールバックを実行します。

- 矢印キーは、`Value` プロパティを変更してコールバックの実行をトリガーし、Figure の `SelectionType` プロパティを `normal` に設定します。
- Enter キーとスペース バーは、`Value` プロパティを変更しませんが、コールバックの実行をトリガーし、Figure の `SelectionType` プロパティを `open` に設定します。

ユーザーがダブルクリックすると、コールバックが各クリック後に実行します。MATLAB は、Figure の `SelectionType` プロパティを最初のクリックで `normal` に設定

し、2回目のクリックで open に設定します。コールバックは、Figure の SelectionType プロパティを問い合わせてクリックが1回であるか2回であるかを確認できます。

リスト ボックスの例. リスト ボックスを利用した詳細については、以下の例を参照してください。

- ・ “リスト ボックス ディレクトリ リーダー” (p.10-54) — リスト ボックスにディレクトリの内容を表示する GUI を生成する方法を示します。これにより、ファイル名をダブルクリックして、様々なファイルタイプを開くことができるようになります。
- ・ “リスト ボックスからワークスペース変数にアクセス” (p.10-61) — リスト ボックスの GUI から、MATLAB ベース ワークスペース内の変数にアクセスする方法を示します。

ポップアップメニュー

ポップアップ メニューの Callback コールバックがトリガーされると、ポップアップ メニューの Value プロパティは、選択された項目のインデックスを含みます。ここで 1 はメニューの最初の項目に対応します。String プロパティは、文字列のセル配列としてメニュー項目を含みます。

メモ ポップアップメニューは、ドロップダウン メニューまたはコンボボックスと呼ばれることがあります。

選択されたメニュー項目のインデックスのみの利用. この例では、選択された項目のインデックスのみを取得します。これは、switch ステートメントを使用して値に基づいたアクションをとります。ポップアップメニューのコンテンツが固定している場合には、この方法を使用できます。そうでない場合、選択された項目に対する実際の文字列を取得するためにインデックスを使用できます。

```
function popupmenu1_Callback(hObject, eventdata)
val = get(hObject,'Value');
switch val
case 1 % User selected the first item
case 2 % User selected the second item
% Proceed with callback...
```

`hObject` はイベントがトリガーされたコンポーネントのハンドルです。

ポップアップメニューの Value プロパティに表示したいリストのインデックスを設定することによって、表示される項目のリストをプログラムでも指定できます。たとえば、

```
set(pmh,'Value',2)
```

は、ハンドル pmh をもつポップアップメニューの 2 番目の項目を選択します。

選択された文字列を判別するインデックスの使用。 この例は、ポップアップメニューで選択された実際の文字列を取得します。これは、ポップアップメニューの Value プロパティを使用して文字列のリストにインデックスを付けます。この方法は、プログラムがユーザーのアクションに基づきポップアップメニューの内容を動的に読み込み、選択した文字列を得る必要がある場合に有効です。セル配列から String プロパティにより返される値を文字列に変換することが必要ですので、注意してください。

```
function popupmenu1_Callback(hObject, eventdata)
val = get(hObject,'Value');
string_list = get(hObject,'String');
selected_string = string_list{val}; % Convert from cell array
                                  % to string
% Proceed with callback...
```

`hObject` はイベントがトリガーされたコンポーネントのハンドルです。

プッシュ ボタン

この例は、プッシュ ボタンのみを含みます。ボタンをクリックすると、GUI が閉じます。



次はプッシュ ボタンの `Callback` プロパティを用いたコールバックです。これは、コマンドラインに文字列 `Goodbye` を表示してから GUI を閉じます。

```
function pushbutton1_Callback(hObject, eventdata)
display Goodbye
```

```
close(gcf)
```

gcbf は、コールバックが実行されているオブジェクトを含む Figure のハンドルを返します。

ラジオ ボタン

次の例に示されるように、「Value」プロパティの状態を確認することにより Callback コールバック内から、ラジオ ボタンの現在の状態を決定することができます。

```
function radiobutton_Callback(hObject, eventdata)
if (get(hObject,'Value') == get(hObject,'Max'))
    % Radio button is selected-take appropriate action
else
    % Radio button is not selected-take appropriate action
end
```

ラジオ ボタンは、Value を、ラジオ ボタンがオン（選択されたとき）のとき Max に設定し、オフ（選択されていない）のとき Min を設定します。hObject は、イベントがトリガーされたコンポーネントのハンドルです。

ラジオ ボタンの Value プロパティを、Max または Min プロパティの値に設定することによって、ラジオ ボタンの状態をプログラムによっても変更できます。たとえば、

```
set(rbh,'Value','Max')
```

は、ハンドル rbh をもつラジオ ボタンを選択された状態にします。

メモ ボタン グループを用いて、ラジオ ボタンの排他的な選択を管理できます。詳細は、“ボタン グループ”(p.12-28)を参照してください。

スライダー

次の例に示されるように、「Value」プロパティの状態を確認することによりコールバック内から、スライダーの現在の状態を決定することができます。

```
function slider1_Callback(hObject, eventdata)
slider_value = get(hObject,'Value');

% Proceed with callback...
```

Max プロパティと Min プロパティは、スライダーの最大値と最小値を指定します。スライダーの範囲は、Max – Min です。hObject は、イベントがトリガーされたコンポーネントのハンドルです。

トグル ボタン

トグル ボタンに対するコールバックは、トグル ボタンの状態を判定するためにトグル ボタンに問合せる必要があります。MATLAB は、トグル ボタンが押されたとき（既定で Max は 1）、Value プロパティを Max プロパティと同じに設定します。トグル ボタンが押されていないとき（既定で Min は 0）、Value プロパティを Min プロパティと同じに設定します。

次のコードは、GUI M ファイルにおいてコールバックをプログラムする方法を説明します。

```
function togglebutton1_Callback(hObject, eventdata)
button_state = get(hObject, 'Value');
if button_state == get(hObject, 'Max')
    % Toggle button is pressed-take appropriate action
    ...
elseif button_state == get(hObject, 'Min')
    % Toggle button is not pressed-take appropriate action
    ...
end
```

`hObject` はイベントがトリガーされたコンポーネントのハンドルです。

トグル ボタンの Value プロパティに、トグル ボタンの Max または Min プロパティの値を設定することによって、トグル ボタンの状態をプログラムにより変更することもできます。たとえば、

```
set(tbh, 'Value', 'Max')
```

は、ハンドル `tbh` をもつトグル ボタンを押された状態にします。

メモ ボタン グループを用いて、トグル ボタンの排他的な選択を管理できます。詳細は、“ボタン グループ”(p.12-28)を参照してください。

パネルとボタン グループのプログラミング

以下のトピックスは、パネルとボタン グループ コールバックに対する基本的なコード例を提供します。

例は、コールバックプロパティが関数ハンドルを用いて指定されることを仮定しています。MATLAB が、引数として、イベントがトリガーされたコンポーネントのハンドルである `hObject` と `eventdata` を渡すことを可能にします。詳細は、“コンポーネントに対するコールバックを与える”(p.12-11)を参照してください。

- ・ “パネル”(p.12-28)
- ・ “ボタン グループ”(p.12-28)

パネル

パネルは GUI コンポーネントをグループ化し、関連するコントロールを視覚的にグループ化することにより、GUI の理解を容易にします。パネルは、プッシュ ボタン、スライダー、ポップアップ メニューなどのユーザー インターフェイス コントロールや座標軸と同様に、パネルとボタン グループを含むことができます。パネル内での各コンポーネントの位置は、パネルの左下隅を基準に求められます。

一般に、GUI がサイズ変更されると、パネルとそのコンポーネントもサイズ変更されます。ただし、パネルとそのコンポーネントのサイズと位置をコントロールできます。GUI の `Resize` プロパティを `on` に設定し、パネルに対して `ResizeFcn` コールバックを与えると、これを行うことができます。

メモ サイズ変更動作への単位の影響の詳細は、“クロスプラットフォーム互換の Units”(p.11-95)を参照してください。

ボタン グループ

ボタン グループはパネルに似ていますが、ボタン グループはラジオ ボタンやトグル ボタンに対する排他的な選択を管理するために使用されます。ボタン グループが、ラジオ ボタン、トグル ボタン、またはその両方のセットを含む場合、ボタン グループはこれらのうちの 1 つのみが選択されることを許可します。ユーザーがボタンをクリックすると、そのボタンが選択され、他のすべてのボタンは非選択になります。

ボタン グループをプログラミングするとき、個々のボタンに対してコールバックをコーディングしません。代わりに、その `SelectionChangeFcn` コールバックを使用して、選択

に対して応答を管理します。次の例、“ボタン グループのプログラミング”(p.12-30)は、これを行うために `uibuttongroup` のイベントデータを使用する方法を説明します。

次の図は、2 つのラジオ ボタンと 2 つのトグル ボタンをもつボタン グループを示します。Radio Button 1 が選択されます。



ユーザーがもう一方のラジオ ボタンまたはトグル ボタンの 1 つをクリックすると、こちらが選択された状態になり、Radio Button 1 が非選択になります。次の図は、Toggle Button 2 をクリックした結果を示します。



ボタン グループの `SelectionChangeFcn` コールバックは、選択が行なわれるときに呼び出されます。ラジオ ボタンとトグル ボタンのセットを含むボタン グループをもつ場合、以下を行いたいとします。

- ・ ラジオ ボタンまたはトグル ボタンが選択されたときに直ちに起こるアクションは、個々のトグル ボタン `Callback` 関数にではなく、ボタン グループの `SelectionChangeFcn` コールバック関数にラジオ ボタンとトグル ボタンをコント

ロールするコードを含めなければなりません。“カラー パレット”(p.15-49) は、SelectionChangeFcn コールバックの実際の例をえます。

- 選択に基づきアクションをするプッシュ ボタンのような他のコンポーネントでは、そのコンポーネントの Callback コールバックが、ボタン グループの SelectedObject プロパティから、選択されたラジオ ボタンまたはトグル ボタンのハンドルを取得できます。

ボタン グループのプログラミング. SelectionChangeFcn コールバックのこの例は、選択されたオブジェクトの Tag プロパティを用いて、実行する適切なコードを選択します。各コンポーネントの Tag プロパティは、そのコンポーネントを識別する文字列で、GUI で一意でなければなりません。

```
function uibuttongroup1_SelectionChangeFcn(hObject, eventdata)
switch get(eventdata.NewValue, 'Tag') % Get Tag of selected object.
    case 'radiobutton1'
        % Code for when radiobutton1 is selected.
    case 'radiobutton2'
        % Code for when radiobutton2 is selected.
    case 'togglebutton1'
        % Code for when togglebutton1 is selected.
    case 'togglebutton2'
        % Code for when togglebutton2 is selected.
    % Continue with more cases as necessary.
    otherwise
        % Code for when there is no match.
end
```

コールバックプロパティの値が関数ハンドルとして指定される場合に限り、`hObject` と `eventdata` 引数はコールバックに対して利用可能です。`eventdata` の詳細は、Uibuttongroup プロパティのリファレンス ページの SelectionChangeFcn プロパティを参照してください。他の例は、uibuttongroup のリファレンス ページと“カラー パレット”(p.15-49)を参照してください。

座標軸のプログラミング

座標軸により、ユーザー GUI が、グラフィックス (たとえば、グラフやイメージ) を表示することが可能になります。以下のトピックは、ユーザー GUI に座標軸をプロットする方法を簡単に説明します。

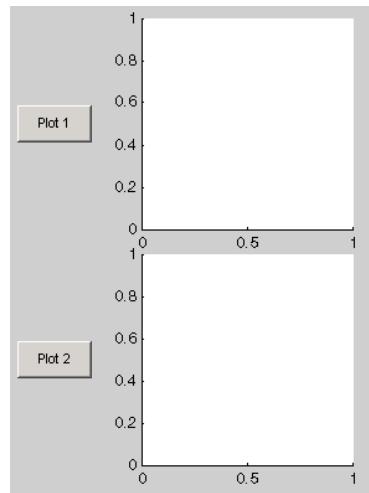
多くの場合、GUI の座標軸以外のコンポーネントに属するコールバックから、座標軸にプロットを作成します。たとえば、ボタンを押すと座標軸にグラフがプロットされます。この場合、ボタンの Callback コールバックはプロットを生成するコードを含みます。

次の例は、2 つの座標軸と 2 つのプッシュ ボタンを含みます。1つ目のボタンをクリックすると、1つの座標軸に contour プロットを作成し、他のボタンをクリックすると、他の座標軸に surf プロットを作成します。この例は、関数 peaks を用いてプロットするデータを作成します。この関数は、ガウス分布のトランスペンドとスケーリングで得られる正方行列を返します。

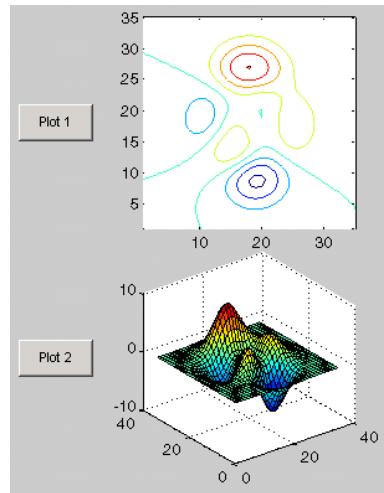
1 このコードを two_axes.m という M ファイルに保存してください。

```
function two_axes
fh = figure;
bh1 = uicontrol(fh,'Position',[20 290 60 30],...
    'String','Plot 1',...
    'Callback',@button1_plot);
bh2 = uicontrol(fh,'Position',[20 100 60 30],...
    'String','Plot 2',...
    'Callback',@button2_plot);
ah1 = axes('Parent',fh,'units','pixels',...
    'Position',[120 220 170 170]);
ah2 = axes('Parent',fh,'units','pixels',...
    'Position',[120 30 170 170]);
%-----
function button1_plot(hObject, eventdata)
    contour(ah1, peaks(35));
end
%-----
function button2_plot(hObject, eventdata)
    surf(ah2, peaks(35));
end
end
```

2 コマンド ラインで `two_axes` とタイプして GUI を実行します。ユーザーがプッシュ ボタンをクリックする前には、例は次のようにになっています。



3 Plot 1 ボタンをクリックして、第 1 軸に等高線図を表示します。Plot 2 ボタンをクリックして、第 2 軸に `surf` プロットを表示します。



2つの座標軸を使用するより複雑な例については、“複数の座標軸をもつ GUI”(p.10-2)を参照してください。

GUI が座標軸を含む場合、座標軸の HandleVisibility プロパティを callback に設定してください。これにより、コールバックが座標軸のコンテンツを変更でき、コマンドライン操作で変更されないようにします。既定値は on です。

座標軸に何かを描く場合、GUI のコードは使用する座標軸のハンドルを指定する必要があります。現在の Figure または対象の Axes が、その HandleVisibility プロパティを 'on' に設定していない場合は Figure を作成できるので、この目的で gca を使用しないでください。詳細は、MATLAB Graphics ドキュメンテーションの “Specifying the Target for Graphics Output” を参照してください。

ヒント 複数の座標軸を取り扱っている場合、データをプロットしようとしている座標軸を、次のようなコマンドで“現在の”座標軸にすることはしない方が良いでしょう。

```
axes(a1)
```

これにより、座標軸 a1 が現在の座標軸になりますが、これは Figure をリストックしたり、すべての未解決のイベントを書き出することもします。これは、コンピューターの資源を消費しますが、コールバックの実行に必要になることはほとんどありません。以下のように、ユーザーが呼び出しているプロット関数の最初の引数として単に座標軸のハンドルを与えると、より効率的です。

```
plot(a1, ...)
```

これは、Figure をリストックしたり、あるいはキューされたイベントを書き出さずに、座標軸 a1 にグラフィックスを出力します。関数 line のように Axes のハンドルを引数としないプロット関数に対して座標軸を指定する場合、以下のように a1 を現在の座標軸にすることができます。

```
set.figure_handle, 'CurrentAxes', a1)
line(x, y, z, ...)
```

詳細は、Figure プロパティ のリファレンス ページの CurrentAxes の説明を参照してください。

詳細は、以下を参照してください。

- ・ 座標軸の動作と外見の多くの特性をコントロールするために、設定できるプロパティについては、MATLAB Graphics ドキュメンテーションの “Axes Properties” を参照してください。
- ・ タイル模様の座標軸の作成は、関数 `subplot` のリファレンス ページを参照してください。
- ・ 一般のプロットについては、MATLAB Graphics ドキュメンテーションの “Plots and Plotting Tools” を参照してください。

ActiveX コントロールのプログラミング

ActiveX コントロールのプログラミングの詳細は、MATLAB External Interfaces ドキュメンテーションの以下のトピックスを参照してください。

- ・ “Responding to Events – an Overview”
- ・ “Writing Event Handlers”

一般的な情報は、MATLAB「外部インターフェイス」ドキュメンテーションの “MATLAB COM Client Support” を参照してください。

メモ ActiveX コントロールは、サイズを変更するメソッドを提供しません。ActiveX コントロールを用いて GUI を作成していて GUI とコントロールのサイズを変更可能にする場合、MATLAB External Interfaces ドキュメンテーションの “MATLAB Figure” に述べるサイズ変更の手法を使用できます。

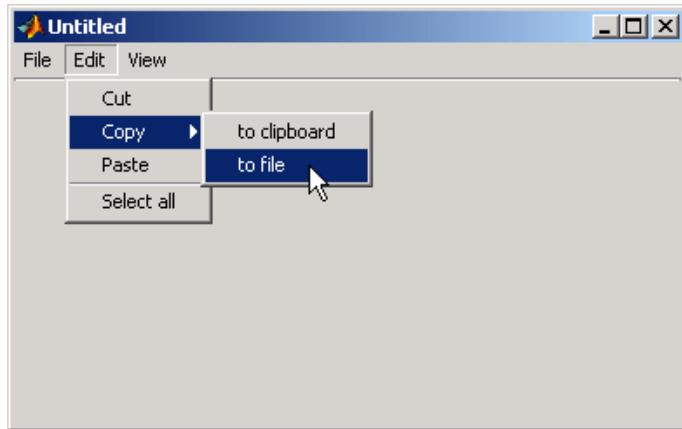
メニュー項目のプログラミング

- ・ “メニュータイトルのプログラミング” (p.12-34)
- ・ “メニュー コールバックからダイアログ ボックスを開く” (p.12-35)
- ・ “メニュー項目のチェックの更新” (p.12-36)

メニュータイトルのプログラミング

メニュー タイトルをクリックすると、サブメニューが自動的に表示されるので、ユーザーは、タイトル レベルのメニューに対するコールバックをプログラムする必要はありません。しかし、メニュー タイトルに関連するコールバックは、サブメニューを有効にしたり無効にするために使用できます。

次の図に示す例を考えます。



Edit > Copy > to file を選択すると、アクション実行のために Copy コールバックは必要ありません。to file 項目に関連する Callback コールバックのみが必要です。

ここでは、あるオブジェクトのみがファイルにコピーされると仮定します。ユーザーは Copy メニューのコールバックを使用して、選択されるオブジェクトのタイプに依存して、to file 項目を使用可能、あるいは使用不可能にすることができます。

次のコードは、to file 項目の Enable プロパティを off に設定して、この項目を無効にします。すると、メニュー項目がグレー表示されます。

```
set(tofilehandle, 'Enable', 'off')
```

Enable を on に設定すると、メニュー項目は有効になります。

メニュー コールバックからダイアログ ボックスを開く

to file メニュー項目に対する Callback コールバックは、次のようなコードを含み、ファイルを保存するための標準のダイアログ ボックスを表示します。

```
[file, path] = uiputfile('animinit.m', 'Save file name');
```

'Save file name' はダイアログ ボックスタイトルです。ダイアログ ボックスでは、ファイル名のフィールドに animinit.m が設定され、フィルタに M files (*.m) が設定されます。詳細は、uiputfile のリファレンス ページを参照してください。

メモ MATLAB では、1 度の関数呼び出しで作成できる標準的なダイアログ ボックスを選択できます。これらのダイアログ ボックスと作成に使用する関数についての詳細は、MATLAB 関数リファレンス ドキュメンテーションの “Predefined Dialog Boxes” を参照してください。

メニュー項目のチェックの更新

チェックはメニュー項目の現在の状態を表すのに非常に有効です。メニュー項目を作成するときに Checked プロパティを on に設定すると、項目は最初にチェックされた状態で表示されます。ユーザーがメニュー項目を選択する度に、その項目に対するコールバックはチェックを on あるいは off にする必要があります。次の例は、メニュー項目の Checked プロパティの値を変更することでこれを行う方法を示します。

```
function menu_copyfile(hObject, eventdata)
if strcmp(get(hObject,'Checked'), 'on')
    set(hObject,'Checked','off');
else
    set(hObject,'Checked','on');
end
```

`hObject` はイベントがトリガーされたコンポーネントのハンドルです。ここでは、メニュー項目の Callback プロパティがコールバックを関数ハンドルとして指定することを仮定して、使用します。詳細は、“コンポーネントに対するコールバックを与える”(p.12-11)を参照してください。

関数 `strcmp` は 2 つの文字列を比較し、同じ場合には論理値 1 (`true`) を返し、そうでない場合は論理値 0 (`false`) を返します。

GUI が最初に表示されるときのチェックの利用は、表示と一致しなければなりません。たとえば、ユーザー GUI に、最初に GUI を開いたときに表示される座標軸があり、GUI が Show axes メニュー項目をもつとき、メニュー項目を作成するときにメニュー項目の Checked プロパティを必ず on に設定して、Show axes メニュー項目の隣に最初にチェックが現れるようにします。

ツールバー ツールのプログラミング

- ・ “プッシュ ツール”(p.12-37)

- ・ “トグル ツール” (p.12-39)

プッシュツール

プッシュツールの ClickedCallback プロパティは、プッシュツールコントロール アクションを指定します。次の例は、プッシュツールを作成し、標準カラー選択ダイアログ ボックスを開くようにプログラムします。ダイアログ ボックスを使用して、GUI の背景色を設定できます。

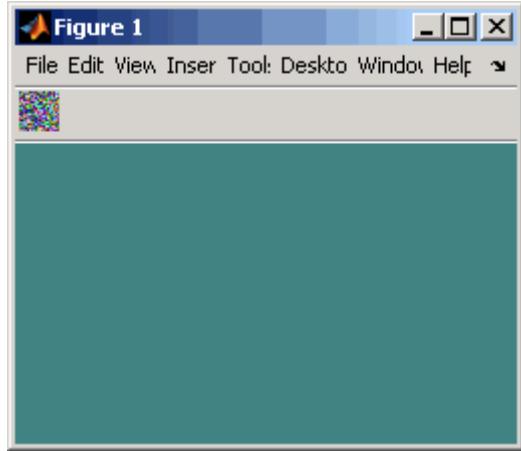
- 1 次のコードを M ファイルにコピーし、現在のフォルダーまたはユーザー パス上に color_gui.m として保存します。コマンド ラインで、color_gui と入力して関数を実行します。

```
function color_gui
    fh = figure('Position',[250 250 250 150], 'Toolbar', 'none');
    th = uiboolbar('Parent', fh);
    pth = uipushtool('Parent', th, 'Cdata', rand(20, 20, 3), ...
        'ClickedCallback', @color_callback);
    %-----
    function color_callback(hObject, eventdata)
        color = uisetcolor(fh, 'Pick a color');
    end
end
```

2 プッシュ ツールをクリックして、カラー選択ダイアログ ボックスを表示し、色をクリックして選択します。



3 カラー選択ダイアログ ボックスの OK をクリックします。GUI 背景色は、選択した色（この場合は緑）に変更します。



メモ “アイコン エディター”(p.15–60)に述べるアイコン エディターを用いてユーザー独自のアイコンを作成します。行列 X と対応するカラーマップ、すなわち、(X , MAP) イメージから RGB (トゥルーカラー) 形式への変換の詳細は、`ind2rgb` のリファレンス ページを参照してください。

トグル ツール

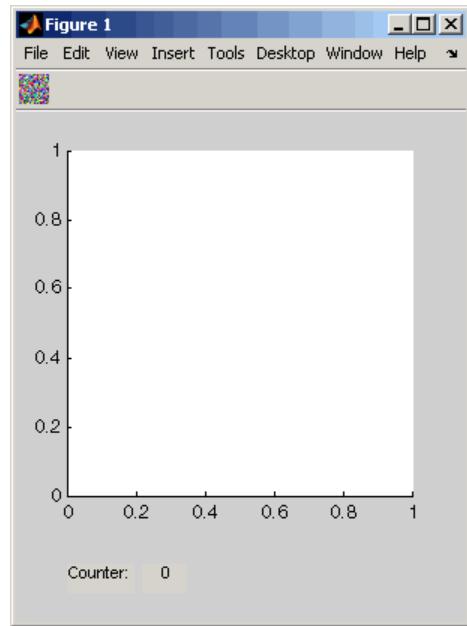
トグル ツールの `OnCallback` プロパティと `OffCallback` プロパティは、トグル ツール がクリックされたり、その `State` プロパティが `on` または `off` に変更されたときに起こる、トグル コントロール アクションを指定します。トグル ツール `ClickedCallback` プロパティは、トグル ツールがクリックされると、状態にかかわらず、起こるコントロール アクションを指定します。

次の例は、トグル ツールを用いて、`peaks` データの表示、表面とメッシュを切り替えます。例では、トグル ツールをクリックした回数もカウントします。

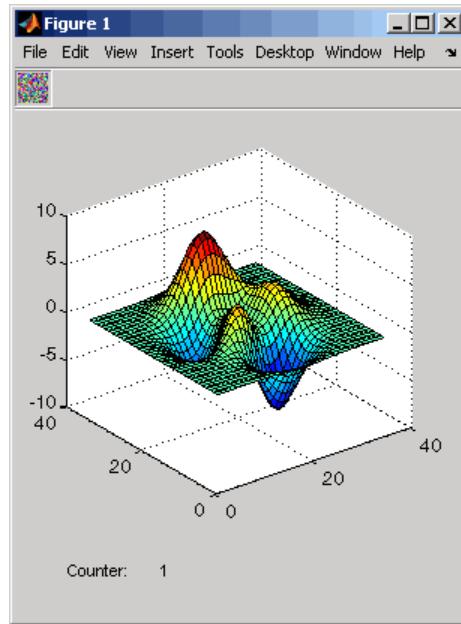
関数 `surf` は 3 次元の影付き表面プロットを作成します。関数 `mesh` は、ワイヤーフレーム パラメトリック表面を作成します。`peaks` は、ガウス分布のトランスレートとスケーリングによって得られる正方行列を返します。

1 次のコードを M ファイルにコピーし、現在のフォルダーまたはユーザー パス上に `toggle_plots.m` として保存します。コマンド ラインで `toggle_plots` と入力して、関数を実行します。

```
function toggle_plots
    counter = 0;
    fh = figure(' Position' ,[250 250 300 340], ' Toolbar' , ' none' );
    ah = axes(' Parent' , fh, ' Units' , ' pixels' ,...
        ' Position' ,[35 85 230 230]);
    th = uitoolbar(' Parent' , fh);
    tth = uitoggletool(' Parent' , th, ' Cdata' , rand(20,20,3),...
        ' OnCallback' , @surf_callback, ...
        ' OffCallback' , @mesh_callback, ...
        ' ClickedCallback' , @counter_callback);
    sth = uicontrol(' Style' , ' text' , ' String' , ' Counter: ' ,...
        ' Position' ,[35 20 45 20]);
    cth = uicontrol(' Style' , ' text' , ' String' , num2str(counter),...
        ' Position' ,[85 20 30 20]);
    %
    function counter_callback(hObject, eventdata)
        counter = counter + 1;
        set(cth, ' String' , num2str(counter))
    end
    %
    function surf_callback(hObject, eventdata)
        surf(ah, peaks(35));
    end
    %
    function mesh_callback(hObject, eventdata)
        mesh(ah, peaks(35));
    end
end
```



2 トグル ツールをクリックして、初期プロットを表示します。カウンターが増えて 1 になります。



3 トグル ツールをクリックを続け、peaks データの surf プロットと mesh プロットの切り替えをします。

アプリケーション定義のデータの管理

- ・ “データ管理の仕組み” (p.13-2)
- ・ “GUI のコールバック間でのデータの共有” (p.13-11)

データ管理の仕組み

このセクションの内容…

- “概要” (p.13-2)
- “入れ子関数” (p.13-4)
- “UserData プロパティ” (p.13-5)
- “アプリケーション データ” (p.13-6)
- “GUI データ” (p.13-8)

概要

大部分の GUI は、アプリケーションに固有のデータを作成または使用します。GUI コンポーネントは、互いにデータをやりとりする必要がある場合があります。そのために、役立つ基本的な仕組みがいくつかあります。

多くの GUI は単独の Figure ですが、ユーザー アプリケーションが 2 つ以上の Figure を必要とする場合には、複数の GUI を同時に機能させることができます。たとえば、GUI で使用されるパラメーターをいくつか表示して取得するために、GUI は複数のダイアログ ボックスを必要とします。ユーザー GUI は、同時にまたは連続して、同時に機能する個々のツールを複数含むことができます。ファイルまたはワーカースペース変数でのやりとりを回避するために、以下の表で述べるメソッドのいずれかを使用できます。

データ共有方法	動作	利用
プロパティ/値の組	入力引数として渡すことで、新たに呼び出された GUI または既存の GUI にデータを送ります。	新規の GUI にデータを送ります
出力	起動した GUI からデータを出力します。	起動された GUI のハンドル構造体を戻すなど、起動された GUI にデータを戻します

データ共有方法	動作	利用
関数ハンドルまたはプライベートデータ	以下の 4 つのメソッドのいずれか 1 つで、関数ハンドルまたはデータを渡します。	GUI 内または GUI 間で機能を表します
	“Nested Functions”: すべての上位の関数の名前空間を共有します	通常は、単一の GUI の Figure 内の、直接的または間接的に囲まれている関数で定義された変数へのアクセスと修正をします
	UserData:この Figure またはコンポーネントのプロパティにデータを格納します。ハンドル参照により、他の GUI とやりとりします。	GUI 内または GUI 間でのデータのやりとり。UserData は、構造体として渡される場合が多く、1 つの変数に制限されます
	アプリケーションデータ (getappdata / setappdata, ...):Figure またはコンポーネントに、名前の付いたデータを格納します。ハンドル参照により、他の GUI とやりとりします。	GUI 内または GUI 間でのデータのやりとり。変数の数やタイプは、この API を利用してアプリケーションデータとして格納できます
	guidata:GUI の handles 構造体に、データを格納します。ハンドル参照により、他の GUI とやりとりします。	GUI 内または GUI 間でのデータのやりとり – アプリケーションデータを管理する簡単な方法。GUI データは、GUI の Figure に関連する構造体です。

例 “アイコン エディター” (p.15-60) では、さらに GUI の Figure 間でのデータの共有を説明します。

以下の 3 つの節では、GUI 内に格納されたアプリケーション定義のデータを管理する方法を与える仕組みについて説明します。

- 入れ子関数 – より高いレベルで定義された変数を共有し、呼び出された関数が、上位か下位、または呼び出し側の兄弟である場合、互いに呼び出します。

- UserData プロパティ – GUI コンポーネントに割り当て、その他のプロパティのように取り出す MATLAB ワークスペース変数。
- アプリケーション データ – アプリケーションが、指定したオブジェクトに関連するデータを格納し取得するために利用します。GUI に対し、この特定のオブジェクトとは通常 GUI の Figure ですが、任意のコンポーネントでも可能です。データは、名前/値の組として保存されます。アプリケーション データを使うと、オブジェクトに対するユーザー定義のプロパティの作成が可能になります。
- GUI データ – GUI データを管理するために関数 `guidata` を使用する。この関数は MATLAB 構造体に GUI データとして 1 つの変数を格納できます。この構造体は GUIDE ではハンドル構造体と呼ばれます。ハンドル構造体を更新したり、取得したりするために関数を使用します。

“GUI のコールバック間でデータを共有する例”(p.9-10)において、簡単な作業 GUI に適用される 3 つの方法を比較できます。

入れ子関数

GUI M ファイルに入れ子関数を置くと、コールバック関数は、データを引数として渡される必要がなくなり、データを自由に共有することが可能になります。

- 1 ユーザー コードの初期化セグメントでは、コンポーネントの作成、変数の定義、データの生成をします。
- 2 GUI コールバックとユーティリティ関数を初期化の下のレベルに入れ子します。

データとコンポーネント ハンドルは、より上のレベルで定義されているので、コールバックとユーティリティ関数はこれらへのアクセスを自動的にもちます。この手法を利用すると、UserData、アプリケーション データ、または多くのインスタンスの GUI データを格納する必要がなくなります。

メモ 入れ子関数の利用に適用される規則と制約については、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”を参照してください。

完成した例は、“入れ子関数を用いてデータを共有する”(p.13-11)を参照してください。

UserData プロパティ

メニュー や Figure 自体を含む、すべての GUI コンポーネントは UserData プロパティ をもちます。UserData プロパティ の値として MATLAB ワークスペース の有効な 値を割り当てる こ とができますが、一度に存 在できる 値は 1 つに限られます。そのデータを 取得するには、そのデータが格納されている コールバックがコンポーネント のハンドルを 知る 必要があります。適切なオブジェクト のハンドルを用いて get と set を利用して、UserData にアクセスします。以下の例では、このパターンを説明します。

- 1 エディット テキスト コンポーネントは、その UserData プロパティ にユーザーが入力する 文字列を格納します。

```
function edittext1_callback(hObject, eventdata)
mystring = get(hObject, 'String');
set(hObject, 'UserData', mystring);
```

- 2 プッシュ ボタンは、エディット テキスト コンポーネント の UserData プロパティ から 文字列を取得します。

```
function pushbutton1_callback(hObject, eventdata)
string = get(edittextHandle, 'UserData');
```

たとえば、メニュー項目が [元に戻す] である場合、そのコードは、edittext1 の String をリセットして、その UserData に格納された 値に 戻す こ とができます。元に戻す 操作を容易にするために、UserData は、スタックまたは循環バッファとして扱われる、文字列のセル配列になる こ とができます。

複数の変数を保存したい場合、UserData を構造体として指定します。ユーザーが定義する各フィールドは、異なる変数をもつ こ とができます。

メモ `hObject` (呼び出しているオブジェクトのハンドル) を使用するには、文字列や関数名ではなく、コンポーネントのコールバック プロパティを関数ハンドルとして指定しなければなりません。このように指定すると、コンポーネントのハンドルは `hObject` として各コールバックに自動的に渡されます。詳細は、“コンポーネントに対するコールバックを与える”(p.12-11)を参照してください。

完成した例は、“データを UserData と共有する”(p.13-16)を参照してください。

アプリケーション データ

アプリケーション データは、ユーザー アプリケーションにとって重要なデータやユーザー アプリケーションで定義されるデータです。関数 `setappdata` と `getappdata` を用いて、Figure または GUI コンポーネント (ActiveX コントロール以外) にアプリケーション データを追加します。アプリケーション データと UserData の主な違いは以下のとおりです。

- ・ アプリケーション データには複数のデータを割り当てるすることができますが、`UserData` に割り当てることができる値は 1 つに限られます。
- ・ コードは、(Tag を使用する場合のように) 名前で、アプリケーション データを参照する必要がありますが、他のプロパティのように `UserData` にアクセスできます。

MATLAB の Handle Graphics オブジェクトのみが、このプロパティを使用します。次の表は、アプリケーション データへのアクセスを提供する関数をまとめています。詳細は、各関数のリファレンス ページを参照してください。

アプリケーション データを管理するための関数

関数	目的
<code>setappdata</code>	オブジェクト (GUI の Figure または他の Handle Graphics オブジェクト) に対して、名前の付いたアプリケーション データを指定します。1 つのオブジェクトに対して名前付きのアプリケーション データの項目を複数指定することができます。ただし、各名前はそのオブジェクトに対して固有でなければならず、また、1 つの値のみ関連付けられます。通常は構造体と関連付けられます。
<code>getappdata</code>	名前付きのアプリケーション データを取得します。名前付きのアプリケーション データを取得するには、アプリケーション データに関連する名前、およびアプリケーション データが関連付けられたオブジェクトのハンドルを知る必要があります。ハンドルのみを指定すると、オブジェクトのアプリケーション データすべてが返されます。
<code>isappdata</code>	名前の付いたアプリケーション データが指定したオブジェクトに存在する場合は、真。
<code>rmappdata</code>	名前の付いたアプリケーション データを指定したオブジェクトから削除します。

アプリケーション データの作成

関数 `setappdata` を使用して、アプリケーション データを作成します。この例は、正規分布をした乱数からなる 35×35 行列を作成し、この行列を管理するために、Figure と関連するアプリケーション データ `mydata` を作成します。

```
matrices.rand_35 = randn(35);
setappdata(figurehandle, 'mydata', matrices);
```

アプリケーション データ構造体へのフィールドの追加

アプリケーション データは、通常必要に応じて、フィールドの追加が可能な構造体として定義されます。この例では、前のトピックで作成されたアプリケーション データ構造体 `mydata` にフィールドを追加します。

1 `getappdata` を使用して構造体を取得します。

前のトピックの例から、アプリケーション データ構造体の名前は `mydata` です。これは、Figure と関連しています。

```
matrices = getappdata(figurehandle, 'mydata');
```

2 新しいフィールドを作成し、値を割り当てます。たとえば、

```
matrices.randn_50 = randn(50);
```

は、`matrices` 構造体にフィールド `randn_50` を追加し、正規分布を示す乱数から構成される 50×50 行列に設定します。

3 `setappdata` を使用してデータを保存します。この例は、`setappdata` を使用して、アプリケーション データ構造体 `mydata` として `matrices` 構造体を保存します。

```
setappdata(figurehandle, 'mydata', matrices);
```

コールバックは、このアプリケーション データを同じ方法で取得（さらに変更）できますが、アクセスするために Figure ハンドルを知る必要があります。最上位レベルで、入れ子関数を使用したり、Figure を作成すると、Figure ハンドルには、より低階で入れ子になっているすべてのコールバックとユーティリティ関数がアクセスできます。入れ子関数の詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”について参照してください。アプリケーション データの利用についての完成した例は、“アプリケーション データとデータを共有する”(p.13-18)を参照してください。

GUI データ

大部分の GUI は、アプリケーションに固有のデータを作成または使用します。これらの仕組みは、アプリケーションが GUI に保存されたデータを保存、取得する方法を提供します。GUI データでは、

- ・ コンポーネントのハンドルを用いると、Figure ハンドルを見つける必要なくコールバック ルーチン内からデータにアクセスできます。
- ・ ユーザーのソース コードの中で、データに対するハードコードされた名前の作成や保持の必要はありません。

関数 `guidata` を使用して、GUI データを管理します。この関数は、1 つの変数を GUI データとして保存できます。GUI データは、以下の点でアプリケーション データとは異なります。

- ・ GUI データは 1 つの変数です。ただし、構造体として定義されると、ユーザーはフィールドの追加や削除を行うことができます。
- ・ アプリケーション データは多くの変数で構成できます。各変数は、別々の 1 つしかない名前で格納されます。
- ・ ユーザーは、関数 `guidata` を用いて GUI データにアクセスします。この関数は GUI データの格納と取得を行います。
- ・ GUI データを格納するために `guidata` を使用すると、既存の GUI データが上書きされます。
- ・ 関数 `getappdata`、`setappdata`、`rmappdata` を使用することは、GUI データに影響しません。

GUI データは、常に、GUI の Figure と関連します。これは、GUI のすべてのコンポーネントのすべてのコールバックに対して利用できます。GUI データを保存、または、取得するときにコンポーネントのハンドルを指定すると、MATLAB はデータをコンポーネントの親の Figure と自動的に関連させます。

GUI データは、いつでも 1 変数のみしか含むことができません。GUI データの書き込みは、既存の GUI データを上書きします。このため、GUI データは、通常、必要に応じて GUI データにフィールドを追加することができる構造体として定義されます。

コンポーネントのハンドルを使用して、コールバック ルーチン内からデータにアクセスすることができ、Figure ハンドルを探す必要はありません。コンポーネントのコールバックプロパティを関数ハンドルとして指定する場合、コンポーネントのハンドルは各

コールバックに、`hObject` として自動的に渡されます。詳細は、“コンポーネントに対するコールバックを与える”(p.12-11)を参照してください。

GUI データ変数は 1 つに限られ、`Figure` と関連するので、ソースコード全体を通してデータに対するハードコードされた名前を作成したり保持する必要がありません。

メモ GUIDE は GUI データを用いて GUI のハンドル構造体を使用し、各コールバックに引数 (`handles` と呼ばれる) として含みます。プログラミングによる GUI コールバックは GUI データを含みませんが、コールバック関数は、コンポーネントのハンドル (コールバックの最初の引数 `hObject`) から GUI データにアクセスできます。ユーザー M ファイルがもともと GUIDE で作成されていた場合は、“GUIDE で作成された M ファイルの GUI データの変更”(p.9-9) を参照してください。

GUI データの作成と更新

1 構造体を作成し、構造体にフィールドを追加します。たとえば、

```
mydata.iteration_counter = 0;
mydata.number_errors = 0;
```

2 GUI データとして構造体を保存します。MATLAB は、GUI データを `Figure` と関連させますが、ハンドルの取得や保存のために、`Figure` 内の任意のコンポーネントのハンドルを使用できます。

```
guidata(hObject, mydata);
```

3 コールバックからの GUI データを変更するには、構造体のコピーを取得し、適切なフィールドを更新し、GUI データを保存します。

```
mydata = guidata(hObject); % Get the GUI data.
mydata.iteration_counter = mydata.iteration_counter +1;
guidata(hObject, mydata); % Save the GUI data.
```

メモ `hObject` を使用するには、コンポーネントのコールバック プロパティを関数ハンドルとして指定しなければなりません。このように指定すると、コンポーネントのハンドルは `hObject` として各コールバックに自動的に渡されます。詳細は、“コンポーネントに対するコールバックを与える”(p.12-11)を参照してください。

GUI データ構造体へのフィールドの追加

GUI データ構造体にフィールドを追加するには、次のようにします。

- 1 次のようなコマンドで構造体のコピーを取得します。`hObject` は、コールバックがトリガーされたコンポーネントのハンドルです。

```
mydata = guidata(hObject)
```

- 2 新しいフィールドに値を割り当て、構造体にフィールドを追加します。たとえば、

```
mydata.iteration_state = 0;
```

は、構造体 `mydata` にフィールド `iteration_state` を追加し、そのフィールドを 0 に設定します。

- 3 データを保存するには、次のコマンドを使用します。

```
guidata(hObject, mydata)
```

ここで、`hObject` は、コールバックがトリガーされるコンポーネントのハンドルです。MATLAB は、`mydata` 構造体の新しいコピーをコンポーネントの親の Figure と関連させます。

完成した例は、“GUI データとデータを共有する”(p.13-21)を参照してください。

GUI のコールバック間でのデータの共有

このセクションの内容…

“入れ子関数を用いてデータを共有する” (p.13-11)

“データを UserData と共有する” (p.13-16)

“アプリケーション データとデータを共有する” (p.13-18)

“GUI データとデータを共有する” (p.13-21)

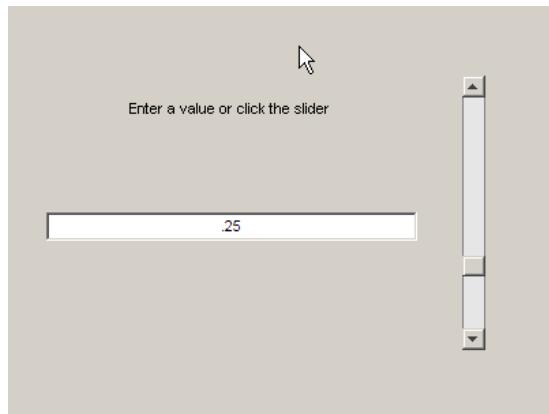
以下の 4 つの節それぞれには、例の GUI の完成したコードがあります。そのコードは M ファイルにコピーして実行できます。これらのメソッドの一般の情報は、“データ管理の仕組み” (p.13-2) を参照してください。

入れ子関数を用いてデータを共有する

GUI のコールバック間でデータを共有するために、GUI データ、アプリケーション データ、UserData プロパティを使用することができます。多くの場合、入れ子関数によって、他のデータ形式を使用せずにコールバック間でデータを共有できます。

入れ子関数の例:コンポーネント間でのデータ渡し

この例は、スライダーとエディット テキストを含む GUI を使用します。スタティック テキストは、エディット テキストに値を入力するか、スライダーをクリックするようユーザーに指示します。この例は、入れ子関数の環境で、スライダーの旧い値、新しい値、エラー カウンターを初期化し保持します。



GUI の動作は、以下のようになります。

- スライダーを移動させると、エディット テキスト コンポーネントは、スライダーの現在の値を表示し、スライダーが以前の位置からどれだけ移動したかを示すメッセージを、コマンド ウィンドウに表示します。

You changed the slider value by 24.11 percent.

- エディット テキスト コンポーネントに値を入力して、Enter キーを押すか、あるいはコンポーネントの外側をクリックすると、スライダーは入力された値を更新し、エディット テキスト コンポーネントは、スライダーが移動した単位数を示すメッセージをコマンド ウィンドウに表示します。
- スライダーに対しての範囲外の値（つまり、スライダーの Min と Max プロパティの間の値ではない値）をエディット テキスト コンポーネントに入力すると、アプリケーションはユーザーが誤った値を入力した回数を示すメッセージを、エディット テキスト コンポーネントに出力します。



次のコードは、コンポーネントを作成し、関数内の初期化セクションにおいて、エラー カウンター、およびスライダーの旧い値と新しい値を初期化します。さらに、2 つのコールバックを使用して、スライダーとエディット テキスト コンポーネント間の入れ換えを実現します。スライダーのコールバックとテキスト エディット のコールバックは、メイン関数内で入れ子になっています。

次のコードリストをコピーして新規の M ファイルに貼り付け、この M ファイルを現在のフォルダー、またはユーザー パス上のどこかに `slider_gui_nested.m` として保存できます。あるいは、ここをクリックして現在のフォルダーに `slider_gui_nested.m` を置きます。コマンド ラインで `slider_gui_nested` と入力して関数を実行します。

```

function slider_gui_nested
fh = figure(' Position' ,[250 250 350 350],...
    'MenuBar' , ' none' , ' NumberTitle' , ' off' ,...
    ' Name' , ' Sharing Data with Nested Functions' );
sh = uicontrol(fh,' Style' , ' slider' ,...
    ' Max' ,100,' Min' ,0,' Value' ,25, ...
    ' SliderStep' , [0.05 0.2],...
    ' Position' ,[300 25 20 300],...
    ' Callback' ,@slider_callback);
eth = uicontrol(fh,' Style' , ' edit' ,...
    ' String' ,num2str(get(sh,' Value' )),...
    ' Position' ,[30 175 240 20],...
    ' Callback' ,@edittext_callback);
sth = uicontrol(fh,' Style' , ' text' , ' String' ,...
    ' Enter a value or click the slider.' ,...
    ' Position' ,[30 215 240 20]);
number_errors = 0;
previous_val = 0;
val = 0;
% -----First Nested Function-----
% Set the value of the edit text component String property
% to the value of the slider.
function slider_callback(hObject, eventdata)
    previous_val = val;
    val = get(hObject,' Value' );
    set(eth,' String' ,num2str(val));
    sprintf(' You changed the slider value by %6.2f percent.' ,...
        abs(val - previous_val))
end
% -----Second Nested Function-----
% Set the slider value to the number the user types in
% the edit text or display an error message.
function edittext_callback(hObject, eventdata)
    previous_val = val;
    val = str2double(get(hObject,' String' ));

```

```
% Determine whether val is a number between the
% slider's Min and Max. If it is, set the slider Value.
if isnumeric(val) && length(val) == 1 && ...
    val >= get(sh,'Min') && ...
    val <= get(sh,'Max')
    set(sh,'Value',val);
    sprintf(' You changed the slider value by %6.2f percent.',...
            abs(val - previous_val))
else
% Increment the error count, and display it.
    number_errors = number_errors+1;
    set(hObject,'String',...
        [ ' You have entered an invalid entry ',...
          num2str(number_errors), ' times.' ]);
    val = previous_val;
end
end
end
```

これらのコンポーネントは最上位レベルに作成されているので、それらのハンドルは、そのルーチンのより下位レベルで入れ子になっているコールバックが直接利用できます。同じことは、エラー カウンター `number_errors`、以前のスライダー値 `previous_val`、新しいスライダー値 `val` に対しても成り立ちます。これらの変数を引数として渡す必要はありません。

両方のコールバックは、入力引数 `hObject` を使用して、コールバックの実行をトリガーするコンポーネントのプロパティを取得し設定します。コンポーネントの `Callback` プロパティは関数ハンドルとして指定されているので、コールバックはこの引数を利用できます。詳細は、“コンポーネントに対するコールバックを与える”(p.12-11)を参照してください。

スライダー コールバック. スライダー コールバック `slider_callback` は、エディット テキストコンポーネント ハンドル `eth` を使用して、エディット テキストの '`String`' プロパティをユーザーがタイプした値に設定します。

スライダーの `Callback` は、新しい値を `val` に割り当てる前に、`previous_val` のスライダーの以前の値 `val` を保存します。これらの変数は、より高いレベルで初期化されるので、両方のコールバックに識別されます。これらは、いずれかのコールバックによって、取得され設定可能です。

```
previous_val = val;
val = get(hObject,'Value');
```

スライダー コールバックの次のステートメントは、ユーザーがスライダーを動かすか、マウス ボタンをはなすときに、エディット テキストコンポーネントに表示された値を更新します。

```
val = get(hObject,'Value');
set(eth,'String',num2str(val));
```

これらのコードは、3 つのコマンドを連結します。

- `get` は、スライダーの現在の値を取得します。
- `num2str` は、値を文字列に変換します。
- `set` は、エディット テキストコンポーネントの `String` プロパティを設定し、値を更新します。

エディット テキスト コールバック. エディット テキストのコールバック、`edittext_callback` は、スライダー ハンドル `sh` を使用して、スライダーの `Max` と `Min` プロパティを決め、スライダーの溝（スライディング バーが移動する範囲内のバー上以外の部分）の位置を決める、スライダーの `Value` プロパティを設定します。

エディット テキストのコールバックは、ユーザーが入力した数値が許容範囲内の数値であるかどうかを調べるためにチェックした後、次のコードを使用して、スライダーの値として設定します。

```
if isnumeric(val) && length(val) == 1 && ...
    val >= get(sh,'Min') && ...
    val <= get(sh,'Max')
    set(sh,'Value',val);
```

値が範囲外にあると、エラー カウンター `number_errors` を増やし、無効な数を入力した回数をユーザーに伝えるメッセージを表示して、`if` ステートメントを続けます。

```
else
    number_errors = number_errors+1;
    set(hObject,'String',...
        [ ' You have entered an invalid entry ',...
        num2str(number_errors), ' times.' ]);
end
```

データを UserData と共有する

各 GUI コンポーネントと Figure 自身は、アプリケーション定義データを保存するために使用できる UserData プロパティをもちます。UserData にアクセスするには、コードバックは特定の UserData プロパティが関連しているコンポーネントのハンドルを知る必要があります。

関数 `get` を使用して UserData を取得し、関数 `set` を使用してそれを設定します。

UserData プロパティの例: コンポーネント間でのデータ渡し

次のコードは、以前の例 “入れ子関数を用いてデータを共有する” (p.13-11) のものと同じですが、UserData プロパティを使用して、エラー カウンターを初期化し増やします。コードではまた、入れ子関数を用いて、メイン関数が定義する他のコンポーネントのハンドルにアクセスしてコードバックも提供します。次のコード リストをコピーして新規の M ファイルに貼り付け、この M ファイルを現在のフォルダー、またはユーザー パス上のどこかに `slider_gui_userdata.m` として保存できます。あるいは、ここをクリックして、現在のフォルダーに `slider_gui_userdata.m` を置きます。コマンドラインで `slider_gui_userdata` とタイプして関数を実行します。

```
function slider_gui_userdata
fh = figure(' Position' ,[250 250 350 350],...
    'MenuBar' , ' none' , ' NumberTitle' , ' off' ,...
    ' Name' , ' Sharing Data with UserData' );
sh = uicontrol(fh,' Style' , ' slider' ,...
    ' Max' ,100,' Min' ,0,' Value' ,25, ...
    ' SliderStep' , [0.05 0.2],...
    ' Position' ,[300 25 20 300],...
    ' Callback' ,@slider_callback);
eth = uicontrol(fh,' Style' , ' edit' ,...
    ' String' ,num2str(get(sh,' Value' )),...
    ' Position' ,[30 175 240 20],...
    ' Callback' ,@edittext_callback);
sth = uicontrol(fh,' Style' , ' text' , ' String' ,...
    ' Enter a value or click the slider.' ,...
    ' Position' ,[30 215 240 20]);
number_errors = 0;
slider.val = 25;
% Set edit text UserData property to slider structure.
set(eth,' UserData' ,slider)
%
```

```
% Set the value of the edit text component String property
% to the value of the slider.
function slider_callback(hObject, eventdata)
    % Get slider from edit text UserData.
    slider = get(eth,'UserData');
    slider.previous_val = slider.val;
    slider.val = get(hObject,'Value');
    set(eth,'String',num2str(slider.val));
    sprintf(' You changed the slider value by %6.2f percent.' ,...
        abs(slider.val - slider.previous_val))
    % Save slider in UserData before returning.
    set(eth,'UserData',slider)
end
%
% -----
% Set the slider value to the number the user types in
% the edit text or display an error message.
function edittext_callback(hObject, eventdata)
    % Get slider from edit text UserData.
    slider = get(eth,'UserData');
    slider.previous_val = slider.val;
    slider.val = str2double(get(hObject,'String'));
    % Determine whether slider.val is a number between the
    % slider's Min and Max. If it is, set the slider Value.
    if isnumeric(slider.val) && ...
        length(slider.val) == 1 && ...
        slider.val >= get(sh,'Min') && ...
        slider.val <= get(sh,'Max')
        set(sh,'Value',slider.val);
        sprintf(' You changed the slider value by %6.2f percent.' ,...
            abs(slider.val - slider.previous_val))
    else
        % Increment the error count, and display it.
        number_errors = number_errors+1;
        set(hObject,'String',...
            [' You have entered an invalid entry' ,...
            num2str(number_errors), ' times.']);
        slider.val = slider.previous_val;
    end
    % Save slider structure in UserData before returning.
    set(eth,'UserData',slider)
```

```
    end  
end
```

スライダーの値. この例では、スライダーのコールバック `slider_callback` とエディットテキストのコールバック `edittext_callback` は、エディットテキストの `UserData` プロパティから構造体 `slider` を取得します。`slider` 構造体は、スライダーの以前の値と現在の値を保持します。すると、コールバックは、新しい値を取得して `slider.val` に割り当てる前に、値 `slider.val` を `slider.previous_val` に保存します。出力する前に、各コールバックは、エディットテキストの `UserData` プロパティに `slider` 構造体を保存します。

```
% Get slider structure from edit text UserData.  
slider = get(eth,'UserData',slider);  
slider.previous_val = slider.val;  
slider.val = str2double(get(hObject,'String'));  
...  
% Save slider structure in UserData before returning.  
set(eth,'UserData',slider)
```

両方のコールバックは、関数 `get` と関数 `set` を使用して、エディットテキストの `UserData` プロパティの `slider` 構造体を取得し保存します。

アプリケーション データとデータを共有する

アプリケーション データは、コンポーネント、メニュー、または `Figure` 自身などの任意のオブジェクトと関連付けできます。アプリケーション データにアクセスするには、コールバックはデータの名前とデータが格納されたコンポーネントのハンドルを知る必要があります。関数 `setappdata`、`getappdata`、`isappdata`、`rmappdata` を使用して、アプリケーション データを管理します。

アプリケーション データに関する詳細は、“アプリケーション データ” (p.13-6) を参照してください。

アプリケーション データの例: コンポーネント間でのデータ渡し

次のコードは、前の例に似ていますが、アプリケーション データを使用して、エディットテキストとスライダー Callback のスライダーの旧い値と新しい値を初期化し保持します。コードではまた、入れ子関数を用いて、メイン関数が定義する他のコンポーネントのハンドルにアクセスしてコールバックも提供します。次のコードリストをコピーして新規の M ファイルに貼り付け、この M ファイルを現在のフォルダー、またはユーザー パス上のどこかに `slider_gui_appdata.m` として保存できます。あるいは、現在

のフォルダーに slider_gui_appdata.m を置くために、ここをクリックしてください。コマンドラインで slider_gui_appdata とタイプして関数を実行します。

```

function slider_gui_appdata
fh = figure(' Position' ,[250 250 350 350],...
    ' MenuBar' , ' none' , ' NumberTitle' , ' off' ,...
    ' Name' , ' Sharing Data with Application Data' );
sh = uicontrol(fh,' Style' , ' slider' ,...
    ' Max' ,100,' Min' ,0,' Value' ,25, ...
    ' SliderStep' , [0.05 0.2],...
    ' Position' ,[300 25 20 300],...
    ' Callback' ,@slider_callback);
eth = uicontrol(fh,' Style' , ' edit' ,...
    ' String' ,num2str(get(sh,' Value' )),...
    ' Position' ,[30 175 240 20],...
    ' Callback' ,@edittext_callback);
sth = uicontrol(fh,' Style' , ' text' , ' String' ,...
    ' Enter a value or click the slider.' ,...
    ' Position' ,[30 215 240 20]);
number_errors = 0;
slider_data.val = 25;
% Create appdata with name ' slider' .
setappdata(fh,' slider' ,slider_data);
%
% Set the value of the edit text component String property
% to the value of the slider.
function slider_callback(hObject, eventdata)
% Get ' slider' appdata.
slider_data = getappdata(fh,' slider' );
slider_data.previous_val = slider_data.val;
slider_data.val = get(hObject,' Value' );
set(eth,' String' ,num2str(slider_data.val));
sprintf(' You changed the slider value by %6.2f percent.' ,...
    abs(slider_data.val - slider_data.previous_val))
% Save ' slider' appdata before returning.
setappdata(fh,' slider' ,slider_data)
end
%
% Set the slider value to the number the user types in
% the edit text or display an error message.

```

```
function edittext_callback(hObject, eventdata)
    % Get 'slider' appdata.
    slider_data = getappdata(fh,'slider');
    slider_data.previous_val = slider_data.val;
    slider_data.val = str2double(get(hObject,'String'));
    % Determine whether val is a number between the
    % slider's Min and Max. If it is, set the slider Value.
    if isnumeric(slider_data.val) && ...
        length(slider_data.val) == 1 && ...
        slider_data.val >= get(sh,'Min') && ...
        slider_data.val <= get(sh,'Max')
        set(sh,'Value',slider_data.val);
        sprintf(' You changed the slider value by %.2f percent.', ...
            abs(slider_data.val - slider_data.previous_val))
    else
        % Increment the error count, and display it.
        number_errors = number_errors+1;
        set(hObject,'String',...
            [ ' You have entered an invalid entry ',...
            num2str(number_errors), ' times.' ]);
        slider_data.val = slider_data.previous_val;
    end
    % Save appdata before returning.
    setappdata(fh,'slider',slider_data);
end
end
```

スライダーの値. この例では、スライダーのコールバック `slider_callback` とエディットテキストのコールバック `edittext_callback` は、スライダーの以前の値と現在の値を保持するアプリケーション データ構造体 `slider_data` を取得します。すると、これらは、新しい値を取得して `slider_data.val` に割り当てる前に、値 `slider_data.val` を `slider_data.previous_val` に保存します。出力する前に、各コールバックは `slider` アプリケーション データに `slider_data` 構造体を保存します。

```
% Get 'slider' appdata.
slider_data = getappdata(fh,'slider');
slider_data.previous_val = slider_data.val;
slider_data.val = str2double(get(hObject,'String'));
...
% Save 'slider' appdata before returning.
```

```
setappdata(fh,'slider',slider_data)
```

両方のコールバックは、関数 `getappdata` と関数 `setappdata` を使用して、`slider_data` 構造体を `slider` アプリケーション データとして取得し保存します。

GUI データとデータを共有する

関数 `guidata` によって管理される GUI データは、GUI のすべてのコールバックに対してアクセス可能です。1 つのコンポーネントに対するコールバックは、他のコンポーネントに対するコールバックから読むことができる GUI データに値を設定できます。詳細は、“GUI データ”(p.13-8)を参照してください。

GUI データの例:コンポーネント間でのデータ渡し

次のコードは、前のトピックのコードに似ていますが、GUI データを使用して、エディットテキストとスライダー Callback のスライダーの旧い値と新しい値を初期化し保持します。コードではまた、入れ子関数を用いて、メイン関数が定義する他のコンポーネントのハンドルにアクセスしてコールバックも提供します。次のコードリストをコピーして新規の M ファイルに貼り付け、この M ファイルを現在のフォルダー、またはユーザー パス上のどこかに `slider_gui_guidata.m` として保存できます。あるいは、現在のフォルダーに `slider_gui_guidata.m` を置くために、ここをクリックしてください。コマンド ラインで `slider_gui_guidata` とタイプして関数を実行します。

```
function slider_gui_guidata
fh = figure('Position',[250 250 350 350],...
    'MenuBar','none','NumberTitle','off',...
    'Name','Sharing Data with GUI Data');
sh = uicontrol(fh,'Style','slider',...
    'Max',100,'Min',0,'Value',25,...
    'SliderStep',[0.05 0.2],...
    'Position',[300 25 20 300],...
    'Callback',@slider_callback);
eth = uicontrol(fh,'Style','edit',...
    'String',num2str(get(sh,'Value')),...%
    'Position',[30 175 240 20],...
    'Callback',@edittext_callback);
sth = uicontrol(fh,'Style','text','String',...
    'Enter a value or click the slider.',...
    'Position',[30 215 240 20]);
number_errors = 0;
slider.val = 25;
```

```
guidata(fh, slider);
% -----
% Set the value of the edit text component String property
% to the value of the slider.
function slider_callback(hObject, eventdata)
    slider = guidata(fh); % Get GUI data.
    slider.previous_val = slider.val;
    slider.val = get(hObject, 'Value');
    set(hObject, 'String', num2str(slider.val));
    sprintf(' You changed the slider value by %6.2f percent. ', ...
        abs(slider.val - slider.previous_val))
    guidata(fh, slider) % Save GUI data before returning.
end
% -----
% Set the slider value to the number the user types in
% the edit text or display an error message.
function edittext_callback(hObject, eventdata)
    slider = guidata(fh); % Get GUI data.
    slider.previous_val = slider.val;
    slider.val = str2double(get(hObject, 'String'));
    % Determine whether slider.val is a number between the
    % slider's Min and Max. If it is, set the slider Value.
    if isnumeric(slider.val) && length(slider.val) == 1 && ...
        slider.val >= get(sh, 'Min') && ...
        slider.val <= get(sh, 'Max')
        set(sh, 'Value', slider.val);
        sprintf(' You changed the slider value by %6.2f percent. ', ...
            abs(slider.val - slider.previous_val))
    else
        % Increment the error count, and display it.
        number_errors = number_errors+1;
        set(hObject, 'String', ...
            [ ' You have entered an invalid entry ', ...
            num2str(number_errors), ' times.' ]);
        slider.val = slider.previous_val;
    end
    guidata(fh, slider); % Save the changes as GUI data.
end
end
```

スライダーの値. この例において、スライダーのコールバック `slider_callback` とエディットテキストのコールバック `edittext_callback` は、スライダーの以前の値と現在の値を保持する、GUI データ構造体 `slider` を取得します。すると、それらは、新しい値を取得して `slider.val` に割り当てる前に、値 `slider.val` を `slider.previous_val` に保存します。戻る前に、各コールバックは GUI データに対する `slider` 構造体を保存します。

```
slider = guidata(fh); % Get GUI data.  
slider.previous_val = slider.val;  
slider.val = str2double(get(hObject,'String'));  
...  
  
guidata(fh,slider) % Save GUI data before returning.
```

両方のコールバックは、関数 `guidata` を使用して、`slider` 構造体を GUI データとして取得し保存します。

コールバック実行の管理

コールバックの一時停止

このセクションの内容…

“コールバックの実行” (p.14-2)

“Interruptible プロパティの動作仕様” (p.14-2)

“Busy Action プロパティの動作仕様” (p.14-4)

“例” (p.14-4)

コールバックの実行

コールバックの実行はイベントドリブンであり、様々な GUI からのコールバックが同じイベントキューを共有します。一般に、コールバックは、マウスクリックまたはキープレスなどのユーザーイベントによってトリガーされます。このため、コールバックが要求される時間、別のコールバックが実行しているかどうか、別のコールバックがある場合には、それがどのコールバックであるかを予測することはできません。

あるコールバックの実行中に、別のコールバックが定義されているイベントをユーザーがトリガーすると、このコールバックは、既に実行しているコールバックを一時停止しようとします。この場合、MATLAB は次の要因に従ってコールバックを処理します。

- ・ すでにコールバックが実行中であるオブジェクトの Interruptible プロパティ。Interruptible プロパティは、実行しているコールバックが一時停止可能かどうかを指定します。
- ・ 直前にコールバックがトリガされ実行しようとするオブジェクトの、BusyAction プロパティ。BusyAction プロパティは、コールバックを待ち行列に追加するべきかどうかを指定し、実行の待ち受けまたはキャンセルを行います。

メモ コールバックとその動作の詳細は、このユーザー ガイドの「“コールバック:概要” (p.12-7)」、さらに MATLAB Graphics ドキュメンテーションの“Callback Properties for Graphics Objects”も参照してください。

Interruptible プロパティの動作仕様

オブジェクトの Interruptible プロパティは、on (既定値) または off になります。

コールバックを実行中のオブジェクトの `Interruptible` プロパティが `on` の場合、そのコールバックを一時停止できます。ただし、コールバックまたはコールバックがトリガーした関数が、`drawnow`、`figure`、`getframe`、`pause`、または `waitfor` を呼び出す場合に限り、コールバックを一時停止できます。これらの関数は、定義されたタスクを実行する前に、待機しているコールバックを含む、イベント キュー内のイベントを処理します。実行しているコールバック、またはそれがトリガーする関数が、これらの関数のいずれも呼び出さない場合には、そのオブジェクトの `Interruptible` プロパティの値によらず、コールバックを一時停止できません。

コールバックが実行中であるオブジェクトの `Interruptible` プロパティが `off` の場合、以下の例外を除きそのコールバックは一時停止できません。割り込みコールバックが `DeleteFcn` または `CreateFcn` コールバックであるか、あるいは `Figure` の `CloseRequest` または `ResizeFcn` コールバックである場合には、実行中のコールバックオブジェクトの `Interruptible` プロパティの値によらず、実行中のコールバックを一時停止します。これらのコールバックも、関数 `drawnow`、`figure`、`getframe`、`pause`、または `waitfor` が実行するときに限り、割り込みできます。

コールバック プロパティに `Interruptible` が適用可能かどうかは、コールバック プロパティが定義されているオブジェクトに依存します。

- オブジェクトが `Figure` の場合、`ButtonDownFcn`、`KeyPressFcn`、`KeyReleaseFcn`、`WindowButtonDownFcn`、`WindowButtonMotionFcn`、`WindowButtonUpFcn`、`WindowScrollWheelFcn` に対して定義されたコールバック ルーチンに限り、`Interruptible` プロパティによって影響されます。

上述した大部分のコールバックのように、連続して実行できるコールバックでは、他のオブジェクトまたは GUI からのコールバックが実行している間に、実行できるように `Figure` の `Interruptible` プロパティを '`off`' に設定することが必要になります。特別な理由がない限り、実行されているコールバックを中断しないでください。

- オブジェクトが GUI コンポーネントの場合、`Interruptible` は、コンポーネントに定義されているプロパティが `ButtonDownFcn`、`Callback`、`CellSelectionCallback`、`KeyPressFcn`、`SelectionChangeFcn`、`ClickedCallback`、`OffCallback`、`OnCallback` プロパティであるときに適用されます。

上述したような連続して繰り返すコールバックが中断されないようにするには、コールバックが繰り返し実行されているオブジェクトの `Interruptible` プロパティの値を「`off`」に設定します。

```
set(hObject,'Interruptible','off');
```

ここで、`hObject` は、(たとえば、他の GUIDE GUI を読み込むために)コールバックが繰り返し呼び出されているオブジェクトに対するハンドルです。

Busy Action プロパティの動作仕様

オブジェクトの `BusyAction` プロパティは、`queue` (既定値) または `cancel` になります。実行しているコールバックのオブジェクトの `Interruptible` プロパティが `off` である (すなわち、実行しているコールバックが一時停止できない) 場合に限り、一時停止しているコールバックのオブジェクトの `BusyAction` プロパティが考慮されます。

一時停止できないコールバックが実行していて、イベント (マウスクリックなど) が新しいコールバックをトリガーすると、MATLAB は、新しいコールバック オブジェクトの `BusyAction` プロパティの値を用いて、要求されたコールバックを待ち行列に追加する、あるいはキャンセルするかを決めます。

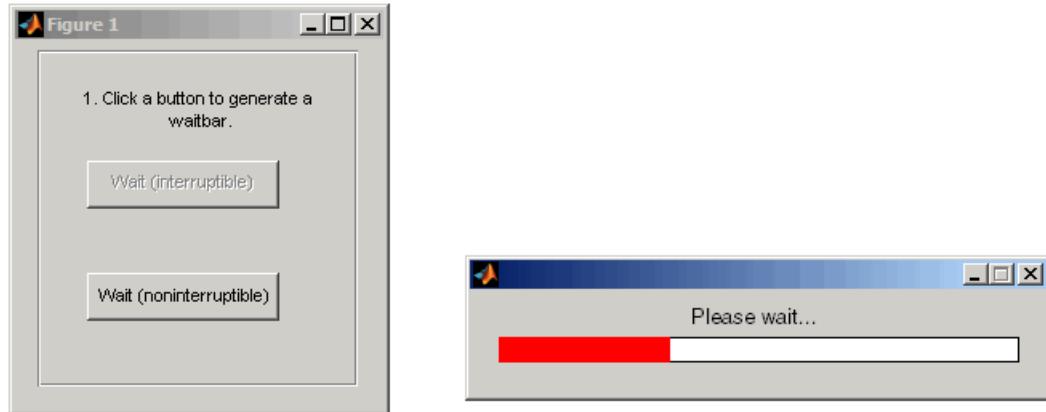
- `BusyAction` の値が `queue` である場合、要求されたコールバックはイベント キューに追加され、実行しているコールバックが実行を停止するときに、その順番になって実行します。
- `BusyAction` の値が `cancel` である場合、イベントは破棄され、要求されたコールバックは実行しません。

一時停止可能なコールバックが実行している場合、実行しているコールバックが終了したり、または `drawnow`、`figure`、`getframe`、`pause`、`waitfor` を呼び出すとき、要求されたコールバックが実行します。要求されたコールバックのオブジェクトの `BusyAction` プロパティは、影響をもちません。

例

この例は、`Interruptible` と `BusyAction` プロパティを用いて、コールバックの一時停止のコントロールを説明します。以下の 2 つの GUI を作成します。

- はじめの GUI には、2 つのプッシュ ボタンがあります。`Interruptible` プロパティが `on` に設定された `Wait (interruptible)` と `Interruptible` プロパティが `off` に設定された `Wait (noninterruptible)` です。いずれかのボタンをクリックすると、ウエイトバーの作成や更新を行うボタンの `Callback` コールバックをトリガードします。



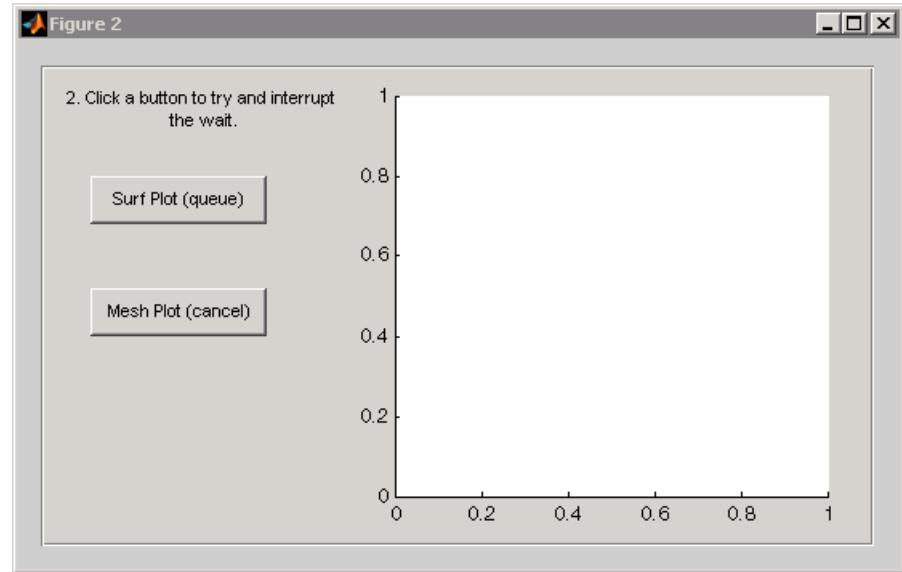
次のコードは、2つの Wait ボタンを作成し、それらのコールバックを指定します。

```

h_interrupt = uicontrol(h_panel1,'Style','pushbutton',...
    'Position',[30,110,120,30],...
    'String','Wait (interruptible)',...
    'Interruptible','on',...
    'Callback',@wait_interruptible);
h_noninterrupt = uicontrol(h_panel1,'Style','pushbutton',...
    'Position',[30,40,120,30],...
    'String','Wait (noninterruptible)',...
    'Interruptible','off',...
    'Callback',@wait_noninterruptible);

```

- 2つめの GUI には、2つのプッシュ ボタンがあります。BusyAction プロパティが queue に設定された Surf Plot (queue) と BusyAction プロパティが cancel に設定された Mesh Plot (cancel) です。いずれかのボタンをクリックすると、ボタンの Callback コールバックをトリガーし、座標軸にプロットを作成します。



次のコードは、2つのプロットボタンを作成し、それらのコールバックを指定します。

```
hsurf_queue = uicontrol(h_panel2,'Style','pushbutton',...
    'Position',[30,200,110,30],...
    'String','Surf Plot (queue)',...
    'TooltipString','BusyAction = queue',...
    'BusyAction','queue',...
    'Callback',@surf_queue);
hmesh_cancel = uicontrol(h_panel2,'Style','pushbutton',...
    'Position',[30,130,110,30],...
    'String','Mesh Plot (cancel)',...
    'BusyAction','cancel',...
    'TooltipString','BusyAction = cancel',...
    'Callback',@mesh_cancel);
```

例の GUI の利用

例の GUI を実行するには、ここをクリックしてください。

メモ 次のリンクは、MATLAB コマンドを実行し、MATLAB ヘルプ ブラウザー内で動作するように設計されています。オンラインあるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

Interruptible と BusyAction プロパティのやりとりを見るために、以下を行います。

- 1 はじめの GUI の Wait ボタンの 1 つをクリックします。いずれのボタンもウエイトバーを作成し、更新します。
- 2 ウエイトバーがアクティブであるときに、2 つめの GUI の Surf Plot または Mesh Plot ボタンをクリックします。Surf Plot ボタンは、peaks データを用いて surf プロットを作成します。Mesh Plot ボタンは、同じデータを用いて mesh プロットを作成します。

以下のトピックスでは、特定の組合わせのボタンをクリックするときに起こることについて説明します。

- ・ “Wait ボタンのクリック” (p.14-7)
- ・ “Plot ボタンのクリック” (p.14-8)

Wait ボタンのクリック.

2 つの Wait ボタンは、Interruptible プロパティ以外は同じです。本質的に同じであるそれらの Callback コールバックは、waitbar を呼び出し、ウエイトバーを作成し更新する、ユーティリティ関数 create_update_waitbar を呼び出します。Wait (Interruptible) ボタンの Callback コールバック wait_interruptible は、waitbar が drawnow を呼び出すたびに、一時停止が可能です。Wait (Noninterruptible) ボタンの Callback コールバック wait_noninterruptible は、(“Interruptible プロパティの動作仕様” (p.14-2) にリストされた特定のコールバックを除き) 一時停止できません。

以下は、Wait (Interruptible) ボタンの Callback コールバック wait_interruptible です。

```
function wait_interruptible(hObject, eventdata)
    % Disable the other push button.
    set(hObject,'Enable','off')
    % Clear the axes in the other GUI.
    cla(hObject,'reset')
```

```
% Create and update the waitbar.
create_update_waitbar
% Enable the other push button
set(h_noninterrupt,'Enable','on')
end
```

このコールバックは、はじめにもう一方のプッシュボタンを無効にし、2つめの GUI の座標軸をクリアします。次に、コールバックは、ユーティリティ関数 `create_update_waitbar` を呼び出し、ウエイトバーを作成し更新します。`create_update_waitbar` が返ると、コールバックはもう一方のボタンを有効にします。

Plot ボタンのクリック. Plot ボタンをクリックするときに起こることは、最初にいずれの Wait ボタンをクリックしたか、および Plot ボタンの BusyAction プロパティに依存します。

- BusyAction プロパティが `queue` である Surf Plot をクリックすると、MATLAB は、Surf Plot コールバック、`surf_queue` を待ち行列に追加します。

はじめに Wait (interruptible) ボタンをクリックすると、ウエイトバーが、`drawnow` を呼び出す、あるいは終了するか破棄されるとき、`surf_queue` が実行し、surf プロットを表示します。

はじめに Wait (noninterruptible) ボタンをクリックすると、ウエイトバーが終了するか破棄されるときに限り、`surf_queue` が実行します。

以下は、`surf_queue` コールバックです。

```
function surf_queue(hObject, eventdata)
    h_plot = surf(h_axes2, peaks_data);
end
```

- Wait (noninterruptible) をクリックした後に、BusyAction プロパティが `cancel` である Mesh Plot をクリックすると、MATLAB は、ボタンクリックのイベントを破棄し、`mesh_cancel` コールバックを待ち行列に追加しません。

Wait (interruptible) をクリックした後、Mesh Plot をクリックすると、Mesh Plot BusyAction プロパティは影響をもちません。MATLAB は、Mesh Plot コールバック `mesh_cancel` を待ち行列に追加します。このコールバックは、ウエイトバーが `drawnow` を呼び出すか終了または破棄されるときに、実行し、mesh プロットを表示します。

以下は、`mesh_plot` コールバックです。

```
function mesh_cancel(hObject, eventdata)
```

```
    h_plot = surf(h_axes2, peaks_data);  
end
```

完成した GUI M ファイルの表示

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、「ここをクリック」して、この例で使用されるコードの完成したリストを MATLAB エディターで表示できます。

メモ 次のリンクは、MATLAB コマンドを実行し、MATLAB ヘルプ ブラウザー内で動作するように設計されています。オンラインあるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

プログラミングで作成する GUI の例

- ・ “はじめに” (p.15-2)
- ・ “座標軸、メニュー、ツールバーを含む GUI” (p.15-3)
- ・ “テーブルのデータを表示してグラフを作成する GUI” (p.15-17)
- ・ “リストのデータを管理する GUI” (p.15-31)
- ・ “カラー パレット” (p.15-49)
- ・ “アイコン エディター” (p.15-60)

はじめに

以下の 5 つの例では、GUI をどのように手動で作成しプログラムできるかを説明します。各例では、コンポーネントとコンポーネントが使う手法を一覧表示して説明します。例を読み込んで試すと、以下についてどのように行うかがわかります。

- ・ GUI で表形式データのグラフを更新し、それらを新規の Figure にコピーする
- ・ ユーザーが選択を行うまで応答しないダイアログを作成する
- ・ GUI が開かれるときに GUI に入力引数を渡す
- ・ GUI からの出力が返るときに出力を取得する
- ・ GUI の予想外の変更を防ぐ
- ・ 複数のプラットフォームで GUI を実行する
- ・ コンポーネント間でコールバックを共有する
- ・ 複数の GUI でデータを共有する
- ・ メニューとコンテキストメニューを作成する
- ・ 外部ユーティリティ関数を利用する
- ・ 適切なサイズ変更動作を実現する
- ・ GUI をモーダルにする
- ・ ツールバーを作成する

例では、1 つの例以外で入れ子関数を使用します。入れ子関数の詳細は、MATLAB 「プログラミングの基礎」ドキュメンテーションの “Nested Functions”について参照してください。

座標軸、メニュー、ツールバーを含む GUI

このセクションの内容…

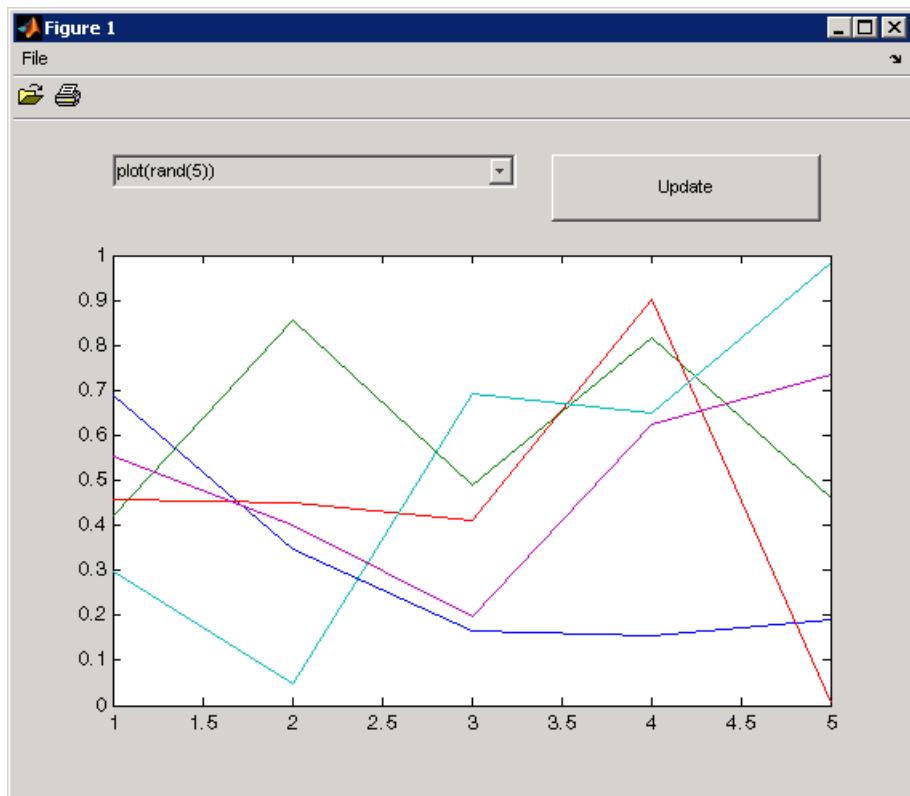
- “座標軸、メニュー、ツールバーの例” (p.15-3)
- “AxesMenu ツールバー の GUI M ファイルの表示と実行” (p.15-5)
- “グラフ作成コマンドとデータの生成” (p.15-6)
- “GUI とそのコンポーネントの作成” (p.15-7)
- “GUI の初期化” (p.15-12)
- “コールバックの定義” (p.15-12)
- “補助関数:プロット タイプのプロット” (p.15-16)

座標軸、メニュー、ツールバーの例

この例は、選択したプロットを座標軸に表示する GUI を作成します。GUI は、以下のコンポーネントを含みます。

- ・ 座標軸
- ・ 5 つのプロットのリストをもつポップアップ メニュー
- ・ 座標軸のコンテンツを更新するためのプッシュ ボタン
- ・ 3 つの項目、[Open]、[Print]、[Close] をもつメニュー バーの [ファイル] メニュー。
- ・ ユーザーがファイルを開いたり、プロットを印刷できる 2 つのボタンをもつツール バー

GUI を実行する場合、次の図に示すように、MATLAB の `rand(5)` コマンドで作成された 5 つの乱数から成るプロットを最初に表示します。



ポップアップメニューで、他のプロットを選択することができます。[Update] ボタンをクリックすると、座標軸上に現在選択されているプロットを表示します。

GUI の [File] メニューは 3 つの項目をもちます。

- ・ [Open] を選択すると、コンピューター上のファイルを開くためのダイアログを表示します。
- ・ [Print] は [印刷] ダイアログを開きます。[印刷] ダイアログで [Yes] をクリックすると、プロットを印刷します。
- ・ [Close] を選択すると、GUI が閉じます。

GUI ツールバーは、2 つのボタンをもちます。

- ・ [Open] ボタン は、[Open] メニュー項目と同じ関数を実行します。[Open] ボタンを押すと、ユーザー コンピューター上のファイルを開くためのダイアログを表示します。
- ・ [Print] ボタン は、[Print] メニュー項目と同じ関数を実行します。これは、[印刷] ダイアログを開きます。[印刷] ダイアログで Yes をクリックすると、プロットを印刷します。

この例は、以下の GUI 作成の手法を説明します。

- ・ GUI が開かれるときに GUI に入力引数を渡す
- ・ GUI からの出力が返るときに出力を取得
- ・ GUI の予想外の変更を防ぐ
- ・ 複数のプラットフォームで GUI を実行
- ・ メニューの作成
- ・ ツールバーの作成
- ・ 適切なサイズ変更の動作の実現

メモ この例は、入れ子関数を使用します。入れ子関数の使用する詳細は、MATLAB 「プログラミングの基礎」ドキュメンテーションの “Nested Functions”について参照してください。

AxesMenu ツールバーの GUI M ファイルの表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の M ファイルにアクセスできます。Web 上で、あるいは PDF でお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下のリンクをクリックします。

- 1 ここをクリックして、M ファイルを現在のフォルダーにコピーしてください。
- 2 エディターに GUI M ファイルを開くには、edit axesToolbar.m と入力するか、または ここをクリックします。
- 3 MATLAB エディターに iconRead M ファイルを開くには、edit iconRead.m と入力するか、または ここをクリックします。

GUI を実行するか、コードを調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 axesToolbar の GUI を実行するには、ここをクリックします。
- 3 MATLAB エディター内に axesToolbar の GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディターで iconRead M ファイルを表示するには、ここをクリックしてください。

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。これらは、現在のフォルダーあるいは他の作業フォルダーに保存してください。

グラフ作成コマンドとデータの生成

例では、2 つの変数 mOutputArgs と mPlotTypes を定義します。

mOutputArgs は、出力値が返されるようにユーザーがリクエストする場合に、出力値を保持するセル配列です。後でこの引数に既定値が割り当てられます。

```
mOutputArgs = {};% Variable for storing output when GUI returns
```

mPlotTypes は、グラフ作成関数とデータを、文字列と無名関数として指定する 5×2 セル配列です。最初の列は、ポップアップ メニューに使用する文字列を含みます。第 2 列は、プロットを作成する関数を、無名関数ハンドルとして含みます。

```
mPlotTypes = [... % Example plot types shown by this GUI
    ' plot(rand(5))' , @(a)plot(a,rand(5));
    ' plot(sin(1:0.01:25))' , @(a)plot(a,sin(1:0.01:25));
    ' bar(1:.5:10)' , @(a)bar(a,1:.5:10);
    ' plot(membrane)' , @(a)plot(a,membrane);
    ' surf(peaks)' , @(a)surf(a,peaks)};
```

データは GUI 関数の最上位レベルで作成されるため、GUI のすべてのコールバックと他の関数で利用できます。

無名関数の利用についての詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションの“Anonymous Functions”を参照してください。

GUI とそのコンポーネントの作成

データと同様、コンポーネントのハンドルが GUI のすべてのコールバックと他の関数で利用できるように、コンポーネントは最上位レベルで作成されます。

- ・ “メイン Figure” (p.15-7)
- ・ “座標軸” (p.15-8)
- ・ “ポップアップ メニュー” (p.15-9)
- ・ “Update プッシュ ボタン” (p.15-9)
- ・ “[File] メニューとそのメニュー項目” (p.15-10)
- ・ “ツールバーとそのツール” (p.15-11)

メイン Figure

次のステートメントは、GUI の Figure を作成します。

```
hMainFigure = figure(... % The main GUI figure
    'MenuBar' , 'none' , ...
    'Toolbar' , 'none' , ...
    'HandleVisibility' , 'callback' , ...
    'Color' , get(0, ...
        'defaultuicontrolbackgroundcolor' ));
```

- ・ 関数 `figure` は GUI の Figure を作成します。

- MenuBar プロパティと Toolbar プロパティを `none` に設定すると、標準のメニューとツールバーが表示されないようにします。
- HandleVisibility プロパティを `callback` に設定すると、Figure は GUI コールバック内からのみアクセスでき、コマンド ラインから追加や削除ができなくなります。
- Color プロパティは、Figure の背景色を定義します。この場合、Color プロパティは、Update プッシュ ボタンなど uicontrol オブジェクトの既定の背景色と同じに設定されます。uicontrol オブジェクトの出荷時の既定の背景色は、システムの既定値でありシステム毎に変わることがあります。このステートメントは、Figure の背景色がコンポーネントの背景色と一致するようにします。

figure プロパティとその既定値についての詳細は、「Figure プロパティ」のリファレンス ページを参照してください。

座標軸

次のステートメントは、座標軸を作成します。

```
hPlotAxes = axes(... % Axes for plotting the selected plot
                  'Parent', hMainFigure, ...
                  'Units', 'normalized', ...
                  'HandleVisibility', 'callback', ...
                  'Position', [0.11 0.13 0.80 0.67]);
```

- 関数 `axes` は座標軸を作成します。座標軸の `Parent` プロパティを `hMainFigure` に設定すると、座標軸をメイン Figure の子にします。
- `Units` プロパティを `normalized` に設定すると、GUI がサイズ変更されるときに座標軸が比例してサイズ変更されます。
- `Position` プロパティは、Figure 内の座標軸の位置とサイズを指定する 4 要素ベクトル [左端からの距離、下端からの距離、幅、高さ] です。`units` は正規化されているため、すべての値は 0 から 1 までの値になります。

メモ `Units` プロパティを指定すると、`Units` プロパティの値に依存する `Position` プロパティやその他のプロパティは、`Units` プロパティの指定に従います。

`Axes` プロパティとその既定値についての詳細は、「Axes プロパティ」のリファレンス ページを参照してください。

ポップアップ メニュー

次のステートメントは、ポップアップ メニューを作成します。

```
hPlotsPopupMenu = uicontrol(... % List of available types of plot
    'Parent', hMainFigure, ...
    'Units', 'normalized',...
    'Position', [0.11 0.85 0.45 0.1],...
    'HandleVisibility', 'callback',...
    'String', mPlotTypes(:,1),...
    'Style', 'popupmenu');
```

- 関数 `uicontrol` は、`Style` プロパティの値に基づき、様々なユーザー インターフェイス コントロールを作成します。ここで、`Style` プロパティは `popupmenu` に設定されます。
- ポップアップ メニューに対し、`String` プロパティはメニュー項目のリストを定義します。これは、セル配列 `mPlotTypes` から導かれた文字列の 5×1 セル配列として定義されます。

`uicontrol` オブジェクトのプロパティとその既定値についての詳細は、「`Uicontrol` プロパティ」のリファレンス ページを参照してください。

Update プッシュ ボタン

次のステートメントは、`uicontrol` オブジェクトとして `Update` プッシュ ボタンを作成します。

```
hUpdateButton = uicontrol(... % Button for updating selected plot
    'Parent', hMainFigure, ...
    'Units', 'normalized',...
    'HandleVisibility', 'callback',...
    'Position', [0.6 0.85 0.3 0.1],...
    'String', 'Update',...
    'Callback', @hUpdateButtonCallback);
```

- 関数 `uicontrol` は、`Style` プロパティの値に基づき、様々なユーザー インターフェイス コントロールを作成します。`Style` プロパティの既定値が `pushbutton` であるので、このステートメントは `Style` プロパティを設定しません。
- プッシュ ボタンに対し、`String` プロパティはボタン上にラベルを定義します。ここでは 文字列 `Update` を定義します。

- Callback プロパティを @hUpdateButtonCallback に設定すると、プッシュ ボタンを提供するコールバック関数の名前を定義します。すなわち、プッシュ ボタンをクリックすると、名前の付いたコールバックの実行を引き起します。このコールバック関数は、後でスクリプトで定義されます。

uicontrol オブジェクトのプロパティとその既定値についての詳細は、「Uicontrol プロパティ」のリファレンス ページを参照してください。

[File] メニューとそのメニュー項目

次のステートメントは、[File] メニューとそれに含まれる 3 つの項目を定義します。

```
hFileMenuItem = uimenu(... % File menu
    'Parent', hMainFigure, ...
    'HandleVisibility', 'callback', ...
    'Label', 'File');

hOpenMenuItem = uimenu(... % Open menu item
    'Parent', hFileMenuItem, ...
    'Label', 'Open', ...
    'HandleVisibility', 'callback', ...
    'Callback', @hOpenMenuItemCallback);

hPrintMenuItem = uimenu(... % Print menu item
    'Parent', hFileMenuItem, ...
    'Label', 'Print', ...
    'HandleVisibility', 'callback', ...
    'Callback', @hPrintMenuItemCallback);

hCloseMenuItem = uimenu(... % Close menu item
    'Parent', hFileMenuItem, ...
    'Label', 'Close', ...
    'Separator', 'on', ...
    'HandleVisibility', 'callback', ...
    'Callback', @hCloseMenuItemCallback);
```

- 関数 uimenu は、メインメニュー、[File] と [File] に含まれる項目を作成します。メインメニューとその項目のそれぞれに対して、希望する親のハンドルを Parent プロパティへ設定して所望のメニュー階層を作成します。ここで、[File] メニューの Parent プロパティを hMainFigure に設定すると、メイン Figure の子になります。このステートメントは Figure にメニュー バーを作成し、Figure 上に [File] メニューを置きます。

各メニュー項目の Parent プロパティを、親のメニューのハンドル hFileMenuItem に設定すると、[File] メニュー上に表示されます。

- ・ メイン メニューとその上の各項目に対して、Label プロパティはメニューに現れる文字列を定義します。
- ・ [Close] メニュー項目に対する Separator プロパティを on に設定すると、この項目の上に区切りの線が表示されます。
- ・ 各メニュー項目の Callback プロパティは、その項目を実行するコールバックを指定します。この例では、[File] メニュー自身を使用可能にするコールバックはありません。これらのコールバックは、後でスクリプトで定義されます。

uicontrol オブジェクトのプロパティとその既定値についての詳細は、「Uicontrol プロパティ」のリファレンス ページを参照してください。

ツールバーとそのツール

次のステートメントは、ツールバーとツールバーが含む 2 つのボタンを定義します。

```

hToolbar = uitoolbar(...    % Toolbar for Open and Print buttons
    ' Parent' ,hMainFigure, ...
    ' HandleVisibility' , ' callback' );
hOpenPushtool = uipushtool(...    % Open toolbar button
    ' Parent' ,hToolbar, ...
    ' TooltipString' , ' Open File' ,...
    ' CData' ,iconRead(fullfile(matlabroot, ...
        ' toolbox\matlab\icons\opendoc.mat')),...
    ' HandleVisibility' , ' callback' , ...
    ' ClickedCallback' , @hOpenMenuItemCallback);
hPrintPushtool = uipushtool(...    % Print toolbar button
    ' Parent' ,hToolbar, ...
    ' TooltipString' , ' Print Figure' ,...
    ' CData' ,iconRead(fullfile(matlabroot, ...
        ' toolbox\matlab\icons\printdoc.mat')),...
    ' HandleVisibility' , ' callback' , ...
    ' ClickedCallback' , @hPrintMenuItemCallback);

```

- ・ 関数 uitoolbar は、メインの Figure 上にツールバーを作成します。
- ・ 関数 uipushtool は、ツールバー上に 2 つのプッシュ ボタンを作成します。
- ・ uipushtool の TooltipString プロパティは、GUI ユーザーがマウス ポインターをボタン上に移動し、そのまま置くときに表示するツール チップを割り当てます。

- CData プロパティは、ボタン上に表示するトルーカラー イメージを指定します。これらの 2 つのボタンに対して、ユーティリティ関数 iconRead はイメージを提供します。
- 各 uipushtool の ClickedCallback プロパティは、GUI ユーザーがボタンをクリックするとき実行するコールバックを指定します。[Open] プッシュボタン と [Print] プッシュボタン は、それらの対応するメニュー項目と同じコールバックを使用することに注意してください。

詳細は、“ツールバーの作成”(p.11-87)を参照してください。

GUI の初期化

次のステートメントは、GUI が最初に表示されたときに現れるプロットを作成します。GUI が実行しているときにユーザーが出力引数を与える場合、ユーザーに返される出力を定義します。

```
% Update the plot with the initial plot type
localUpdatePlot();

% Define default output and return it if it is requested by users
mOutputArgs{1} = hMainFigure;
if nargout>0
    [varargout{1:nargout}] = mOutputArgs{:};
end
```

- 関数 localUpdatePlot は、座標軸上で選択されたプロットタイプをプロットします。ポップアップメニューに関して、uicontrol の Value プロパティは、String プロパティの選択されたメニュー項目のインデックスを指定します。既定値は 1 なので、GUI が最初に表示されたときは 'plot(rand(5))' が選択されます。関数 localUpdatePlot は、スクリプトの後にコールバックと同じレベルで入れ子関数として定義される補助関数です。
- 既定の出力は、メイン Figure のハンドルです。

コールバックの定義

このトピックは、GUI のコンポーネントを提供するコールバックを定義します。コールバックの定義は、コンポーネントの定義や GUI に対して作成されたデータよりも低いレベルにあるので、それらはすべてのデータとコンポーネントハンドルへのアクセスをもちます。

GUI はコールバックで使用可能になる 6 つのコンポーネントをもちますが、4 つのコールバック関数のみも칃ます。これは、[Open] メニュー項目と [Open] ツールバー ボタン が同じコールバックを共有するためです。同様に、[Print] メニュー項目と [Print] ツールバー ボタン は同じコールバックを共有します。

- ・ “[Update] ボタン コールバック” (p.15-13)
- ・ “[Open] メニュー項目コールバック” (p.15-14)
- ・ “[Print] メニュー項目コールバック” (p.15-14)
- ・ “[Close] メニュー項目コールバック” (p.15-15)

メモ これらは、コンポーネントの定義 “GUI とそのコンポーネントの作成” (p.15-7) に指定されたコールバックです。

[Update] ボタン コールバック

関数 `hUpdateButtonCallback` は、[Update] プッシュ ボタンをクリックすると実行されます。[Update] ボタンをクリックすると、コールバック関数の実行がトリガーされます。

```
function hUpdateButtonCallback(hObject, eventdata)
    % Callback function run when the Update button is pressed
    localUpdatePlot();
end
```

関数 `localUpdatePlot` は、座標軸上で選択されたプロット タイプをプロットする補助関数です。これは後でスクリプト“補助関数:プロット タイプのプロット” (p.15-16) に定義されます。

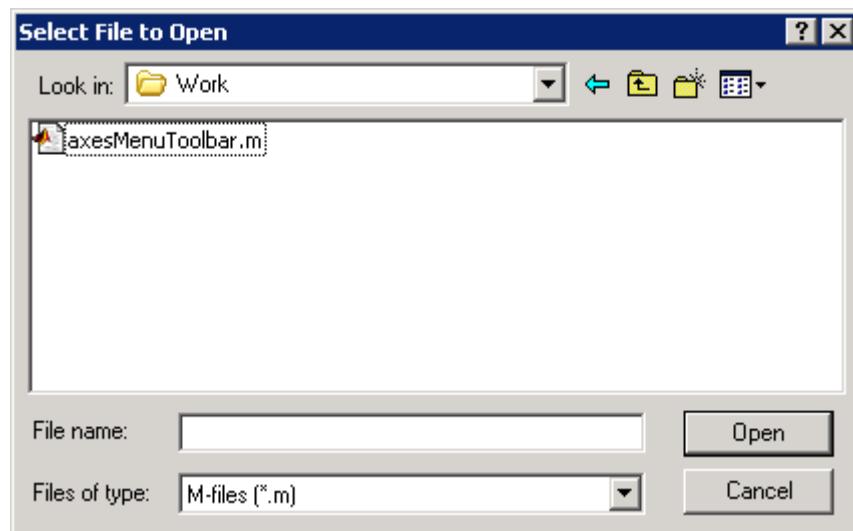
メモ [Update] プッシュ ボタン コンポーネントの `Callback` プロパティ `@hUpdateButtonCallback` は関数ハンドルとして定義されているので、MATLAB は、`hUpdateButtonCallback` に、2 つの引数 `hObject` と `eventdata` を自動的に渡します。`hObject` は、コールバックの実行をトリガーしたコンポーネントのハンドルを含みます。`eventdata` は、将来利用するために予約されています。ユーザー定義のコールバックに関する `function` 定義行では、これら 2 つの入力引数を記述する必要があります。

[Open] メニュー項目コールバック

関数 `hOpenMenuItemCallback` は、[Open] メニュー項目の選択や [Open] ツールバー ボタンのクリックで実行されます。 メニュー項目の選択またはツールバー ボタンのクリックは、コールバック関数の実行をトリガーします。

```
function hOpenMenuItemCallback(hObject, eventdata)
% Callback function run when the Open menu item is selected
    file = uigetfile('*.m');
    if ~isequal(file, 0)
        open(file);
    end
end
```

関数 `hOpenMenuItemCallback` は、最初に関数 `uigetfile` を呼び出し、ファイルを取得するため標準のダイアログ ボックスを開きます。このダイアログ ボックスはすべての M ファイルをリストします。関数 `uigetfile` は、ファイル名を返すと、関数 `open` を呼び出して、M ファイルを開きます。

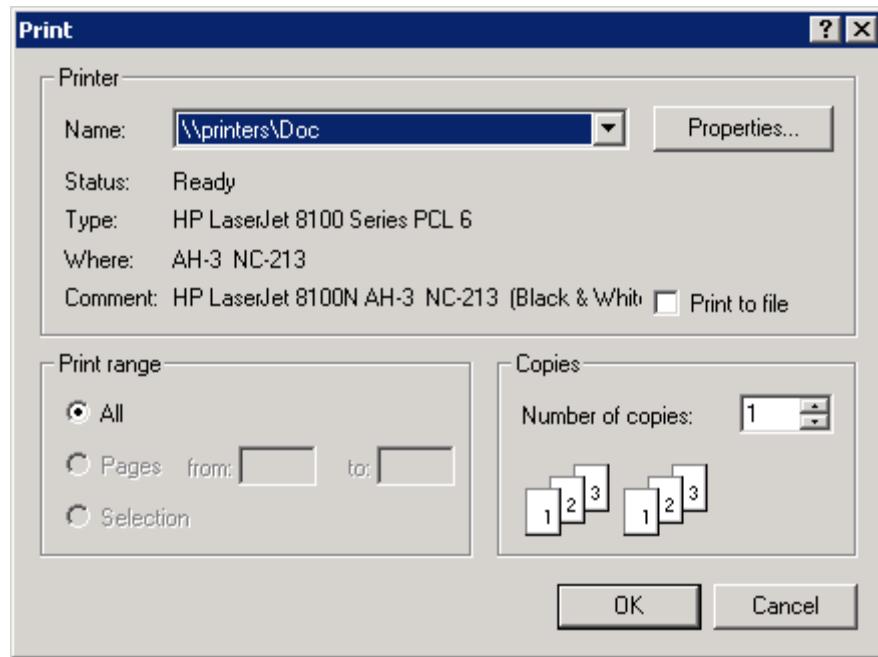


[Print] メニュー項目コールバック

関数 `hPrintMenuItemCallback` は、[Print] メニュー項目の選択や [Print] ツールバー ボタンのクリックで実行されます。 メニュー項目の選択またはツールバー ボタンのクリックは、コールバック関数の実行をトリガーします。

```
function hPrintMenuItemCallback(hObject, eventdata)
% Callback function run when the Print menu item is selected
    printdlg(hMainFigure);
end
```

関数 `hPrintMenuItemCallback` は、関数 `printdlg` を呼び出します。この関数は、現在の図を印刷するための標準的なシステム ダイアログ ボックスを開きます。以下に示す [印刷] ダイアログ ボックスは違うものが表示されることもあります。



[Close] メニュー項目コールバック

関数 `hCloseMenuItemCallback` は、[File] メニューから [Close] を選択すると実行されます。GUI ユーザーが [ファイル] メニューから [閉じる] を選択すると実行します。

```
function hCloseMenuItemCallback(hObject, eventdata)
% Callback function run when the Close menu item is selected
selection = ...
questdlg([' Close ' get(hMainFigure, ' Name' ) ' ?'], ...
[' Close ' get(hMainFigure, ' Name' ) ' ...'], ...
```

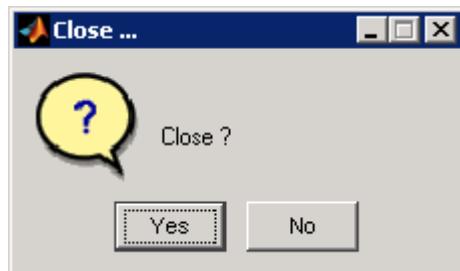
```

        ' Yes' , ' No' , ' Yes' );
if strcmp(selection, ' No' )
    return;
end

delete(hMainFigure);
end

```

関数 `hCloseMenuItemCallback` は、関数 `questdlg` を呼び出して、次の図に示す、質問ダイアログ ボックスを作成し開きます。



ユーザーが [No] ボタンをクリックすると、コールバックが終了します。ユーザーが [はい] ボタンをクリックすると、コールバックは GUI を削除します。

関数 `localUpdatePlot` の説明は、“補助関数:プロット タイプのプロット”(p.15-16)を参照してください。

補助関数:プロット タイプのプロット

例は、コールバック関数と同じレベルで入れ子関数として関数 `localUpdatePlot` を定義します。このため、`localUpdatePlot` は同じデータとコンポーネント ハンドルへのアクセスをもちます。

```

function localUpdatePlot
% Helper function for plotting the selected plot type
mPlotTypes{get(hPlotsPopupMenu, 'Value'), 2}(hPlotAxes);
end

```

関数 `localUpdatePlot` はポップアップメニューの `Value` プロパティを使用して 5×2 セル配列 `mPlotTypes` の 1 番目の列から選択されたメニュー項目を識別し、セル配列の 2 列から対応する無名関数を呼び出し、座標軸にプロットを作成します。

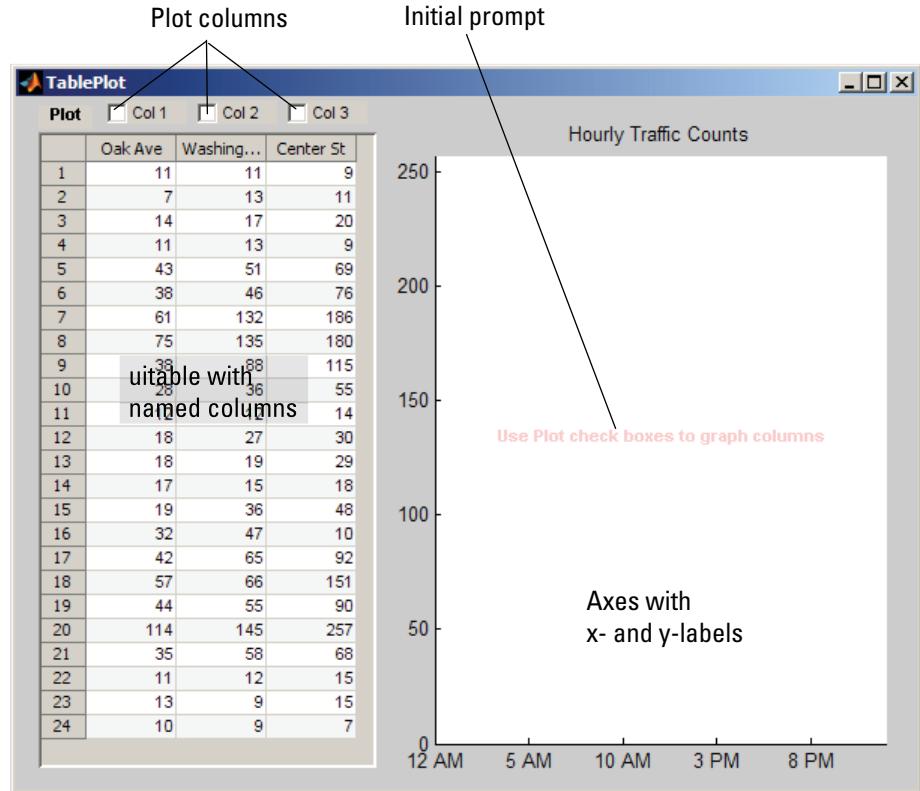
テーブルのデータを表示してグラフを作成する GUI

このセクションの内容…

- “tableplot の例について” (p.15-17)
- “tableplot の GUI M ファイルの表示と実行” (p.15-21)
- “uitable の設定と uitable とのやりとり” (p.15-22)
- “tableplot のサブ関数のまとめ” (p.15-28)
- “tableplot についてさらに検討する” (p.15-28)

tableplot の例について

tableplot の例の GUI では、3 列のテーブル (uitable オブジェクト) でデータを表し、データ列を線グラフとしてプロットすることができます。テーブルでデータ値を選択するプロットでは、選択された観測に対してマーカーを表示します。この手法はデータのブラシ選択と呼ばれ、MATLAB で利用できます。「データ解析」ドキュメンテーションの “Marking Up Graphs with Data Brushing” を参照してください。この GUI でデータのブラシ選択を実行すると、この機能は uitable には適用されないので、MATLAB のブラシ選択は利用しません。GUI のメインのコンポーネントが呼び出されると、最初に開いたときに GUI は次のように見えます。



テーブルには、MATLAB のサンプル データ (count.dat) を表示します。これは、3 地点を通過する自動車の台数をカウントしたデータです。この例では、別のデータセットを読み込み、プロットに手動で適切な列名と別のタイトルを割り当てます。そのため、tableplot.m のメイン関数を変更します。データを変更するその他の方法については述べていません。この機能を追加する通常の方法では、GUI にデータファイルやワークスペース変数を識別する入力引数を与え、列ヘッダーに使うテキスト文字列を与えます。

メモ GUIDE で uitable を含む GUI を作成することもできます。GUIDE を用いて uitable からデータをプロットする例として、“テーブルのデータを対話的に調べる GUI” (p.10-32) を参照してください。

`tableplot` のメイン関数は、以下の UI コンポーネントをもつ GUI の Figure を作成します。

- ・ 3 列のデータをもつ `uitable`
- ・ タイトル付きの座標軸
- ・ データの列をプロットするための 3 つのチェック ボックス
- ・ 2 つのスタティック テキスト文字列

tableplot の例で検討する手法

この例では、`uitable` やそれが保持するデータとやりとりする方法をいくつか示します。

- ・ 列の名前を抽出して、メニュー項目として使用する
- ・ データの特定の列をグラフ表示する
- ・ テーブルのセルを選択すると、グラフをブラシ選択する

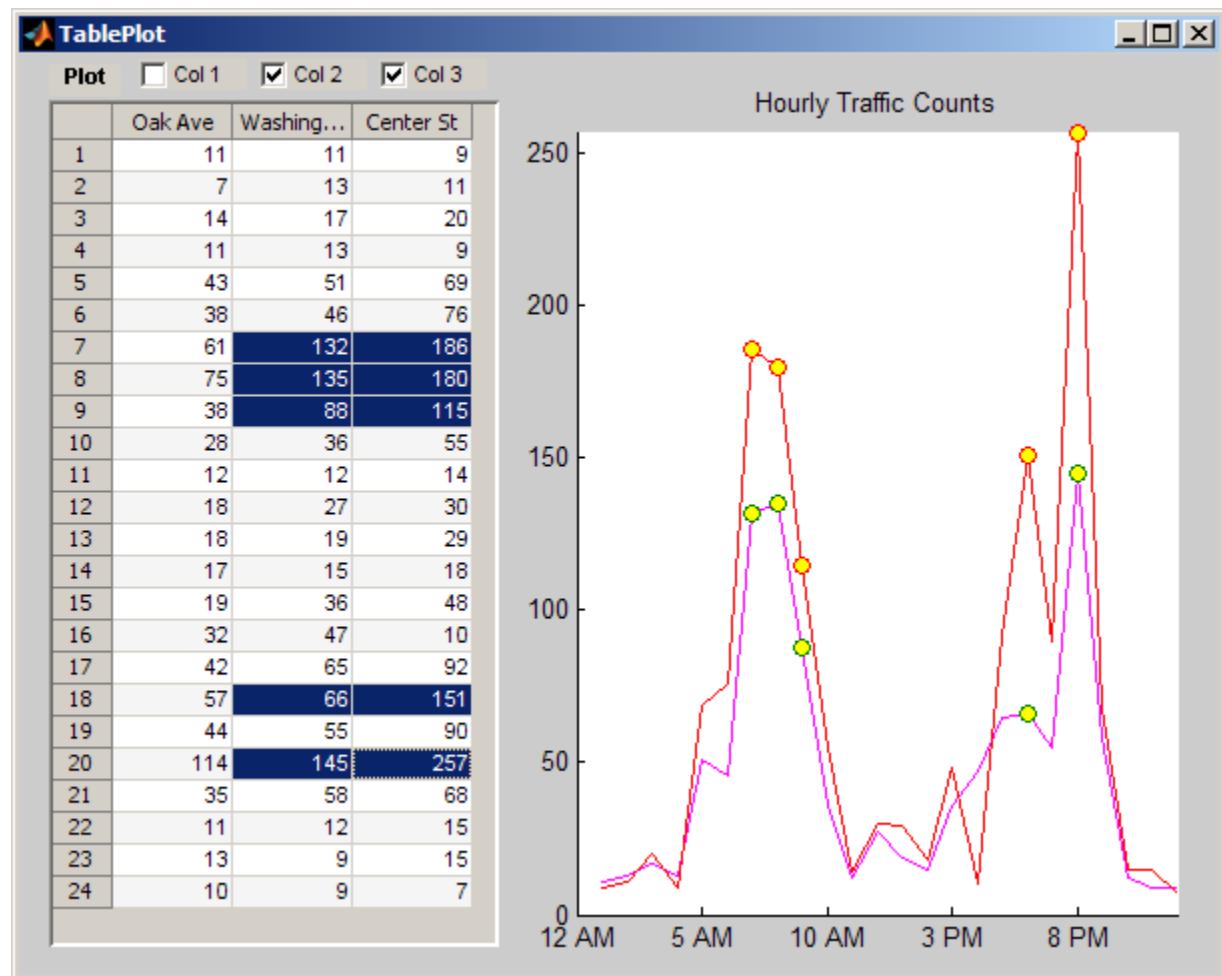
`2-D Axes` は、チェック ボックスの選択に応じて、テーブルの観測を選択した結果をリアルタイムで表示してデータの線グラフを示します。

プロットの作成と除去、ブラシ選択を組合せるために、`uicontrol` のコールバックは引数で渡します。この引数は、互いのハンドルとグラフィック オブジェクトのハンドルを 1 つ以上指定します。次の表では、コールバックとそれらがオブジェクト ハンドルをどのように使うかについて説明します。

UI オブジェクト	ハンドル	コールバック タイプ	コールバック シグネチャ	説明
<code>uitable</code>	<code>htable</code>	<code>Cell Selection Callback</code>	<code>{@select_callback}</code>	選択されていないマーカーの <code>x</code> , <code>y</code> , <code>z</code> の値を空に設定します。 <code> eventdata</code> のマークを表示します。
チェック ボックス	—	コールバック	<code>{@plot_callback, 1}</code>	列 1 のデータの線グラフをプロットします。
チェック ボックス	—	コールバック	<code>{@plot_callback, 2}</code>	列 2 のデータの線グラフをプロットします。

UI オブジェクト	ハンドル	コールバック タイプ	コールバック シグネチャ	説明
チェックボックス	—	コールバック	[@plot_callback, 3]	列 3 のデータの線グラフをプロットします。
マーカー	hmkr	—	—	プロット上の選択されたテーブル データをブラシ選択するためにテーブルの select_callback により使用されます。
スタティックテキスト	hprompt	—	—	Axes にプロンプトが表示され、データをプロットするときに消えます。
スタティックテキスト	—	—	—	チェックボックスの行のラベル

次の図は、2 つの列のプロットの結果を示します。各列の値が大きい方から 5 つの値を選択します。



丸いマーカーは、テーブルのセルが選択されると、動的に表示または非表示になります。マーカーを表示するためにラインをプロットする必要はありません。ラインは個別にプロットされ、3つのチェックボックスを用いて削除されます。

tableplot の GUI M ファイルの表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の M ファイルにアクセスできます。Web 上で、

あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下のリンクをクリックします。

- 1 ここをクリックして、tableplot M ファイルを現在のフォルダーにコピーしてください。
- 2 edit tableplot と入力するか、または ここをクリックすると、エディターに GUI M ファイルが開きます。

GUI を実行するか、コードを調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 tableplot の GUI を実行するには、ここをクリックします。
- 3 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。これらは、現在のフォルダーあるいは他の作業フォルダーに保存してください。

uitable の設定と uitable とのやりとり

この例には 1 つの M ファイル、tableplot.m があります。この M ファイルには、メイン関数と 2 つのサブ関数 (uicontrol のコールバック) が含まれます。メイン関数は、uicontrol と 1 つの Axes をもつ Figure を作成します。Figure のメニュー バーは、そのコンテンツが必要ないため非表示になります。

```
% Create a figure that will have a uitable, axes and checkboxes
figure('Position', [100, 300, 600, 460],...
    'Name', 'TablePlot', ... % Title figure
```

```
' NumberTitle' , ' off' ,... % Do not show figure number
' MenuBar' , ' none' ); % Hide standard menu bar menus
```

メイン関数 `tableplot` は、ワークスペースにデータ行列を読み込んだ直後に `uitable` を設定します。テーブルのサイズは、行列のサイズに一致します。`uitable` の行列は、1-D または 2-D でなければなりません。

```
% Load some tabular data (traffic counts from somewhere)
count = load(' count.dat' );
tablesize = size(count); % This demo data is 24-by-3

% Define parameters for a uitable (col headers are fictional)
colnames = {' Oak Ave' , ' Washington St' , ' Center St' };
% All column contain numeric data (integers, actually)
colfmt = {' numeric' , ' numeric' , ' numeric' };
% Disallow editing values (but this can be changed)
coledit = [false false false];
% Set columns all the same width (must be in pixels)
colwdt = {60 60 60};
% Create a uitable on the left side of the figure
htable = uitable(' Units' , ' normalized' ,...
    ' Position' , [0.025 0.03 0.375 0.92],...
    ' Data' , count,... % 行列をデータとして渡す
    ' ColumnName' , colnames,... % 列名を列ごとに指定
    ' ColumnFormat' , colfmt,... % データ形式を列ごとに指定
    ' ColumnWidth' , colwdt,... % 列幅を列ごとに指定
    ' ColumnEditable' , coledit,... % 列編集可能フラグを列ごとに指定
    ' ToolTipString' ,... % ツールヒント文字列
    ' Select cells to highlight them on the plot' ,... % セル選択機能
    ' CellSelectionCallback' , {@select_callback}));
```

列は、`ColumnName` プロパティで設定する名前をもちます。すべての列は、数値データ (`ColumnFormat` プロパティ) を保持するとして指定され、幅 60 ピクセルに設定されます (`ColumnWidth` プロパティは常にピクセルとして解釈されます)。ツールのヒントの文字列が与えられ、`count` 行列が、`Data` パラメーターとしてテーブルに渡されます。大部分の `Uitable` プロパティは既定のものです。`CellEditCallback` プロパティと関連する `ColumnEditable` プロパティ (table セルを編集不可とします) を含みます。

次に、Figure の右半分に Axes を設定します。これは、ユーザーのアクションに応答してラインやマーカーをプロットします。

```
% Create an axes on the right side; set x and y limits to the
% table value extremes, and format labels for the demo data.
haxes = axes('Units', 'normalized',...
    'Position', [.465 .065 .50 .85],...
    'XLim', [0 tablesize(1)],...
    'YLim', [0 max(max(count))],...
    'XLimMode', 'manual',...
    'YLimMode', 'manual',...
    'XTickLabel',...
    {'12 AM', '5 AM', '10 AM', '3 PM', '8 PM'});
title(haxes, 'Hourly Traffic Counts') % Describe data set
% Prevent axes from clearing when new lines or markers are plotted
hold(haxes, 'all')
```

次に、plot への呼び出しを用いて、マーカーに対する lineseries を作成します。これは、count データセット全体をグラフ化します。このデータセットは、テーブルにコピーされた後もワークスペースにとどまりますが、マーカーは直ちに非表示になります。マーカーは、データテーブルのセルが選択されると表示されます。

```
% Create an invisible marker plot of the data and save handles
% to the lineseries objects; use this to simulate data brushing.
hmksrs = plot(count, 'LineStyle', 'none',...
    'Marker', 'o',...
    'MarkerFaceColor', 'y',...
    'HandleVisibility', 'off',...
    'Visible', 'off');
```

メイン関数は、3列のデータと2つのスタティックテキスト文字列のプロットをコントロールするために3つのチェックボックスを定義します。これに対するコードは、tableplot の M ファイルで参照できます。

Cell Selection コールバック

CellSelectionCallback に対するコードは、以下のようになります。これは、Axes のマーカーを表示したり、非表示にします。

```
function select_callback(hObject, eventdata)
% hObject Handle to uitable1 (see GCBO)
% eventdata Currently selected table indices
```

```
% Callback to erase and replot markers, showing only those
% corresponding to user-selected cells in table.
% Repeatedly called while user drags across cells of the uitable

    % hmkrs are handles to lines having markers only
    set(hmkrs, 'Visible', 'off') % turn them off to begin

    % Get the list of currently selected table cells
    sel = eventdata.Indices;      % Get selection indices (row, col)
                                    % Noncontiguous selections are ok
    selcols = unique(sel(:, 2));  % Get all selected data col IDs
    table = get(hObject, 'Data'); % Get copy of uitable data

    % Get vectors of x,y values for each column in the selection;
    for idx = 1:numel(selcols)
        col = selcols(idx);
        xvals = sel(:, 1);
        xvals(sel(:, 2) ~= col) = [];
        yvals = table(xvals, col)';
        % Create Z-vals = 1 in order to plot markers above lines
        zvals = col*ones(size(xvals));
        % Plot markers for xvals and yvals using a line object
        set(hmkrs(col), 'Visible', 'on', ...
            'XData', xvals, ...
            'YData', yvals, ...
            'ZData', zvals)
    end
end
```

エディターに select_callback のコードを表示するには、ここをクリックします。

選択したセルの行と列は、eventdata.Indices に渡され、sel にコピーされます。たとえば、テーブルの行 3 の 3 列すべてが選択されると、次のようにになります。

```
 eventdata =
 Indices: [3x2 double]

 sel =
 3     1
 3     2
```

3 3

列 2 と列 3 の行 5, 6, 7 が選択されると、次のようにになります。

```
eventdata =
Indices: [6x2 double]

sel =
    5     2
    5     3
    6     2
    6     3
    7     2
    7     3
```

すべてのマーカーを非表示にした後、コールバックは選択された列を識別します。そのとき、これらの列について繰り返し、選択のための行インデックスが見つかります。選択に現れないすべての行のインデックスの *x* 値は、空に設定されます。*x*-値のベクトルは、テーブルから *y*-値をコピーしてダミーの *z*-値を指定するために使用されます。*z*-値を設定すると、ラインの上にマーカーがプロットされます。最終的に、*x*-、*y*-、*z*-値が、マーカーのそれぞれのベクトルの XData、YData、ZData に割り当てられ、マーカーが再度表示されます。マーカーのデータが空でないときに限り、マーカーが表示されます。

ユーザーは、Ctrl+テーブル セルをクリックすることで、個々のマーカーを追加または削除できます。セルがこのようにハイライト表示されると、マーカーが消えるときにハイライト表示も消えます。セルがハイライトされていない場合、ハイライト表示されマーカーが表示されます。

Plot Check Box コールバック

3 つの [Plot] チェック ボックスはすべて、同じコールバック plot_callback を共有します。この GUI には、標準の hObject と eventdata パラメーターの他、1 つの引数があります。

- column — コールバックの対象となるボックス (とデータの列) を識別する整数

これは、以下の目的で、関数ワークスペースにあるハンドルも使用します。

- htable – データをプロットしてラインを削除するために、表のデータと列名を取得します。column 引数は、描く列または削除する列を識別します。
- haxes – ラインを描き、Axes からラインを削除します。
- hprompt – Axes からプロンプト（表示されているのは、最初のラインがプロットされるまでに限られます）を削除します。

column 引数を受け取ると、コールバックは以下の動作をします。

- テーブルからデータを抽出し、plot を呼び出します。列からのデータを YData として指定し、その DisplayName プロパティを列の名前に設定します。
- ラインの DisplayName プロパティに基づき、チェックボックスの選択を解除したときにプロットから適切なラインを削除します。

plot_callback のコードは、以下のようになります。エディターでこのコードを表示するには、ここをクリックします。

```
function plot_callback(hObject, eventdata, column)
    % hObject      Handle to Plot menu
    % eventdata    Not used
    % column       Number of column to plot or clear

    colors = { ' b' , ' m' , ' r' } ; % Use consistent color for lines
    colnames = get(htable, 'ColumnName' );
    colname = colnames{column};
    if get(hObject, 'Value' )
        % Turn off the advisory text; it never comes back
        set(hprompt, 'Visible', 'off')
        % Obtain the data for that column
        ydata = get(htable, 'Data' );
        set(haxes, 'NextPlot', 'Add')
        % Draw the line plot for column
        plot(haxes, ydata(:,column),...
              'DisplayName', colname, ...
              'Color', colors{column});
    else % Adding a line to the plot
        % Find the lineseries object and delete it
        delete(findobj(haxes, 'DisplayName', colname))
    end
end
```

tableplot のサブ関数のまとめ

tableplot の例は、以下の表に一覧表示されたコールバックを含みます。このコールバックは、前の節で説明したものです。エディター ウィンドウに関数を開くには、関数名をクリックしてください。

関数	説明
plot_callback	[Plot] チェック ボックスにより呼び出され、uitable からデータを抽出してプロットしたり、切替えたときにプロットからラインを削除します。
select_callback	マーカーを削除してから再びプロットします。uitable でユーザーが選択したセルがある場合は、それに対応するマーカーのみを表示します。

select_callback は eventdata (セル選択のインデックス) を使用します。他のコールバックはイベント データをもちません。

tableplot についてさらに検討する

tableplot の GUI は、有用性を向上させるためにいくつかの方法で一般化できます。

- ・ テーブルの値を編集できる。

ユーザーによる編集を可能にするには、uitable の ColumnEditable プロパティを true に設定し、必要ならば、その CellEditCallback をコーディングします。データセルを編集すると、ラインプロットと、表示されている場合はセル選択マーカーの更新が必要になることがあります。ただし、そのためには、グラフを再びプロットするために、既存の plot_callback と select_callback のコードに似たコードを与える必要があります。

メモ 関数 refreshdata は、表示しているワークスペース変数が変更されるとグラフを更新します。ただし、tableplot の GUI は独自のデータソースを含むので、その中の refreshdata をこの目的で使用することはできません。

- ・ GUI が開くときに、表に読み込むワークスペース変数を指定するコマンド ライン引数を解析する。

`tableplot` を呼び出すときにワークスペース変数の名前を指定すると、その `opening` 関数は、引数を有効にできます。引数は存在しますか? 引数は数値ですか? 引数は 1 次元または 2 次元ですか? 引数がこれらの判定を満たす場合は、引数に `uitable` の Data プロパティを割り当て、次の手順に進みます。

- `uitable` の列名を指定するために 2 番目のコマンド ライン引数を解析する。
オプションの列名は文字列のセル行列として与えます。このセル配列は、データ行列と同じ幅をもちます。この引数が与えられないと、この操作では `Col_1`、`Col_2`、...`Col_n` のような既定の名前の列が割り当てられます。
- `uitable` の列数が増える場合はプロットする列のためのチェック ボックスを追加し、列が減る場合はチェック ボックスを削除する。
テーブルに列を追加するときには新規のチェック ボックスを追加し、テーブルから列を減らす場合はチェック ボックスを削除することができます。`uitable` と GUI を大きくすることができる場合、GUI が大きくなりスクリーンに合わなくなるまで、チェック ボックスの大きさはそのままにして間隔をあけることができます。`uitable` の幅を一定に保つ場合、メニュー上の項目をチェックしたり、あるいはリスト ボックスから名前を選択するなど、プロットする列を選択する別の仕組みが必要になります。
- GUI が実行後、読み込むワークスペース変数を選択するために `uicontrol` とダイアログを組み込む。
たとえば、`whos` を用いて現在のフォルダーを調べるリスト ボックスを追加し、テーブルに与えるために、数値で 2 以下の次元をもつ変数のみを選択できます。リスト ボックスのこれらの項目の 1 つを選択すると、その変数が読み込まれ、`uitable` のデータを置き換えます。この操作は、新規のデータに既定の列名を与える必要があります。
- 列名をオン ザ フライでユーザーが変更できるダイアログを提供する。
これを行う場合、コールバックは `uitable` の列ヘッダーと、(上述したように、メニューを用いてライン プロットを実現する場合) メニュー項目も変更する必要があります。
- 値をプロットする前に値を正規化するオプションを与えるか、あるいは線形プロットではなく片対数プロットで表示する。
データ行列は、非常に異なるデータ範囲と測定の単位をもつ列をもつことがあります。したがって、任意の行列の列を同時にプロットする 1 つの方法は、y 軸に適切な範囲を指定することです。x 軸は、常に、行インデックスを示します。既定では、Axes の `YLim` プロパティは '`auto`' なので、y の範囲はプロットされるデータの最小から最大までの範囲まで適応させます。範囲を設定するコードを与える場合、範囲を変更することができるだけ少なくなるようにロバストにします。あるいは、たとえば正規化や標準化などの方法で、プロットする前に列のデータ値を変換できます。

ユーザーが片対数プロットを作成できるようにすることもできます。このプロットには、 y の値の範囲を圧縮する効果があります。これは `plot_callback` に影響します。`plot_callback` には、ある `uicontrol` の状態に基づき、`plot` または `semilogy` を呼び出すかどうかを決めるロジックが必要になります。

リストのデータを管理する GUI

このセクションの内容…

- “List Master の例について” (p.15-31)
- “List Master の GUI M ファイルの表示と実行” (p.15-34)
- “List Master の利用” (p.15-35)
- “List Master のプログラミング” (p.15-40)
- “List Master に “Import from File” オプションを追加する” (p.15-48)
- “List Master に “Rename List” オプションを追加する” (p.15-48)

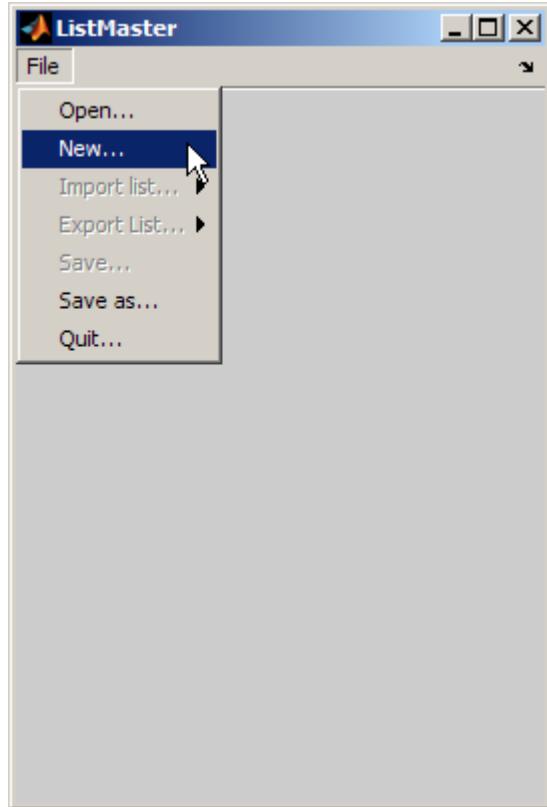
List Master の例について

List Master という GUI を使用すると、ユーザーは、しなければならないことのリストや買い物リスト、携帯電話の番号、音楽やビデオのレコーディング カタログ、項目別に分類したセットなど、複数のリストを管理します。これは、以下の機能をもちます。

- ・ 既存の List Master の GUI から新規の GUI を作成する機能
- ・ 番号付きまたは番号なしの一連の項目を含む、スクローリング リスト ボックス
- ・ テキスト ボックスやプッシュ ボタンによる、リスト項目の編集、追加、削除、並べ替えの機能
- ・ リスト データをインポートやエクスポートしたり、GUI 自体を保存することでリストを保存する機能

[File] メニューでは、GUI を作成、開く、保存する、あるいはリスト データのインポート、エクスポートを取り扱います。次の左側の図には、データが与えられていない新規の List Master の GUI の [File] メニューを表示します。右側は、[File] メニューから作成されたサンプルの GUI で、インポートされたテキスト データを表示しています。

空の List Master の GUI



コントロールとデータをもつ List Master の GUI



コンポーネント

新規の List Master の GUI の Figure は、上記の左側の図に示すように、[File] メニューを含みます。[File] メニューには、List Master の GUI のオープン、作成、データのインポートとエクスポート、GUI の保存、クローズの項目が含まれます。メニュー項目は、必要に応じて有効または無効になります。

新規のリストのタイトルを指定した後、GUI は以下のコンポーネントを追加します。

- GUI のタイトルを表示して、GUI の他のコンポーネントすべてを含むパネル
 - インポートされたリストを表示する、リスト ボックス

- 3つのプッシュ ボタン。各プッシュ ボタンで、それぞれ以下を行います。
 - 選択した項目を現在のリストの上の方に移動する
 - 選択した項目を現在のリストの下の方に移動する
 - 選択した項目をリストから削除する
- リスト項目が数値かそうでないかを指定するチェック ボックス
- 以下を含む “編集パネル”
 - 現在のリスト項目を編集するためのテキスト ボックス
 - 編集に対して行う処理を指定するラジオ ボタン ([Replace] または [Add])

リスト ボックスの選択された状態のリスト項目は、常に 1 つに限られます。その項目のコピーは、エディット ボックスに現れます。ユーザーがこれをエディット ボックスで変更して Return キーを押すと、編集されたテキストは、現在のリストの選択を置き換えるか、あるいは現在のリストの後に挿入され表示されます。[Replace] と [Add] ラジオ ボタンの状態は、編集された項目を GUI が挿入する位置をコントロールします。List Master の GUI を保存すると、そのリスト全体が同時に保存されます。GUI に含まれるリストが変更された場合、GUI は、終了する前に保存するようにメッセージを表示します。

List Master の例で検討する手法

この例は、以下の方法を示します。

- リスト データを受け取る準備のできた GUI の新規のインスタンスを作成可能にする
- GUI の複数のインスタンスを同時に実行可能とする
- ワークスペースから GUI にテキスト データをインポートする
- GUI からのリストをワークスペースまたはテキストファイルにエクスポートする
- 後で使用するために GUI の現在の状態を保存する
- GUI をサイズ変更可能にする
- uicontrol 間で情報を渡すためにアプリケーション データ (appdata) を利用する
- Return キーを (クリックした後ではなく) 押すときに限りエディット テキスト データを送る
- 必要なとき (またはそうでない場合でも) リスト項目に自動的の番号付け、または番号の付け直しを行う

- いくつかの uicontrol に同じコールバックを与え、他の関数からコールバックを呼び出す

List Master の GUI M ファイルの表示と実行

List Master の例には、1 つの M ファイル、2 つの MAT ファイル、1 つのテキストファイルが含まれています。

- listmaster.m – 必要となるサブ関数をすべて含む GUI M ファイル
- listmaster_icons.mat – プッシュ ボタンに対する CData として使用される 3 つのアイコン
- senators110cong.mat – United States の上院議員の電話帳の項目を含むセル配列
- senators110th.txt – senators110cong.mat と同じデータを含むテキストファイル

List Master は新規の GUI を作成するときに MAT ファイル listmaster_icons.mat を探します。[ファイル] > [新規作成] メニューを使用します。senators110cong.mat と senators110th.txt のファイルは、GUI の作成と動作には必要ありません。

[File] > [import list] を使い、新規の List Master の GUI にデータを与えるためにどちらかのファイルを使用できます。

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の M ファイルにアクセスできます。Web 上で、あるいは PDF でお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下のリンクをクリックします。

- 1 ここをクリックして、List Master に対する 4 つのファイルを現在のフォルダーにコピーしてください。
- 2 edit listmaster と入力するか、または ここをクリックすると、エディターに GUI M ファイルが開きます。
- 3 MATLAB エディターにサンプル データ ファイル senators110th.txt を表示するには、edit senators110th.txt と入力するか、ここをクリックしてください。

GUI を実行するか、コードを調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 listmaster GUI を実行するには、ここをクリックします。
- 3 MATLAB エディター内に GUI M ファイルを表示するには、edit listmaster.m と入力するか、ここをクリックしてください。(読み取り専用)
- 4 MATLAB エディターにサンプル データ ファイル senators110th.txt を表示するには、edit senators110th.txt と入力するか、ここをクリックしてください。

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。これらは、現在のフォルダーあるいは他の作業フォルダーに保存してください。

List Master の利用

List Master の GUI は、いつでも [ファイル] > [新規作成] でそれ自身の新規インスタンスを作成し、これらのインスタンスを任意の数、同時に開くことができます。

List Master の起動

List Master の利用をはじめるには、listmaster.m がユーザー パス上にあることを確かめてメイン関数を実行します。

```
1  
    >> listmaster  
  
ans =  
    1
```

関数は、タイトル [ListMaster] と [File] メニューをもつ空白 GUI を開き、その Figure ハンドルを返します。

2 リスト データを取り扱うために GUI をセットアップするには、[File] メニューから [New] を選択します。

GUI はダイアログ ボックスを表します (inputdlg を利用)。

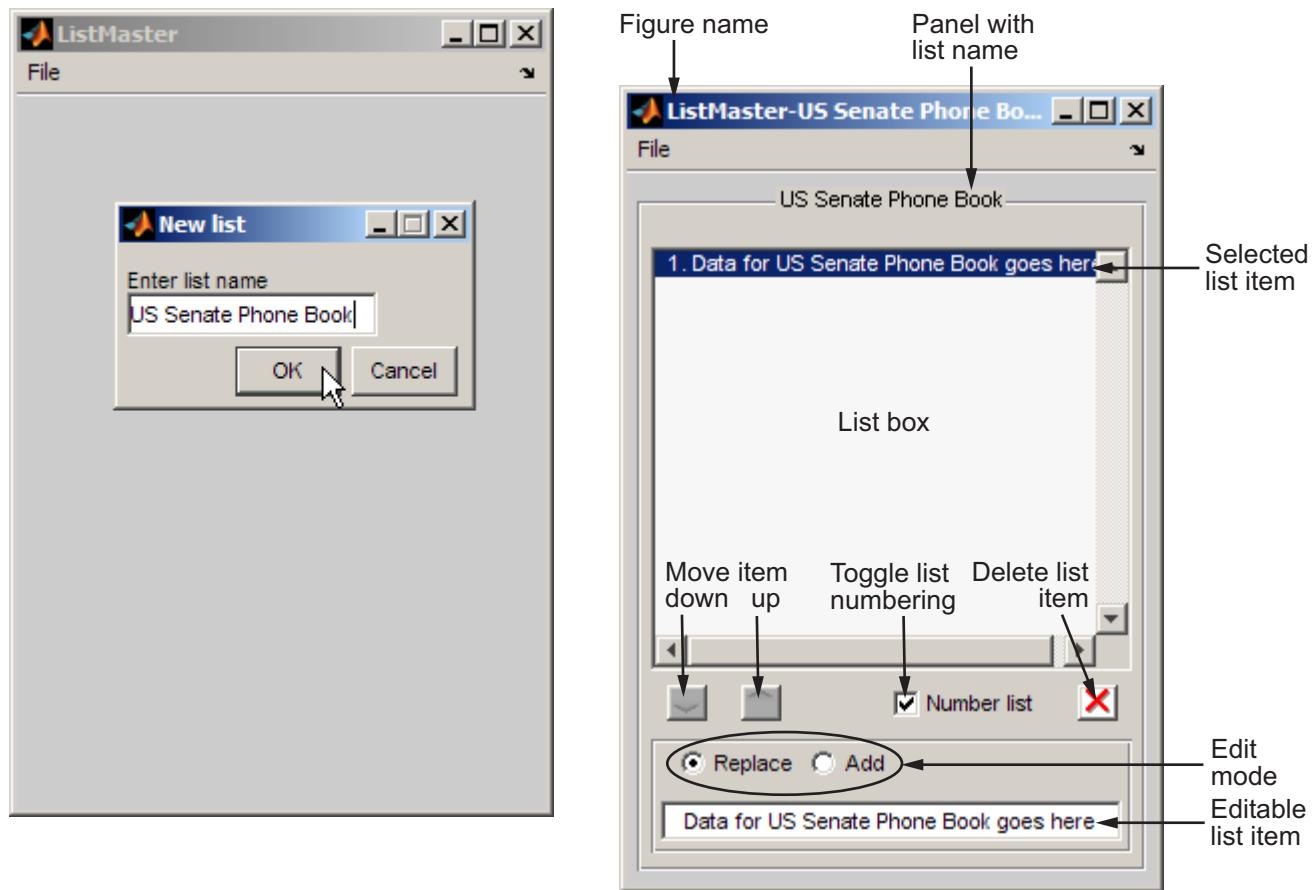
3 作成するリストの名前を入力して [OK] をクリックします。

この例で与えられたサンプル データを使用する場合、次の図の左側に示すように、[US Senate Phone Book] のようなリストを呼び出すことができます。List Master はリストの名前を 32 文字以下の文字数に制限します。

[New] メニュー項目のコールバック、lmnew は、Figure をラベルして uipanel ではじまる GUI のコントロールを作成します。パネルは、その他のコントロールすべてを囲みます。下記の右側に示すように、そのコントロールにラベルを付けた状態をもつ GUI になります。

新規の List Master の GUI を作成する

List Master Controls



すべてのコントロールの位置が normalized Units で指定されるので、GUI はサイズ変更可能で、ボタンアイコンとテキストフォントのサイズは固定しています。

List Master にデータをインポートする

List Master の GUI には任意の時点ですべてのデータをインポートできます。GUI にすでに含まれているデータは、インポートするデータで置き換えられます。

メモ List Master の GUI は、だれかがコンテンツを変更しても GUI の名前を変更することができます。“List Master に “Rename List” オプションを追加する” (p.15-48) では、このような機能を追加する方法を説明します。

MATLAB ワークスペースのセル配列から GUI のテキストにインポートすることができます。セル配列の各要素は、1 つのリスト項目に相当する 1 行のテキストを含む必要があります。たとえば、以下のように grocery 項目のリストを定義できます。

```
groceries = {'1 dozen large eggs';
             '1/2 gallon milk';
             '5 lb whole wheat flour';
             '1 qt. vanilla ice cream'};
```

例の MAT-ファイル senators110cong.mat を読み込み、変数エディターに表示すると、このような構造であることがわかります。この MAT-ファイルを読み込み、変数エディターに開くには、ここをクリックします。

リスト内の語の区切り文字には、スペースのみを使用してください。リストが tab を含む場合、リストボックスはそれらを表示しません。これらは空白としても表示されません。

そのままでは、List Master の例の M ファイルを用いてテキストファイルからデータをインポートすることはできません。そのため [File] メニューの項目がコメントアウトされ、そのコールバック (lfileimport) にはコードが含まれていません。詳細は、“List Master に “Import from File” オプションを追加する” (p.15-48) を参照してください。

List Master で作業するためにデータをインポートする必要はありません。GUI により、[Add] ラジオ ボタンを選択することでリストを作成することができます。エディット テキストボックスに 1 つずつ項目を入力して、Return キーを押すと、それぞれがリストに追加されます。ユーザーは、作成する任意のリストをエクスポートすることができます。

List Master からデータをエクスポートする

[File] > [Export list] > [to workspace] を使うと、MATLAB ワークスペースにセル配列の現在のリストを保存することができます。lwsexport コールバックはこの操作を実行します。ユーザーが変数名を指定した後、変数を作成するために関数 assignin を呼び出します。1 つのリスト項目のみをエクスポートする場合には、以下のようにシステムのクリップボードを使用できます。

1 コピーするリスト ボックスの項目をクリックするか、エディット ボックスで項目のテキストを選択します。

2 項目をコピーするには、Ctrl+C を押します。

3 項目を貼り付けようとしているドキュメンテーションを開きます。

4 項目を貼り付ける位置にカーソルを置き、Ctrl+V を押します。

この項目は、外部ドキュメンテーションに表示されます。

他のドキュメンテーションから文字列をコピーし、テキスト エディット ボックスに貼り付けることもできます。ただし、Return と Tab 文字は、なくなることがあります。

リスト ボックスのコンテンツは、String プロパティを設定してプログラミングでのみ変更可能なので、システムのクリップボードからリスト ボックスに貼り付けることはできません。つまり、新規の項目をリストに貼り付けるには、エディット テキスト ボックスから 1 つずつ追加する必要があります。また、リスト全体はコピーしてから他のドキュメンテーションに貼り付けることもできません。

[File] > [Export list] > [to file] を利用して、テキスト ファイルにリストのコンテンツ全体を保存することができます。メニュー項目のコールバック (lmyfileexport) は、フォルダーをナビゲートしてファイル名を指定するために標準のファイル ダイアログを開きます。次に、ファイルの作成、書き込み、閉じるために fopen、fprintf、fclose を呼び出します。

GUI を保存する

GUI を保存するためにリストをエクスポートする必要はありません。[Save] と [Save as] メニュー オプションは、GUI 全体を保存することでリストを保存します。それらは、MATLAB 関数 saveas を呼び出し、Figure とそのコンテンツすべてを FIG-ファイルとしてディスクに書き込みます。保存された GUI は、現在のフォルダー ブラウザーでダブルクリックするか、あるいはコマンド ラインから hgload('figfilename.fig') を呼び出すことで、再び開くことができます。しかしながら、GUI が機能するには、listmaster.m は、現在のフォルダー あるいは MATLAB パス上になければなりません。

List Master のプログラミング

List Master の GUI M ファイルは 22 個の関数を含みます。これらの関数は、5 つのグループにまとめられます。

- ・ “List Master のメイン プログラム” (p.15-40)
- ・ “List Master の Setup 関数” (p.15-42)
- ・ “List Master メニューのコールバック” (p.15-43)
- ・ “Master List のコールバックのリスト” (p.15-45)
- ・ “List Master のユーティリティ関数” (p.15-46)

List Master のメイン プログラム

メイン関数 `listmaster` は、スクリーンの中央付近に Figure を縦長書式で開きます。次に、サブ関数 `lm_make_file_menu` を呼び出し、[ファイル] メニューを作成します。以下の表に、メニュー項目を記載し、それらのコールバックをリストします。MATLAB エディターに関数を表示するには、Callback 列で関数名をクリックします。MATLAB エディターにコールバックを表示するには、コールバックをクリックします。List Master の関数やコールバックへの以前のリンクや後に続くリンクと同様に、ユーザーがすでにこの M ファイルを作業フォルダーにコピーしても、これらのリンクは Creating Graphical Interfaces の examples フォルダーに M ファイルを表示します。

メニュー項目	用途	コールバック
[Open...]	既存の List Master の Figure を開きます	<code>lmopen</code>
[New...]	既存の Figure がすでにリストを含む場合、初期の GUI または新規の Figure にコントロールを追加することで、List Master を作成します	<code>lmnew</code>
[Import list...]	ワークスペースのセル配列からリストデータを読み込みます	<code>lmwsimport</code>
[Export list...]	現在のリストを含む、ワークスペースのセル配列またはテキストファイルを作成します	<code>lmwsexport</code> 、 <code>lmfileexport</code>
[Save]	現在の List Master とそのコンテンツを FIG-ファイルとして保存します	<code>lmsave</code>

メニュー項目	用途	コールバック
[Save as...]	現在の List Master を FIG-ファイルに別名で保存します	lmsaveas
[Quit]	List Master を終了します。オプションで最初のものを保存します。	lmquit

[File] メニューをもつ空白 GUI を作成後、関数 `listmaster` を終了します。

メイン関数は、以下のように Figure をセットアップします。

```
fh = figure('MenuBar','none',...
    'NumberTitle','off',...
    'Name','ListMaster',...
    'Tag','lmfigtag',...
    'CloseRequestFcn',@lmquit,...
    'Units','pixels',...
    'Position',pos);
```

MenuBar をオフにすると、既定の Figure ウィンドウのメニューを除きます。これは、プログラムが後でそれ自身の [File] メニューで置き換えます。NumberTitle は、そのタイトルバーの Figure の通し番号を消去するためにオフにされます。これは、Name プロパティにより、タイトル ListMaster が与えられます。

モニター上の GUI の初期の Position は、以下のように計算されます。

```
su = get(0,'Units');
set(0,'Units','pixels')
scnsize = get(0,'ScreenSize');
scnsize(3) = min(scnsize(3),1280); % Limit superwide screens
figx = 264; figy = 356;           % Default (resizable) GUI size
pos = [scnsize(3)/2 - figx/2 scnsize(4)/2 - figy/2 figx figy];
...
set(0,'Units',su)               % Restore default root screen units
```

[Open] メニュー オプションは、`listmaster.m` で作成された Figure のみを開きます。List Master の各 Figure には、`lmfigtag` に設定された Tag があります。プログラムが FIG-ファイルを開くとき、これはこのプロパティの値を用いて Figure が List Master の GUI であるかを判定します。Tag が他の値をもつと、プログラムは Figure を閉じ、エラーの警告を表示します。

[Quit] メニュー オプションは、Figure が保存される必要があるかどうかをチェックした後 GUI を閉じます。コンテンツが変更された場合、そのコールバック (`lquit`) が `lmsaveas` コールバックを呼び出し、保存可能になります。Figure の `CloseRequestFcn` は、Figure のクローズ ボックスをユーザーがクリックするときに `lquit` コールバックも使用します。

List Master の Setup 関数

GUI は初めにメニュー以外のコントロールをもちませんが、ユーザーは [File] > [Save as] を使い、空白 GUI を FIG-ファイルとして保存できます。保存された FIG-ファイルを開くと、`listmaster.m` が現在ユーザー パス上にあるものとして、関数 `listmaster` を実行する場合と同じ結果が得られます。

ただし、通常はリストを作成します。これは、[File] > [New] をクリックすることで行います。この場合、GUI に `uicontrol` を与えるセットアップ関数を実行します。「`lmnew`」コールバックは、次の列でセットアップ関数を呼び出してこれらのタスクを管理します。下記で 3 つのセットアップ関数を一覧表示して説明します。MATLAB エディターにコールバックを表示するには、コールバックをクリックします。この関数は、Creating Graphical Interfaces の examples フォルダーから開きます。

セットアップ関数	用途
<code>lm_get_list_name</code>	新規のリストの名前を取得するために <code>inputdlg</code> を呼び出します。32 文字のサイズ制限をします。
<code>lm_make_ctrl_btns</code>	リストのナビゲーションのために 3 つのプッシュ ボタンと、行番号付けをコントロールするためにチェック ボックスを作成します。 <code>listmaster_icons.mat</code> を読み込みます。CData としてプッシュ ボタンにアイコンを適用し、コントロールのコールバックを設定します。
<code>lm_make_edit_panel</code>	編集モードをコントロールする 2 つのラジオ ボタンをもつボタン グループを作成します。その下に 1 行のエディット テキスト ボックスを配置します。エディット テキスト コントロールのためのコールバックを設定します。

すると、`lmnew` は `enable_updown` を呼び出します。これはリスト ボックスのコールバックです (`lmtablisttag1` タグ付き)。これは、リストを修正する他のすべての関数からも呼び出されます。`enable_updown` サブ関数は、リスト ボックスで項目を上下に移動させる、プッシュ ボタンの組の `Enable` プロパティを設定します。これは、選択された項目が、リストの最も上にあるときには [上に移動] ボタンを無効にし、最も下にあると

きには [下に移動] ボタンを無効にします。この場合、これは現在のリストの選択をエディット テキスト ボックスにコピーします。エディット テキストの内容は置き換えられます。最終的に、GUI のデータまたは状態が変更されたことを示すために、これは Figure のアプリケーション データで “dirty” フラグを設定します。詳細は、“List Master のユーティリティ関数” (p.15-46) を参照してください。

最終的に、データを受け取るように GUI をセットアップしたので、`lmnew` は [File] > [Import list] メニュー オプションとそのサブ項目を有効にします。

List Master メニューのコールバック

List Master には、7 つのメニュー項目と 3 つのサブメニュー項目があります。4 つめのサブメニュー項目 [Import list from file] は、実現するために読者の練習用に提供されたスタブです。メニュー項目とそれらのコールバックは、“List Master のメイン プログラム” (p.15-40) の節の表に一覧表示されています。

ユーザーの入力を取得するために、メニュー コールバックは組み込みの MATLAB GUI 関数を呼び出します。以下が使用されています。

- `errordlg` (オープン、リストをワークスペースにエクスポート、リストをファイルにエクスポート)
- `inputdlg` (新規、リストをワークスペースにエクスポート)
- `listdlg` (インポート リスト)
- `questdlg` (リストをワークスペースにエクスポート、終了)
- `uigetfile` (オープン)
- `uiputfile` (リストをファイルにエクスポート、別名で保存)

これらはすべて、モーダルのダイアログです。

[New] メニュー項目には、GUI が空白であるか、あるいはすでにリスト ボックスや関連するコントロールを含むかどうかにより、2 つのモードの操作があります。`lmnew` コールバックは、Figure の Name プロパティを解析することで、いずれのケースであるかを判別します。

- GUI が空白の場合、名前は “ListMaster” になります。
- GUI がすでにリストを含む場合、名前は “Listmaster-” の後にリスト名が続きます。

空白 GUI から呼び出されると、関数は名前をリクエストしてから、Figure にすべてのコントロールを与える。リストを含む GUI から呼び出されると、lmnew はメインの関数 listmaster を呼び出し、新規の GUI を作成し、Figure にコントロールを与えるときに (Figure 自体ではなく) Figure のハンドルを使用します。

Master List のコールバックのリスト

下記にメニュー項目に関連しない 6 つのコールバックを一覧表示して説明します。MATLAB エディターにコールバックを表示するには、コールバックをクリックします。この関数は、Creating Graphical Interfaces の examples フォルダーから開きます。

コールバック関数	用途
<code>move_list_item</code>	[Move Up] と [Move Down] プッシュ ボタンで呼び出され、項目をリストで上下に移動させます。
<code>enable_updown</code>	[Move Up] と [Move Down] ボタンを有効または無効にするために様々なサブ関数から呼び出され、エディット テキスト ボックスとリスト ボックスの同期を保ちます。
<code>delete_list_item</code>	現在選択されている項目をリストから削除するために [削除] ボタンから呼び出されます。項目を元の状態に戻す場合はエディット テキスト ボックスに保持します。
<code>enter_edit</code>	ユーザーが文字を入力するとき、エディット テキスト ボックスで呼び出される KeypressFcn。ユーザーが Return をタイプするとき、アプリケーション データに Edit フラグを設定します。
<code>commit_edit</code>	ユーザーが Return をタイプするか、どこかをクリックするときに、エディット テキスト ボックスで呼び出される Callback。これは、enter_edit により設定されたアプリケーション データの Edit フラグをチェックし、Return が最後に押されたキーである場合に限り、リストにエディット テキストを送ります。これにより、不注意で編集することがなくなります。
<code>toggle_list_numbers</code>	<code>lnumlistbtn</code> チェック ボックスの Callback。チェック ボックスの値に応じて、項目をリストするか、削除するために行番号を前置します。

コンポーネントのハンドルの識別. これらと他の List Master のサブ関数に共通する特性は、コンポーネントのハンドルを取得する方法です。コンポーネントのハンドルと他のデータを共有するために、多くの GUI が使用する、関数 `guidata` を使うのではなく、これらのサブ関数は、`Tag` を調べることで動的に必要なハンドルを取得します。これはハードコードされ、変化しません。ハンドルを見つけるには、以下のコードを使用します。

```
% Get the figure handle and from that, the listbox handle
fh = ancestor(hObject,'figure');
lh = findobj(fh,'Tag','lmtablisttag1');
```

ここで、`hObject` は、現在実行しているコールバックを呼び出したオブジェクトです。さらに、「`lmtablisttag1`」は、リストボックスのハードコードされた Tag プロパティです。通常、`ancestor` を用いて Figure のハンドルを調べると、現在の List Master が識別されます。同様に、Figure のハンドルを `findobj` に指定すると、同時に開く List Master のインスタンス数に関わらず、1 つのリストボックス ハンドルが返ります。

メモ オブジェクトが元々もっていたのと同じハンドルをもつオブジェクトを利用することはできないので、保存された Figure を開くときにハンドルを見つけるための上記のようなメソッドが必要になります。FIG-ファイルから GUI を読み込む場合、MATLAB は、そのコンポーネントのオブジェクトすべてに対して新規のハンドルを生成します。その結果、保存された後に再び開くことがある Figure のオブジェクトハンドルに対するリファレンスを格納すべきではありません。その代わり、オブジェクトのハンドルを調べるためにオブジェクトのタグを使用してください。これらは、明示的に設定されない限り変化しません。

List Master のユーティリティ関数

コールバックには、下記に一覧表示して説明する 4 つの小さなユーティリティ関数を使用するものがあります。MATLAB エディターにコールバックを表示するには、コールバックをクリックします。この関数は、Creating Graphical Interfaces の examples フォルダーから開きます。

ユーティリティ関数	用途
<code>number_list</code>	リストが更新され、行番号付け機能がオンである場合、行番号を生成するために呼び出されます。
<code>guidirty</code>	保存されたバージョンと異なることを示すために、Figure のアプリケーション データの論理値の <code>dirty</code> フラグを <code>true</code> または <code>false</code> に設定します。

ユーティリティ関数	用途
isguidirty	Figure の dirty フラグの論理的な状態を返します。
make_list_output_name	Figure の Name プロパティから得られるリスト名を有効な MATLAB 識別子に変換します。これは、GUI を保存したり、あるいはそのデータをエクスポートする場合の既定値として機能します。

各リスト項目の前に 5 つのスペースを追加してから、これらの空白文字 3、2、1 に対して数値を置き換えることで、リストの番号付けが機能します。(数字を表示するためには必要なように) さらに、文字 4 にピリオドを配置して、番号はテキストエディットボックスに表示される現在の项目的コピーから取り除かれ、([Number list] チェックボックスが選択されると) エディットが送られるときに再び先頭に追加されます。これは、リストの大きさを、999 項目までに番号付け可能な範囲に制限します。GUI にそれよりも長いリストを番号付けようとする場合、番号フィールドに文字を追加するために number_list を変更できます。

メモ リストの項目にすでに番号が付けられている場合には、リストのデータをインポートする前に、番号付けの機能をオフにする必要があります。そのようなケースでは、項目の番号がリストに表示されますが、リスト項目をリスト内で上下に移動させても番号を付け直すことはしません。

関数 guidirty は、それが変更されたことを示すために、以下のように、setappdata を用いて Figure のアプリケーションデータを設定します。

```
function guidirty(fh,yes_no)
% Sets the "Dirty" flag of the figure to true or false

setappdata(fh,'Dirty',yes_no);
% Also disable or enable the File->Save item according to yes_no
saveitem = findobj(fh,'Label','Save');
if yes_no
    set(saveitem,'Enable','on')
else
    set(saveitem,'Enable','off')
end
```

関数 isguidirty は、GUI を閉じようとしたときに Figure を保存する必要があるかどうかを決めるために、getappdata を用いてアプリケーションデータをクエリします。

メモ アプリケーション データを使用して、uicontrol 間や作成する GUI の他のオブジェクトの間で情報をやりとりします。Handle Graphics オブジェクトにアプリケーションデータを割り当てることができます。データは任意のタイプになることができ、他のオブジェクトのタイプとは区別されます。アプリケーション データは、set または get が機能するオブジェクトプロパティではありません。アプリケーション データを格納するには関数 setappdata を、取得するには関数 getappdata を使用する必要があります。詳細は、“アプリケーション データ”(p.13-6) の節を参照してください。

List Master に “Import from File” オプションを追加する

List Master の機能を完成させるときには、[File] > [Import list] > [from file] メニュー項目をアクティブにしてください。行 106-108 からコメントを削除し、コールバックにユーザー独自のコードを追加することで、この機能を追加できます。[File] > [Import list] > [from file] メニュー項目を有効にします。出発点として使用できる関連のコードとして、[File] > [Export list] > [to file] により「Imfileexport」を参照してください。

List Master に “Rename List” オプションを追加する

リストにデータをインポートするとき、リストの内容全体をインポートされたデータで置き換えます。新しいリストの内容が非常に異なる場合、リストに新規の名前を付けることもあります。このリスト名は、リストボックスの上に表示されます。[Rename list] のような、メニュー項目またはコンテキストメニュー項目を追加することを考えます。この項目のコールバックは、次のようにになります。

- ・ ユーザーから名前を取得するために `lm_get_list_name` を呼び出します。これは、おそらく、呼び出し側にプロンプトの文字列を指定せるように、それを変更後に行われます。
- ・ ユーザーがダイアログをキャンセルした場合、何も行わない。
- ・ Tag として '`lmtitlepanel|tag`' をもつ uipanel のハンドルを取得する (“コンポーネントのハンドルの識別”(p.15-45) を参照)。
- ・ uipanel の Title プロパティを、ユーザーが直前に指定した文字列に設定します。

リストの名前を変更後、[別名で保存] を用いて新規の FIG ファイルに GUI を保存できます。GUI がすでに保存されていた場合、GUI を新規のファイルに保存すると、元の名前とコンテンツをもつバージョンの GUI を保存します。

カラー パレット

このセクションの内容…

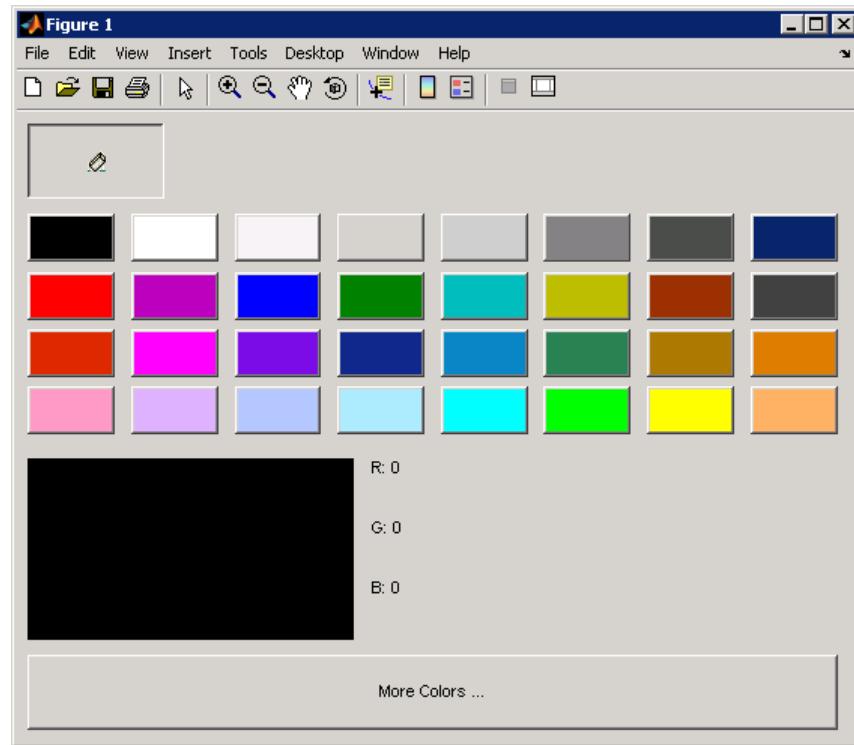
- “カラー パレットの例について” (p.15-49)
- “カラー パレットの例で用いる手法” (p.15-53)
- “カラー パレット GUI M ファイルの表示と実行” (p.15-53)
- “カラー パレットのためのサブ関数のまとめ” (p.15-54)
- “M ファイルの構造” (p.15-55)
- “GUI プログラムの手法” (p.15-56)

カラー パレットの例について

この例は、カラー パレットからカラーを選択したり、あるいは標準的なカラー選択ダイアログ ボックスを表示する GUI、colorPalette を作成します。他の例 “アイコン エディター” (p.15-60) は、アイコンの設計に使用できる GUI で、パネルの子として colorPalette を組み込みます。

関数 `colorPalette` はカラー パレットを使用して GUI の Figure またはパネルを置きます。この例を含む M ファイルへのリンクは、“カラー パレット GUI M ファイルの表示と実行”(p.15-53)を参照してください。

下図は、Figure の子としてパレットを示します。



コンポーネント

`colorPalette` は、以下のコンポーネントを含みます。

- ・ トグル ボタンとして定義されたカラー セルの配列
- ・ アイコンをもつ消しゴムのトグル ボタン
- ・ カラー セルの配列と消しゴムのボタンを含むボタン グループ。ボタン グループは、これらのトグル ボタンを排他的に取り扱います。

- ・ [More Colors] プッシュ ボタン
- ・ テキストコンポーネントとして定義される、カラー セルの下の選択したカラー のプレビュー
- ・ 赤、青、緑のカラー値を指定するテキストコンポーネント

カラー パレットの利用

カラー パレットの利用のための基本的な手順があります。

- 1 カラー セルトグル ボタンをクリックする
 - ・ プレビュー エリアに選択したカラーを表示する
 - ・ 新しく選択されたカラーに対する red, green, blue の値は、プレビュー エリアの右の R, G, B フィールドに表示されます。
 - ・ colorPalette は、現在選択されたカラーを取得するために、ホスト GUI が使用できる関数ハンドルを返します。
- 2 消しゴムのボタンをクリックすると、colorPalette はホスト GUI がデータ点からカラーを除くために使用できる値 NaN を返します。



- 3 [More Colors] ボタンをクリックすると、カラー設定のための標準のダイアログ ボックスを表示します。



colorPalette 関数の呼び出し

関数 `colorPalette` を次のようなステートメントで呼び出すことができます。

```
mGetColorFcn = colorPalette('Parent', hPaletteContainer)
```

関数 `colorPalette` は、入力引数としてプロパティ値の組を受け取ります。カスタム プロパティ `Parent` のみサポートします。このプロパティは、親の Figure のハンドルまたは カラー パレットを含むパネルを指定します。`colorPalette` への呼び出しが親を指定しない場合、現在の Figure `gcf` を使用します。認識されないプロパティ名または無効な値は無視されます。

`colorPalette` は現在選択されているカラーを取得するために、ホスト GUI が呼び出せる関数ハンドルを返します。ホスト GUI はカラー パレットが閉じる前であれば、いつでも出力された関数ハンドルを使用できます。実行の詳細は、“2つの GUI 間でデータを共有する”(p.15-58) を参照してください。“アイコン エディター”(p.15-60) は、`colorPalette` を使用するホスト GUI の例です。

カラー パレットの例で用いる手法

この例は、以下の手法を説明します。

- ・ GUI から出力が返るときに出力を取得する
- ・ データの有効性を確かめたユーザー定義の入力のプロパティ/値の組のサポート
- ・ 2 つの GUI 間でのデータ共有

これらの手法と他のプログラム手法の例は、“アイコン エディター”(p.15-60)を参照してください。

メモ この例は、入れ子関数を使用します。入れ子関数の詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”について参照してください。

カラー パレット GUI M ファイルの表示と実行

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の M ファイルにアクセスできます。Web 上で、あるいは PDF でお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下のリンクをクリックします。

- 1 ここをクリックして、GUI M ファイルを現在のフォルダーにコピーしてください。
- 2 `edit colorPalette.m` と入力するか、または ここをクリックすると、エディターに GUI M ファイルが開きます。

GUI を実行するか、コードを調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)

2 colorPalette GUI を実行するには、ここをクリックします。

3 MATLAB エディター内に GUI M ファイルを表示するには、ここをクリックします。(読み取り専用)

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。これらは、現在のフォルダーあるいは他の作業フォルダーに保存してください。

カラー パレットのためのサブ関数のまとめ

カラー パレットの例は、次の表にリストしたコールバックを含みます。

関数	説明
colorCellCallback	カラー セルがクリックされるときに hPalettePanelSelectionChanged によって呼び出されます。
eraserToolCallback	消しゴムボタンがクリックされるときに hPalettePanelSelectionChanged によって呼び出されます。
hMoreColorButtonCallback	[More Colors] ボタンがクリックされると実行します。これは、uisetcolor を呼び出して、標準的なカラー選択ダイアログ ボックスを開き、localUpdateColor を呼び出してプレビューを更新します。
hPalettePanelSelectionChanged	GUI のユーザーが新しい色をクリックするときに実行します。これは、uibuttongroup が含むツールとカラー セルを排他的に管理する SectionChangeFcn コールバックです。ツールとカラー セルのそれぞれを使用可能にする適当なコールバックを呼び出します。

メモ 3 つの eventdata フィールドは ボタン グループ(uibuttongroup)とともに使用するために定義されます。これらのフィールドによって、ボタン グループで保持される以前と現在のラジオ ボタンまたはトグル ボタンの選択を決定できます。詳細は、「uibuttongroup Properties」リファレンス ページの SelectionChangeFcn を参照してください。

この例は、次の表にリストされた補助関数も含みます。

関数	説明
layoutComponent	パレットに消しゴムツールやカラー セルを動的に作成します。これは、localDefineLayout を呼び出します。
localUpdateColor	選択した色のプレビューを更新します。
getSelectedColor	現在選択された色を返します。次に colorPalette コーラーに返されます。
localDefineLayout	希望するカラー セルと GUI のツール サイズを計算します。これは、localDefineColors と localDefineTools を呼び出します。
localDefineTools	パレットに示されるツールを定義します。この例では、唯一のツールは[消しゴム] ボタンです。
localDefineColors	カラー セルの配列に示されている色を定義します。
processUserInputs	プロパティ/値の組のプロパティがサポートされるか判定します。これは、localValidateInput を呼び出します。
localValidateInput	プロパティ/値の組の値を有効にします。

M ファイルの構造

カラー パレットの GUI は入れ子関数を使用してプログラミングされます。その M ファイルは以下の順序で構成されます。

- 1 help コマンドに応答して表示されるコメント
- 2 データ作成。例は入れ子関数を使用するので、このデータを最上位に定義すると、引数としてデータを渡さなくてもすべての関数に対してデータへのアクセスが可能になります。
- 3 コマンド ライン入力処理
- 4 GUI の Figure とコンポーネントの作成
- 5 GUI の初期化

6 出力が要求される場合に出力を返す

7 コールバック定義。GUI コンポーネントを使用可能にするこれらのコールバックは関数 `colorPalette` のサブ関数で、従って、引数として渡されなくても最上位で作成されたデータとコンポーネント ハンドルへのアクセスをもちます。

8 補助関数定義。これらの補助関数は関数 `colorPalette` のサブ関数で、従って、引数として渡されなくても最上位で作成されたデータとコンポーネント ハンドルにアクセスできます。

メモ 入れ子関数の詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”について参照してください。

GUI プログラムの手法

このトピックは、`colorPalette` の作成で使用されるように、以下の GUI プログラム手法を説明します。

- ・ “GUI への入力引数渡し” (p.15-56)
- ・ “出力時にコーラーに出力を渡す” (p.15-57)
- ・ “2 つの GUI 間でデータを共有する” (p.15-58)

これらの手法と他のプログラム手法の例は、“アイコン エディター” (p.15-60) を参照してください。

GUI への入力引数渡し

GUI への入力は、カスタムのプロパティ/値の組です。`colorPalette` には、そのようなプロパティとして、`Parent` が可能です。名前は、大文字と小文字の区別をします。`colorPalette` の構文は次のようになります。

```
mGetColorFcn = colorPalette('Parent', hPaletteContainer)
```

プロパティの定義と初期化. 関数 `colorPalette` は、入力引数を受け取るために最初に変数 `mInputArgs` を `varargin` として定義します。

```
mInputArgs = varargin; % Command line arguments when invoking
% the GUI
```

すると、関数 `colorPalette` は、 3×3 セル配列に有効なカスタム プロパティを定義します。

```
mPropertyDefs = {...      % The supported custom property/value
                  % pairs of this GUI
'parent', @localValidateInput, 'mPaletteParent';
```

- ・ 1 番目の列は、プロパティ名を含みます。
- ・ 第 2 列は、入力のプロパティ値が有効であるか確認する関数 `localValidateInput` に対する関数ハンドルを含みます。
- ・ 第 3 列は、プロパティの値を保持するローカル変数です。

`colorPalette` がプロパティを既定値で初期化します。

```
mPaletteParent = []; % Use input property 'parent' to initialize
```

入力引数の処理. `processUserInputs` 補助関数は、入力のプロパティ/値の組を処理します。`colorPalette` は、コンポーネントを作成する前にコンポーネントの親を決定するために、`processUserInputs` を呼び出します。

```
% Process the command line input arguments supplied when
% the GUI is invoked
processUserInputs();
```

- 1 入力がある場合、`processUserInputs` は `mPropertyDefs` セル配列の 1 列目の中から、入力で与えた各プロパティ名と一致するものがあるか、順番にチェックします。
- 2 一致を見つけると、`processUserInputs` は、`mPropertyDefs` セル配列の第 3 列に、その変数のプロパティに対する入力であった値を割り当てます。
- 3 すると、`processUserInputs` は、`mPropertyDefs` セル配列の第 2 列に指定された補助関数を呼び出し、そのプロパティに対して渡された値を有効にします。

出力時にコーラーに出力を渡す

ホスト GUI が関数 `colorPalette` を出力引数と共に呼び出すと、これは、現在選択された色を取得するためにホスト GUI が呼び出せる関数ハンドルを返します。

ホスト GUI は、`colorPalette` を 1 度だけ呼び出します。この呼び出しは、指定した親にカラー パレットを作成してから、関数ハンドルを返します。ホスト GUI はカラー パレットが壊される前であれば、いつでも出力された関数を呼び出すことができます。

colorPalette M ファイルのデータ定義セクションは、セル配列を作成して出力を保持します。

```
mOutputArgs = {};% Variable for storing output when GUI returns
```

出力する直前に、colorPalette は関数ハンドル mgetSelectedColor をセル配列 mOutputArgs に割り当て、mOutputArgs を varargout に割り当て引数を返します。

```
mOutputArgs{} = @getSelectedColor;
if nargout>0
    [varargout{1:nargout}] = mOutputArgs{:};
end
```

2つの GUI 間でデータを共有する

次に説明する Icon Editor の例の GUI は、GUI、colorPalette を組み込み、アイコンセルの色の選択を可能にします。colorPalette は、iconEditor に関数ハンドルを返します。すると、iconEditor は、選択されたカラーを取得するためにいつでも返された関数を呼び出すことができます。以下の 2 節では、2 つの GUI がどのように連携して動作するかを説明します。

colorPalette GUI. 関数 colorPalette はセル配列 mOutputArgs を定義して、その出力引数を保持します。

```
mOutputArgs = {};% Variable for storing output when GUI returns
```

出力する直前に、colorPalette は mOutputArgs にその getSelectedColor 補助関数の関数ハンドルを割り当て、その後、mOutputArgs を varargout に割り当て、引数を返します。

```
% Return user defined output if it is requested
mOutputArgs{1} = @getSelectedColor;
if nargout>0
    [varargout{1:nargout}] = mOutputArgs{:};
end
```

iconEditor は colorPalette の関数 getSeclectedColor を、colorPalette が iconEditor に返す関数を実行するときいつでも実行します。

```
function color = getSelectedColor
% function returns the currently selected color in this
% colorPlatte
```

```
color = mSelectedColor;
```

iconEditor GUI. 関数 iconEditor は colorPalette を 1 度だけ呼び出し、その親を iconEditor のパネルに指定します。

```
% Host the ColorPalette in the PaletteContainer and keep the  
% function handle for getting its selected color for editing  
% icon.  
mGetColorFcn = colorPalette('parent', hPaletteContainer);
```

この呼び出しへは、iconEditor のコンポーネントとして colorPalette を作成し、それから、iconEditor が現在選択されている色を取得するために呼び出すことができる関数ハンドルを返します。

iconEditor の補助関数 localEditColor は mGetColorFcn を呼び出します。これは、colorPalette の関数 getSelectedColor を実行するために、colorPalette によって返される関数です。

```
function localEditColor  
% helper function that changes the color of an icon data  
% point to that of the currently selected color in  
% colorPalette  
if mIsEditingIcon  
    pt = get(hIconEditAxes, 'currentpoint');  
    x = ceil(pt(1,1));  
    y = ceil(pt(1,2));  
    color = mGetColorFcn();  
  
    % update color of the selected block  
    mIconCData(y, x, :) = color;  
  
    localUpdateIconPlot();  
end  
end
```

アイコン エディター

このセクションの内容…

“例について” (p.15-60)

“アイコン エディターの GUI M ファイルの表示と実行” (p.15-62)

“サブ関数のまとめ” (p.15-65)

“M ファイルの構造” (p.15-67)

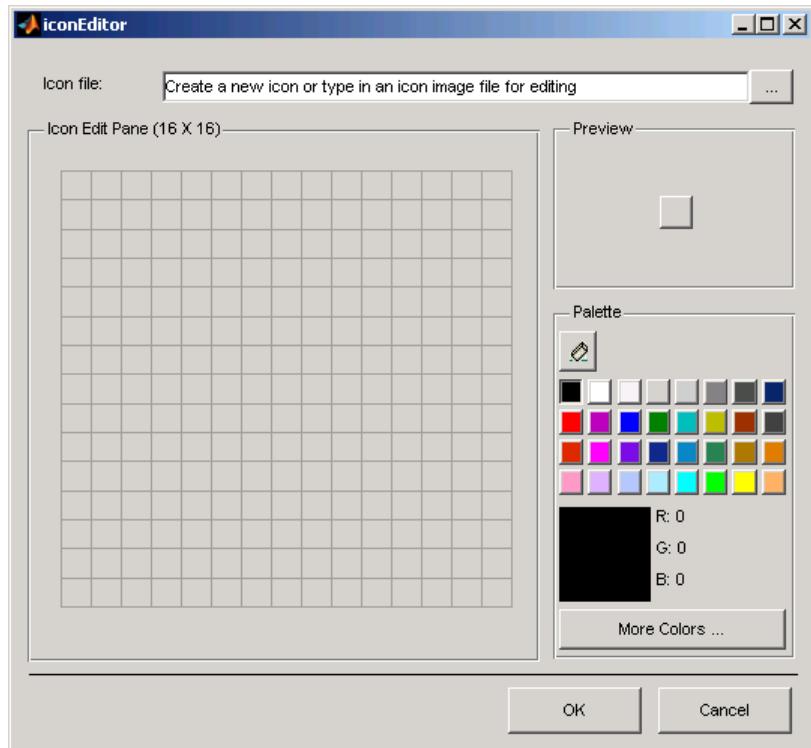
“GUI プログラムの手法” (p.15-67)

例について

この例は、アイコンの作成と編集ができる GUI を作成します。この例の M ファイルへのリンクは、“アイコン エディターの GUI M ファイルの表示と実行” (p.15-62) を参照してください。

メモ アイコン エディターの例は、チュートリアルの例として提供されています。これは、MATLAB でサポートされる機能ではありません。ただし、アイコンを編集するための同様の GUI は GUIDE から利用できます。詳細は、GUIDE ドキュメンテーションの “ツール アイコンの編集” (p.6-129) を参照してください。

下図は、エディターを示します。



アイコン エディターの GUI のコンポーネント

GUI は、以下のコンポーネントを含みます。

- ユーザーに指示したり、編集するファイル名を含むエディット テキスト ボックス。このエディット テキストは、スタティック テキストを用いてラベリングされます。
- エディット テキストの右のプッシュ ボタンによって、編集する既存のアイコン ファイルを選択できます。
- 座標軸を含むパネル。座標軸は、アイコンを描くために 16×16 グリッドを表示します。
- アイコンが作成されているときに、アイコンのプレビューを示すボタンを含むパネル
- 別のスクリプトで作成され、この GUI に組み込まれるカラー パレット。“カラー パレット” (p.15-49)を参照してください。

- ・ [OK] と [キャンセル] ボタンからアイコン エディターを分けるラインとして設定されるパネル
- ・ GUI が $m \times n \times 3$ 配列としてアイコンを返し、GUI を閉じる [OK] プッシュ ボタン
- ・ アイコンを返さずに GUI を閉じる [キャンセル] プッシュ ボタン

アイコン エディターの例で使用される手法

この例は、以下の GUI プログラミング手法を説明します。

- ・ ユーザーが選択を行うまで値を返さない GUI の作成
- ・ GUI から出力が返るときに出力を取得する
- ・ データの有効性を確かめたユーザー定義の入力のプロパティ/値の組のサポート
- ・ GUI がコマンド ラインから変更されるのを防ぐ
- ・ 複数のプラットフォーム上で実行する GUI の作成
- ・ 2 つの GUI 間でのデータ共有
- ・ 適切なサイズ変更の動作を行う

メモ この例は、入れ子関数を使用します。入れ子関数の詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”について参照してください。

アイコン エディターの GUI M ファイルの表示と実行

この例は、3 つの M ファイルと 1 つのアイコン イメージを使用します。

- ・ iconEditor.m
- ・ iconRead.m
- ・ colorPalette.m
- ・ eraser.gif

MATLAB ヘルプ ブラウザーでこのドキュメンテーションをお読みいただいている場合、以下のリンクをクリックして、例の M ファイルにアクセスできます。Web 上で、あるいは PDF でこれをお読みいただいている場合、リンクを使用して MATLAB ヘルプ ブラウザー内の対応する節をお読みください。

この例の GUI のレイアウトまたはコードを修正しようとする場合、現在のフォルダーにその M ファイルのコピーを最初に保存する必要があります。これを行うには、現在のフォルダーへの書き込み権限が必要です。例のファイルを現在のフォルダーにコピーして、それらを開くには以下のリンクをクリックします。

- 1 ここをクリックして、4 つのファイルを現在のフォルダーにコピーしてください。
- 2 `edit iconEditor.m` と入力するか、または ここをクリックすると、エディターに `iconEditor` の GUI M ファイルが開きます。
- 3 `edit iconRead.m` と入力するか、または ここをクリックすると、エディターに `iconRead` の GUI M ファイルが開きます。
- 4 `edit colorPalette.m` と入力するか、または ここをクリックすると、エディターに `colorPalette` の GUI M ファイルが開きます。
`colorPalette` M ファイルの詳細は、前の例 “カラー パレット” (p.15-49) を参照してください。

GUI を実行するか、コードを調べようとする場合、以下の手順に従います。

- 1 MATLAB パスに例のファイルを追加するには、ここをクリックします。(現在のセッションに対してのみ)
- 2 `iconEditor` GUI を実行するには、ここをクリックします。
- 3 MATLAB エディター内に `iconEditor` M ファイルを表示するには、ここをクリックします。(読み取り専用)
- 4 MATLAB エディターで `iconRead` M ファイルを表示するには、ここをクリックしてください。
- 5 MATLAB エディターで `colorPalette` M ファイルを表示するには、ここをクリックしてください。

メモ GUI ファイルを既存の examples フォルダーで保存すると、元のファイルが上書きされるので保存しないようにしてください。これらは、現在のフォルダーあるいは他の作業フォルダーに保存してください。

アイコン エディターの利用

MATLAB パスに examples フォルダーを追加、あるいは現在のフォルダーにアイコンエディターの M ファイルをコピーした後、アイコンを作成するには以下の手順に従います。

- 1 次のようなコマンドを用いてアイコン エディターを起動します。

```
myicon = iconEditor('iconwidth', 32, 'iconheight', 56);
```

ここで、iconwidth と iconheight プロパティは、アイコンのサイズをピクセル単位で指定します。

- 2 グリッドの四角に色を付けます。

- ・ パレットのカラー セルをクリックします。すると、その色がパレット プレビューに表示されます。
- ・ グリッドの特定の四角をクリックして、これらの四角に選択した色を移します。
- ・ 左マウス ボタンを押したままマウスをグリッド上にドラッグして、触れている四角に選択した色を移します。
- ・ 他の色で上書きして色を変更します。

- 3 いくつかの四角の色を消します。

- ・ パレットの[消しゴム] ボタンをクリック
- ・ これらの四角を消すために特定の四角をクリック
- ・ マウスをクリックしながらドラッグして、触れている四角を消す
- ・ カラー セルをクリックして消しゴムを無効にする

- 4 [OK] をクリックして GUI を閉じ、作成したアイコンを myicon に $32 \times 65 \times 3$ 配列として返します。[キャンセル] をクリックして GUI を閉じ、myicon に空の配列 [] を返します。

サブ関数のまとめ

icon editor の例は、次の表にリストされるコールバックを含みます。

関数	説明
hMainFigureWindowButtonDownFcn	GUI の Figure 内でユーザーがマウス ボタンをクリックするときに実行します。これは、localEditColor を呼び出します。
hMainFigureWindowButtonUpFcn	マウス ボタンをはなすときに実行します。
hMainFigureWindowButtonMotionFcn	ボタンを押したまま、Figure 内でマウスをユーザーがドラッグするとき実行します。これは、localEditColor を呼び出します。
hIconFileDialogCallback	編集するアイコンのファイル名を入力して変更した後に、実行します。これは、localUpdateIconPlot を呼び出します。
hIconFileDialogButtonDownFcn	Icon file エディットボックスを初めてクリックするとき実行します。
hOKButtonCallback	[OK] プッシュ ボタンをクリックするとき実行します。
hCancelButtonCallback	[キャンセル] プッシュ ボタンをクリックするとき実行します。
hIconFileButtonCallback	ユーザーが [Icon ファイル] プッシュ ボタンをクリックすると実行します。これは、localUpdateIconPlot を呼び出します。

この例は、次の表にリストされた補助関数も含みます。

関数	説明
localEditColor	現在選択されている色へのアイコン データ点の色を変更します。関数 colorPalette によって返される関数 mGetColorFcn を呼び出します。これは、 localUpdateIconPlot も呼び出します。
localUpdateIconPlot	アイコン プレビューを更新します。これは、アイコンがファイルから読み込まれるときに座標軸も更新します。
processUserInputs	プロパティ/値の組のプロパティがサポートされるか判定します。これは、 localValidateInput を呼び出します。
localValidateInput	プロパティ/値の組の値を有効にします。
prepareLayout	複数のプラットフォームでの実行に対し、外見と特徴に必要な変更を行います。

M ファイルの構造

iconEditor は入れ子関数を用いてプログラムされます。その M ファイルは以下の順序で構成されます。

- 1 help コマンドに応答して表示されるコメント
- 2 データ作成。例は入れ子関数を使用するので、このデータを最上位に定義すると、引数としてデータを渡さなくてもすべての関数に対してデータへのアクセスが可能になります。
- 3 GUI の Figure とコンポーネントの作成
- 4 コマンド ライン入力処理
- 5 GUI の初期化
- 6 GUI のユーザーが [OK] または [キャンセル] をクリックするまでプログラムの実行は行われません。
- 7 要求される場合に出力を返します。
- 8 コールバック定義。GUI コンポーネントを使用可能にするこれらのコールバックは、関数 iconEditor のサブ関数です。従って、引数として渡される必要がなく、最上位で作成されるデータとコンポーネントハンドルにアクセスできます。
- 9 補助関数定義。これらの補助関数は関数 iconEditor のサブ関数であり、従って、引数としてデータを渡さなくても最上位に作成されたデータとコンポーネントハンドルにアクセスできます。

メモ 入れ子関数の詳細は、MATLAB「プログラミングの基礎」ドキュメンテーションの“Nested Functions”について参照してください。

GUI プログラムの手法

このトピックは、iconEditor の作成で使用される、以下の GUI プログラム手法を説明します。

- ・ “ユーザー選択後に限られた出力” (p.15-68)
- ・ “GUI への入力引数渡し” (p.15-69)

- ・ “GUI から返される出力の取得” (p.15-70)
- ・ “不注意によるアクセスからの GUI の保護” (p.15-71)
- ・ “複数プラットフォームでの GUI の実行” (p.15-71)
- ・ “GUI をモーダルにする” (p.15-73)
- ・ “2 つの GUI 間でデータを共有する” (p.15-73)
- ・ “適切なサイズ変更動作の実現” (p.15-75)

ユーザー選択後に限られた出力

初期化コードの終わりに、出力の直前に、iconEditor は GUI をブロックするために、メイン Figure のハンドルと共に uiwait を呼び出します。

```
% Make the GUI blocking
uiwait(hMainFigure);

% Return the edited icon CData if it is requested
mOutputArgs{1} = hMainFigure;
mOutputArgs{2} = mIconCData;
if nargout>0
    [varargout{1:nargout}] = mOutputArgs{ :) ;
end
```

uiwait への呼び出しの位置は、重要です。uiwait の呼び出しは、GUI が初期化された後、ファイルが編集されたアイコン データを返す直前に iconEdit M ファイルを逐次的な実行を停止します。

[OK] ボタンをクリックすると、そのコールバック hOKButtonCallback は uiresume を呼び出します。これによって、M ファイルは停止した位置から実行を再開し、編集したアイコン データを返します。

```
function hOKButtonCallback(hObject, eventdata)
% Callback called when the OK button is pressed
    uiresume;
    delete(hMainFigure);
end
```

[キャンセル] ボタンをクリックすると、そのコールバック `hCancelButtonCallback` はアイコン データを効率的に削除し、`uiresume` を呼び出します。これによって、M ファイルは停止した位置で実行を再開できますが、null 行列を返します。

```
function hCancelButtonCallback(hObject, eventdata)
% Callback called when the Cancel button is pressed
    mIconCData =[];
    uiresume;
    delete(hMainFigure);
end
```

GUI への入力引数渡し

GUI への入力は、カスタムのプロパティ/値の組です。`iconEdit` には、そのようなプロパティとして、`IconWidth`、`IconHeight`、`IconFile` の 3 つがあります。名前は大文字と小文字を区別しません。

プロパティの定義と初期化. `iconEdit` は、最初に、変数 `mInputArgs` を `varargin` として定義し、入力引数を受け取ります。

```
mInputArgs = varargin; % Command line arguments when invoking
% the GUI
```

すると、関数 `iconEdit` は 3×3 セル配列に有効なカスタム プロパティを定義します。

```
mPropertyDefs = [...] % Supported custom property/value
% pairs of this GUI
    'iconwidth' , @localValidateInput, 'mIconWidth';
    'iconheight' , @localValidateInput, 'mIconHeight';
    'iconfile' , @localValidateInput, 'mIconFile');
```

- ・ 1 番目の列は、プロパティ名を含みます。
- ・ 第 2 列は、入力のプロパティ値が有効であるか確認する関数 `localValidateInput` に対する関数ハンドルを含みます。
- ・ 第 3 列は、プロパティの値を保持するローカル変数です。

すると、`iconEdit` はプロパティを既定値で初期化します。

```
mIconWidth = 16; % Use input property 'iconwidth' to initialize
mIconHeight = 16; % Use input property 'iconheight' to initialize
```

```
mIconFile = fullfile(matlabroot, 'toolbox/matlab/icons/' );
```

`mIconWidth` と `mIconHeight` の値は、ピクセルとして解釈されます。関数 `fullfile` は、一部分からフル ファイル名を構成します。

入力引数の処理. 補助関数 `processUserInputs` は入力のプロパティ/値の組を処理します。`iconEdit` は、レイアウトが完成した後に GUI を初期化する入力が必要になる直前に `processUserInputs` を呼び出します。

```
% Process the command line input arguments supplied when
% the GUI is invoked
processUserInputs();
```

- 1 入力がある場合、`processUserInputs` は `mPropertyDefs` セル配列の 1 列目の中から、入力で与えた各プロパティ名と一致するものがあるか、順番にチェックします。
- 2 一致を見つけると、`processUserInputs` は、`mPropertyDefs` セル配列の第 3 列に、その変数のプロパティに対する入力であった値を割り当てます。
- 3 すると、`processUserInputs` は、`mPropertyDefs` セル配列の第 2 列に指定された補助関数を呼び出し、そのプロパティに対して渡された値を有効にします。

GUI から返される出力の取得

`iconEditor` を出力引数と共に呼び出すと、 $n \times m \times 3$ 配列としてトゥルーカラー イメージを出力します。

M ファイルのデータ定義部分は、出力を保持するためにセル配列を作成します。

```
mOutputArgs = {};% Variable for storing output when GUI returns
```

M ファイルの逐次的な実行を停止する `uiwait` への呼び出しに従い、`iconEdit` は作成されたアイコンの配列 `mIconEdit` をセル配列 `mOutputArgs` に割り当て、次に、引数を返すために `mOutputArgs` を `varargout` に割り当てます。

```
mOutputArgs{} = mIconCData;
if nargout > 0
    [varargout{1:nargout}] = mOutputArgs{:};
end
```

このコードは、出力前に `iconEditor` が実行する最後のコードです。これは、[OK] または [キャンセル] ボタンをクリックして、`hOKButtonCallback` または

`hCancelButtonCallback` の実行をトリガーした後にのみ実効します。これは、`uiresume` を呼び出し、実行を再開します。

不注意によるアクセスからの GUI の保護

`prepareLayout` ユーティリティ関数は、すべてのコンポーネントの HandleVisibility プロパティを設定することで、`iconEditor` がコマンド ラインから不注意に変更されることを防ぎます。`iconEditor` は、M ファイルの初期化の部分でメイン Figure の関数ハンドルと共に `prepareLayout` を呼び出します。

```
% Make changes needed for proper look and feel and running on
% different platforms
prepareLayout(hMainFigure);
```

`prepareLayout` は、最初に `findall` を使用して、Figure に含まれるすべてのオブジェクトのハンドルを検索します。検索されたハンドルのリストは `colorPalette` を含みます。これは、`iconEditor` とその子に組み込まれます。Figure のハンドルは入力引数 `topContainer` として `prepareLayout` に渡されます。

```
allObjects = findall(topContainer);
```

次に、`prepareLayout` は `Callback` をもつすべてのこれらのオブジェクトの HandleVisibility プロパティを設定します。

```
% Make GUI objects available to callbacks so that they cannot
% be changed accidentally by other MATLAB commands
set(allObjects(isprop(allObjects,'HandleVisibility')),...
    'HandleVisibility','Callback');
```

`HandleVisibility` を `Callback` に設定すると、コールバック ルーチンやコールバック ルーチンにより呼び出された関数内から GUI ハンドルが可視になりますが、コマンド ラインから呼び出された関数内からは可視ではありません。これによって、現在の Figure である GUI をユーザーがコマンド ラインから不注意に変更できなくなります。

複数プラットフォームでの GUI の実行

`prepareLayout` ユーティリティ関数は、すべての GUI コンポーネントの様々なプロパティを設定して、GUI が複数のプラットフォーム上で、適切な外見や特徴をもつように保ちます。`iconEditor` は、M ファイルの初期化の部分でメイン Figure の関数ハンドルと共に `prepareLayout` を呼び出します。

```
% Make changes needed for proper look and feel and running on
```

```
% different platforms
prepareLayout(hMainFigure);
```

最初に、`prepareLayout` は `findall` を使用して、`Figure` に含まれるすべてのオブジェクトのハンドルを検索します。検索されたハンドルのリストは、`colorPalette` も含みます。これは、`iconEditor` とその子に埋め込まれます。`Figure` のハンドルは入力引数 `topContainer` として `findall` に渡されます。

```
function prepareLayout(topContainer)
...
allObjects = findall(topContainer);
```

背景色. 既定のコンポーネント背景色は GUI が実行しているシステムの標準のシステム背景色です。この色はコンピューターシステムが異なると変わります。たとえば、PC の標準的なグレーの影は UNIX システムのものとは異なり、既定の GUI 背景色とは一致しないことがあります。

関数 `prepareLayout` は、GUI の背景色を既定のコンポーネント背景色と同じに設定します。これは、他のアプリケーション GUI と同様に、GUI 内で同じ外見を提供します。

これは、最初にルート オブジェクトから既定のコンポーネント背景色を検索します。次に、`Figure` の `Color` プロパティを使用して、GUI 背景色を設定します。

```
defaultColor = get(0,'defaultuicontrolbackgroundcolor');
if isa(handle(topContainer),'figure')

...
% Make figure color match that of GUI objects
set(topContainer,'Color',defaultColor);
end
```

Units の選択. 関数 `prepareLayout` は、GUI がサイズ変更される可能性を考慮して、使用する単位を決めます。これは `strcmpi` を使用して、GUI の `Resize` プロパティの値を決めます。結果によって、すべてのオブジェクトの `Units` プロパティを `normalized` あるいは `characters` に設定します。

```
% Make the GUI run properly across multiple platforms by using
% the proper units
if strcmpi(get(topContainer,'Resize'),'on')
    set(allObjects(isprop(allObjects,'Units')),...
```

```

    ' Units' , ' Normalized' );
else
    set(allObjects(isprop(allObjects, ' Units' )), ...
        ' Units' , ' Characters' );
end

```

サイズ変更可能な Figure に対して、規格化された単位は、Figure と各コンポーネントの左下隅を(0,0)に右上隅を(1.0,1.0)にマップします。このため GUI が表示されるとき、コンポーネントのサイズはその親のサイズに自動的に調整されます。

サイズ変更不可能な Figure に対して、GUI が異なるコンピューター上に表示されるとき、character units はコンポーネントのサイズと相対的な間隔を自動的に調整します。

文字単位は、既定のシステム フォントからの文字によって定義されています。文字単位の幅は、システム フォントにおいて文字 x の幅に等しくなります。文字単位の高さは、テキストの 2 行のベースライン間の距離です。文字単位が正方形でないことに注意してください。

GUI をモーダルにする

iconEditor は、モーダル Figure です。モーダルな Figure は、すべての通常の Figure と MATLAB コマンド ウィンドウの上に重なったままになります。これにより、ユーザーは他のウィンドウとのやりとりを可能できずに、応答させます。iconEditor は、メイン Figure の WindowStyle プロパティを modal に設定して、モーダルにします。

```

hMainFigure = figure(...  
    ...  
    ' WindowStyle' , ' modal' , ...

```

WindowStyle プロパティの利用についての詳細は、MATLAB 関数リファレンス ドキュメンテーションの「Figure プロパティ」を参照してください。

2 つの GUI 間でデータを共有する

iconEditor は GUI、colorPalette を組み込み、アイコン セルの色の選択を可能にします。colorPalette は、関数ハンドルによって、選択した色を iconEditor に返します。

colorPalette GUI. iconEditor のように、colorPalette はセル配列 mOutputArgs を定義し、その出力引数を保持します。

```
mOutputArgs = [] ; % Variable for storing output when GUI returns
```

出力する直前に、colorPalette は mOutputArgs にその getSelectedColor 補助関数の関数ハンドルを割り当て、その後、mOutputArgs を varargout に割り当て、引数を返します。

```
% Return user defined output if it is requested
mOutputArgs{1} =@getSelectedColor;
if nargout>0
    [varargout{1:nargout}] = mOutputArgs{::};
end
```

iconEditor は colorPalette の 関数 getSeclectedColor を、colorPalette が iconEditor に返す関数を実行するときいつでも実行します。

```
function color = getSelectedColor
% function returns the currently selected color in this
% colorPlatte
    color = mSelectedColor;
```

iconEditor GUI. 関数 iconEditor は colorPalette を 1 度だけ呼び出し、その親を iconEditor のパネルに指定します。

```
% Host the ColorPalette in the PaletteContainer and keep the
% function handle for getting its selected color for editing
% icon.
mGetColorFcn = colorPalette('parent', hPaletteContainer);
```

この呼び出しが、iconEditor のコンポーネントとして colorPalette を作成し、それから、iconEditor が現在選択されている色を取得するために呼び出すことができる関数ハンドルを返します。

iconEditor の localEditColor 補助関数は mGetColorFcn を呼び出します。これは、colorPalette の関数 getSelectedColor を実行するために、colorPalette によって返される関数です。

```
function localEditColor
% helper function that changes the color of an icon data
% point to that of the currently selected color in
% colorPalette
if mIsEditingIcon
    pt = get(hIconEditAxes, 'currentpoint' );
    x = ceil(pt(1,1));
```

```

y = ceil(pt(1,2));
color = mGetColorFcn();
% update color of the selected block
mIconCData(y, x, :) = color;
localUpdateIconPlot();
end
end

```

適切なサイズ変更動作の実現

`prepareLayout` ユーティリティ関数は、すべての GUI コンポーネントの `Units` プロパティを設定して、GUI を複数のプラットフォーム上で適切にサイズ変更可能とします。`iconEditor` は、M ファイルの初期化の部分でメイン Figure の関数ハンドルと共に `prepareLayout` を呼び出します。

```
prepareLayout(hMainFigure);
```

最初に、`prepareLayout` は `findall` を使用して、Figure に含まれるすべてのオブジェクトのハンドルを検索します。検索されたハンドルのリストは `colorPalette` を含みます。これは、`iconEditor` とその子に組み込まれます。Figure のハンドルは入力引数 `topContainer` として `findall` に渡されます。

```

function prepareLayout(topContainer)
...
allObjects = findall(topContainer);

```

次に、`prepareLayout` は `strcmpi` を使用して GUI がサイズ変更可能かどうかを判定します。結果によって、すべてのオブジェクトの `Units` プロパティを `normalized` あるいは `characters` に設定します。

```

if strcmpi(get(topContainer, 'Resize'), 'on')
    set(allObjects(isprop(allObjects, 'Units')), ...
        'Units', 'Normalized');
else
    set(allObjects(isprop(allObjects, 'Units')), ...
        'Units', 'Characters');
end

```

メモ `Figure` の `Resize` プロパティの既定値 `on` を受け取るので、`iconEditor` はサイズ変更可能です。

サイズ変更可能な `Figure`. 規格化された単位は、`Figure` と各コンポーネントの左下隅を $(0,0)$ に、右上隅を $(1.0,1.0)$ にマップします。このため、GUI がサイズ変更されるとき、コンポーネントのサイズはその親のサイズに対して自動的に変更されます。

サイズ変更不可能な `Figure`. GUI が異なるコンピューター上に表示されるとき、`character units` はコンポーネントのサイズと相対的な間隔を自動的に調整します。

文字単位は、既定のシステム フォントからの文字によって定義されています。文字単位の幅は、システム フォントにおいて文字 x の幅に等しくなります。文字単位の高さは、テキストの 2 行のベースライン間の距離です。文字単位が正方形でないことに注意してください。

例

Use this list to find examples in the documentation.

Simple Examples (GUIDE)

- “例:簡単な GUI” (p.2-9)
- “操作確認のためのモーダル ダイアログの使用” (p.10-98)

Simple Examples (Programmatic)

- “例:簡単な GUI” (p.3-2)

Application Examples (GUIDE)

- “多くのコンポーネントをもつ GUI の取り扱い” (p.6-24)
- “複数の座標軸をもつ GUI” (p.10-2)
- “3-D 表示のアニメーションのための GUI” (p.10-15)
- “テーブルのデータを対話的に調べる GUI” (p.10-32)
- “リスト ボックス ディレクトリ リーダー” (p.10-54)
- “リスト ボックスからワークスペース変数にアクセス” (p.10-61)
- “Simulink モデル パラメーターを設定する GUI” (p.10-66)
- “アドレス ブック リーダー” (p.10-81)

Programming GUI Components (GUIDE)

- “プッシュ ボタン” (p.8-30)
- “トグル ボタン” (p.8-32)
- “ラジオ ボタン” (p.8-32)
- “チェック ボックス” (p.8-33)
- “エディット テキスト” (p.8-34)
- “スライダー” (p.8-36)
- “リスト ボックス” (p.8-36)
- “ポップアップ メニュー” (p.8-38)
- “パネル” (p.8-39)
- “ボタン グループ” (p.8-42)
- “座標軸” (p.8-44)

“ActiveX コントロール” (p.8-48)
“メニュー項目” (p.8-59)

Application-Defined Data (GUIDE)

“データを UserData と共有する” (p.9-11)
“アプリケーション データとデータを共有する” (p.9-14)
“GUI データとデータを共有する” (p.9-17)
“例 – ユーザーの入力に対するモーダル ダイアログ ボックスの取り扱い” (p.9-23)
“例 – アイコン操作ツールとして協同している個々の GUIDE GUI” (p.9-31)

GUI Layout (Programmatic)

“ファイル テンプレート” (p.11-4)
“チェック ボックス” (p.11-15)
“エディット テキスト” (p.11-16)
“リスト ボックス” (p.11-19)
“ポップアップ メニュー” (p.11-21)
“テーブル” (p.11-23)
“プッシュ ボタン” (p.11-25)
“ラジオ ボタン” (p.11-26)
“スライダー” (p.11-27)
“スタティック テキスト” (p.11-29)
“トグル ボタン” (p.11-30)
“パネル” (p.11-33)
“ボタン グループ” (p.11-35)
“座標軸の追加” (p.11-37)
“ActiveX コントロールの追加” (p.11-40)

Programming GUI Components (Programmatic)

“チェック ボックス” (p.12-21)

- “テキストの編集” (p.12-21)
- “リスト ボックス” (p.12-23)
- “ポップアップメニュー” (p.12-24)
- “プッシュ ボタン” (p.12-25)
- “ラジオ ボタン” (p.12-26)
- “スライダー” (p.12-26)
- “トグル ボタン” (p.12-27)
- “パネル” (p.12-28)
- “ボタン グループ” (p.12-28)
- “座標軸のプログラミング” (p.12-30)
- “ActiveX コントロールのプログラミング” (p.12-34)
- “メニュー項目のプログラミング” (p.12-34)
- “ツールバー ツールのプログラミング” (p.12-36)

Application-Defined Data (Programmatic)

- “入れ子関数の例:コンポーネント間でのデータ渡し” (p.13-11)
- “UserData プロパティの例:コンポーネント間でのデータ渡し” (p.13-16)
- “アプリケーション データの例:コンポーネント間でのデータ渡し” (p.13-18)
- “GUI データの例:コンポーネント間でのデータ渡し” (p.13-21)

Application Examples (Programmatic)

- “座標軸、メニュー、ツールバーを含む GUI” (p.15-3)
- “テーブルのデータを表示してグラフを作成する GUI” (p.15-17)
- “リストのデータを管理する GUI” (p.15-31)
- “カラー パレット” (p.15-49)
- “アイコン エディター” (p.15-60)

A

ActiveX controls
 adding to GUI layout 6–75
 programming 8–48 12–34
aligning components
 in GUIDE 6–87
Alignment Tool
 GUIDE 6–87
application data
 appdata functions 9–5 13–6
application-defined data
 application data 9–5 13–6
 GUI data 9–7 13–8
 in GUIDE GUIs 9–1
 UserData property 9–4 13–5
axes
 designating as current 8–48
 multiple in GUI 10–2 10–15
axes, plotting when hidden 10–76

B

background color
 system standard for GUIs 6–136 11–94
backward compatibility
 GUIDs to Version 6 5–4
button groups 6–23 11–10
 adding components 6–34

C

callback
 arguments 8–13
 defined 12–7
 self-interrupting 10–22
callback execution 14–2
callback templates
 adding to a GUIDE GUI 8–9
callback templates (GUIDE)
 add comments 5–7

callbacks
 attaching to GUIs 12–11
 GUIDE restrictions on 8–9
 renaming in GUIDE 8–20
 sharing data 9–10
 specifying as cell arrays 12–14
 specifying as strings 12–12
 used in GUIDE 8–2
 used in GUIs 12–8
check boxes 8–33 12–21
color of GUI background 5–13
colorPalette GUI example
 about 15–49
 accessing 15–53
 programming 15–56
 structure 15–55
 subfunctions 15–54
 techniques used 15–53
command-line accessibility of GUIs 5–10
compatibility across platforms
 GUID design 6–135
component identifier
 assigning in GUIDE 6–37
component palette
 show names 5–6
components for GUIs
 GUIDE 6–20
components in GUIDE
 aligning 6–87
 copying 6–79
 cutting and clearing 6–79
 front-to-back positioning 6–80
 moving 6–81
 pasting and duplicating 6–79
 resizing 6–84
 selecting 6–78
 tab order 6–96
confirmation
 exporting a GUI 5–2
 GUID activation 5–2

- context menus
 associating with an object 6–118
 creating in GUIDE 6–99
 creating with GUIDE 6–112
 menu items 6–114
 parent menu 6–112
- cross-platform compatibility
 GUI background color 6–136 11–94
 GUI design 6–135
 GUI fonts 6–135 11–93
 GUI units 6–137 11–95
- D**
- data
 sharing among GUI callbacks 9–10
- default system font
 in GUIs 6–135 11–93
- dialog box
 modal 10–98
- E**
- edit box
 setting fonts of 11–18
- edit text 8–34 12–21
- edit text input
 validation of 8–34
- exporting a GUI
 confirmation 5–2
- F**
- FIG-file
 generate in GUIDE 5–14
 generated by GUIDE 5–11
- files
 GUIDE GUI 7–2
- fixed-width font
 in GUIs 6–136 11–93
- fonts
- using specific in GUIs 6–136 11–94
- function prototypes
 GUIDE option 5–11
- G**
- graphing tables
 GUI for 15–17
- GUI
 adding components with GUIDE 6–19
 application-defined data (GUIDE) 9–1
 command-line arguments 8–25
 compatibility with Version 6 5–4
 designing 6–2
 GUIDE options 5–9
 help button 10–78
 laying out in GUIDE 6–1
 naming in GUIDE 7–2
 opening function 8–25
 renaming in GUIDE 7–3
 resize function 10–95
 resizing 5–10
 running 7–10
 saving in GUIDE 7–4
 standard system background color 6–136 11–94
 using default system font 6–135 11–93
 with multiple axes 10–2 10–15
- GUI のサイズ変更関数 10–95
- GUI components
 aligning in GUIDE 6–81
 GUIDE 6–20
 how to add in GUIDE 6–31
 moving in GUIDE 6–81
 tab order in GUIDE 6–96
- GUI data
 application-defined data 9–7 13–8
- GUI example
 axesMenuToolbar 15–3
 colorPalette 15–49
 iconEditor 15–60

- listmaster 15–31
- tableplot 15–17
- GUI export
 - confirmation 5–2
- GUI FIG-files
 - opening 10–60
- GUI files
 - in GUIDE 7–2
- GUI initialization
 - controlling for singleton 9–9
- GUI layout in GUIDE
 - copying components 6–79
 - cutting and clearing components 6–79
 - moving components 6–81
 - pasting and duplicating components 6–79
 - selecting components 6–78
- GUI object hierarchy
 - viewing in GUIDE 6–134
- GUI options (GUIDE)
 - function prototypes 5–11
 - singleton 5–11
 - system color background 5–11
- GUI singleton
 - initialization of 9–9
- GUI size
 - setting with GUIDE 6–15
- GUI template
 - selecting in GUIDE 6–5
- GUI to manage lists
 - example 15–31
- GUI units
 - cross-platform compatible 6–137 11–95
- GUIDE
 - adding components to GUI 6–19
 - application examples 10–1
 - application-defined data 9–1
 - command-line accessibility of GUIs 5–10
 - coordinate readouts 6–81
 - creating menus 6–99
 - generate FIG-file only 5–14
- generated M-file 5–11
- grids and rulers 6–94
- GUI background color 5–13
- GUI files 7–2
- help menu 2–4
- how to add components 6–31
- laying out using coordinates 6–33
- Object Browser 6–134
- preferences 5–2
- renaming files 7–3
- resizing GUIs 5–10
- saving a GUI 7–4
- selecting template 6–5
- starting 6–4
- tables 6–64
- tool summary 4–3
- toolbar editor 6–122
- video demos 2–4
- what is 4–2
- GUIDE callback templates
 - add comments 5–7
- GUIDE GUI
 - opening 2–27
- GUIDE GUI の名前変更 7–3
- GUIDE GUIs
 - figure toolbars for 6–121
- GUIs
 - multiple instances 5–12
 - single instance 5–12

H

- handles structure
 - adding fields 9–8 13–10
- help button for GUIs 10–78
- hidden figure, accessing 10–76

I

- identifier

assigning to GUI component 6-37
interrupting callback
example of 10-22

L

Layout Editor
show component names 5-6
Layout Editor window
show file extension 5-7
show file path 5-7
list boxes 8-36 12-23
example 10-54
listmaster GUI example
about 15-31
accessing 15-34
operating 15-35
programming 15-40
techniques used 15-33

M

M-file
generated by GUIDE 5-11
menu item
check 8-60 12-36
menus
callbacks 8-59 12-34
context menus in GUIDE 6-112
creating in GUIDE 6-99
drop-down menus 6-101
menu bar menus 6-101
menu items 6-107 6-114
parent of context menu 6-112
pop-up 8-38 12-24
specifying properties 6-105
modal dialogs
about 10-98
moving components
in GUIDE 6-81

N

naming a GUI
in GUIDE 7-2
NextPlot
problems with in GUIs 11-40

O

Object Browser (GUIDE) 6-134
options
GUIDE GUIs 5-9

P

panels 6-23 11-10
adding components 6-34
pop-up menus 8-38 12-24
preferences
GUIDE 5-2
Property Editor
for tables 6-67
pushbutton
alternating label of 10-21

R

radio buttons 8-32 12-26
ResizeFcn
for scaling text 8-39
resizing components
in GUIDE 6-84
resizing GUIs 5-10
running a GUI 7-10

S

saving GUI
in GUIDE 7-4
shortcut menus
creating in GUIDE 6-112
singleton GUI

-
- defined 5–12
 - GUIDE option 5–11
 - size of GUI
 - setting with GUIDE 6–15
 - sliders 6–21 11–11
 - status bar
 - show in GUIDE Layout Editor 6–18
 - system color background
 - GUIDE option 5–11

 - T**
 - tab order
 - components in GUIDE 6–96
 - Tab Order Editor 6–96
 - Table Property Editor 6–67
 - tables
 - for GUIs 6–64
 - Tag property
 - assigning in GUIDE 6–37
 - template for GUI
 - selecting in GUIDE 6–5
 - toggle buttons 8–32 12–27
 - toolbar
 - show in GUIDE Layout Editor 6–18
 - Toolbar Editor
 - using 6–122
 - toolbar menus

 - U**
 - uibuttongroups
 - adding to a GUI 6–36
 - uicontrols
 - validating input from users 10–11
 - uipanel
 - ResizeFcn for 8–39
 - uipanels
 - adding to a GUI 6–36
 - uitable
 - graphing from 15–17
 - units for GUIs
 - cross-platform compatible 6–137 11–95
 - UserData property
 - application-defined data 9–4 13–5

 - V**
 - validating
 - user input in uicontrols 10–11
 - video demos
 - for GUIDE 2–4