CONTENTS

ビジュアルヘルプ - コントロールの追加とコントロールパネルの作成

はじめに

サンプルのエクスペリメントを見ると、テーブルやグラフ以外に値をコントロールするためのコントロールパネルのようなものがたくさん見られます。
これらは、Igor Pro 標準の機能ではなく、その分析に合わせた処理を行いやすくするために Igor プログラマー（上級ユーザー）が開発したものです。

グラフィカルユーザーインターフェース（GUI）を作成するために使うことができるさまざまなオブジェクトを、「コントロール」という用語で表します。
これらをコントロールと呼びますが、オブジェクトの中には値を表示するだけのものもあります。
ウィジェットという言葉が、他のアプリケーションプログラムでは使われることがあります。

使うことができるコントロール

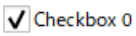
ボタン（Button）

プログラマーが作成したプロシージャを呼び出します。

チャート（Chart）

機械式チャート記録器をエミュレートします。
チャートは、データ収集プロセスの監視や、長期間のデータ記録の調査に使用できます。
チャートのプログラミングはかなり複雑です。
詳しくは、FIFO とチャートの説明を参照してください。

チェックボックス（Checkbox）

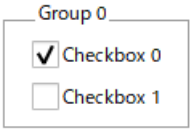


プログラマーが作成したプロシージャで使うためのオフ/オン値を設定します。

カスタムコントロール（CustomControl）

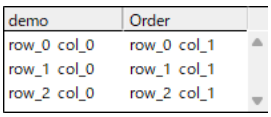
カスタムのコントロール形式です。
プログラマーが自由に指定、修正できます。

グループボックス（GroupBox）

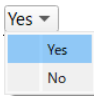
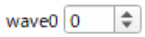
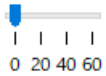


コントロールをボックスまたはラインで囲み、グループ化して表示します。

リストボックス（ListBox）



表示または選択するための項目をリストアップします。

ポップアップメニュー（PopupMenu） 	プログラマーが作成したプロシージャで使う値をユーザーが選択するために使います。
変数の設定（SetVariable） 	数値または文字列のグローバル変数を設定し、表示します。 ユーザーはクリックまたはキー入力により変数を設定できます。 数値変数に対しては、変数に格納された値を増減するためのアップ/ダウンボタンをコントロールに含めることができます。
スライダー（Slider） 	機械的なスライダーの動作を再現します。 離散的または連続的な値を選択できるようにします。
タブコントロール（TabControl）	複雑なパネル内のコントロールのグループを選択できるようにします。
タイトルボックス（TilteBox）	説明テキストまたはメッセージを表示します。
値の表示（ValDisplay） 	通常はグローバル変数を参照する数値式の読み取り値を表示します。 読み取り値は、数値テキスト、温度計バー、またはその両方の形式で表示できます。

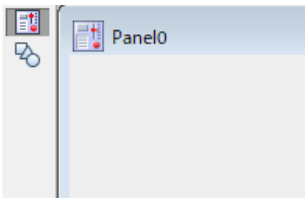
プログラマーは、ユーザーがコントロールをクリックまたは入力した時に呼び出されるプロシージャを指定することができます。
これをコントロールのアクションプロシージャと呼びます。
例えば、ボタンのアクションプロシージャは、PopupMenu、CheckBox、SetVariable コントロールの値を照会し、その後、何らかのアクションを実行します。

コントロールパネルは、これらのコントロールを含むシンプルなウィンドウです。
これらのウィンドウには、それ以外の目的はありません。
また、グラフウィンドウやグラフに埋め込まれたパネルペイン（領域）にコントロールを配置することもできます。
コントロールは、テーブル、ノートブック、レイアウトなどの他のウィンドウ形式では使うことはできません。
グラフで使われる場合、コントロールはプレゼンテーションの一部とは見なされないため、グラフを印刷またはエクスポートする時には含まれません。

操作のモード

コントロールに関しては、操作には 2 つのモードがあります：コントロールを使うモードと、コントロールを編集するモードです。

これを確認するには、Graph または Panel メニューから Show Tools を選択します。
ウィンドウの左上隅に 2 つのアイコンが表示されます。



一番上のアイコンが選択されている場合、コントロールを使うことができます。
次のアイコンが選択されている場合、描画ツールパレットが 2 番目のアイコンの下に表示されます。
コントロールを変更するには、描画ツールパレットから矢印ツールを選択します。

一番上のアイコンが選択されている場合、またはアイコンが非表示になっている場合は、使用または操作モードになっています。
Ctrl+Alt キーを押すと、一時的に編集モードまたは描画モードに切り替えることができます。
コントロールのドラッグやサイズ変更、ダブルクリックをするにはこれを使います。
Ctrl+Alt キーを押しながらダブルクリックすると、コントロールを変更するダイアログが表示されます。

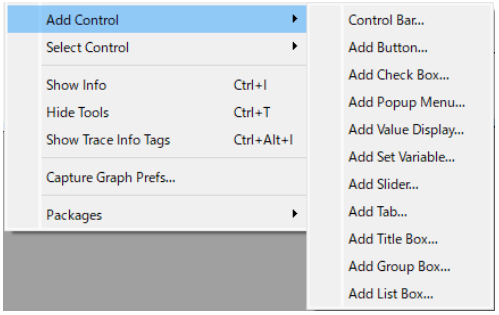
Graph または Panel メニューの Select Control サブメニューから項目を選択して、編集モードに切り替えることもできます。

重要： Graph メニューと Panel メニューで Add Controls サブメニューを有効にするには、変更モードにする必要があります。
2 番目のアイコンをクリックするか、Ctrl+Alt キーを押しながら Add Controls サブメニューを選択します。

各コントロールの詳細

Graph メニューと Panel メニューで Add Controls サブメニューから、追加したいコントロールを選択します。
（以降で説明するうちのいくつかはメニューに含まれません。主にプログラミングで実装します。）

作成手順の詳細は後半で説明します。

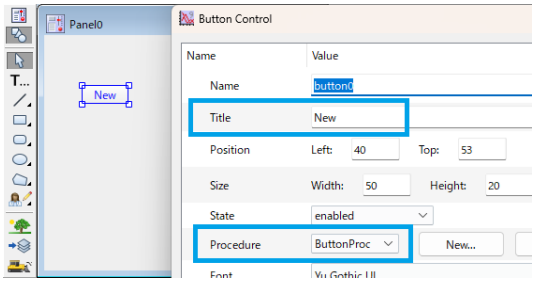


ボタン（Button）

ボタンをクリックすると、プログラマーが指定したプロシージャが実行されます。

ボタンをクリックしても何も起こらない場合は、そのボタンに割り当てられたプロシージャが存在しないことを意味します。
プロシージャウィンドウがコンパイルされていない場合、割り当てられたプロシージャを持つボタンをクリックすると、エラーダイアログが表示されます。

この状況を修正するには、Macros メニューから Compile を選択してください。
エラーが発生しなくなれば、ボタンは機能するようになります。



ボタンは通常、丸みを帯びた外観ですが、プログラマーがカスタム画像を割り当てれば、どのような外観にもすることができます。

チャート（Chart）

これは Add Controls メニューにはありません。
Chart コントロールは、ペンが紙の上を移動しながら、その下で紙がスクロールし、ペンが紙に書き込んでいく機械式チャートレコーダーをエミュレートするために使うことができます。
チャートは、データ取得プロセスの監視や、長期間にわたるデータ記録の調査に使うことができます。

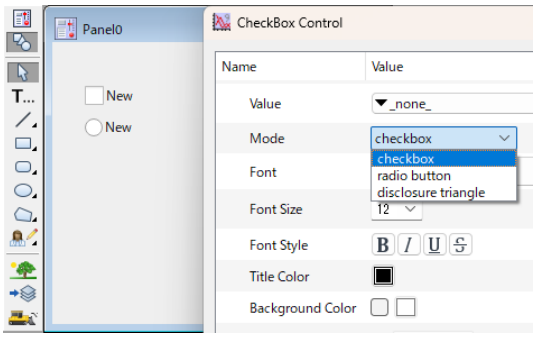
チャートコントロールの使用に関する詳細な説明については、Using Chart Recorder Controls（Advanced Topics.ihf）を参照してください。

チャートのプログラミングはかなり複雑ですが、チャートの使用は実際には非常に簡単です。
詳細は、FIFOs and Charts（Advanced Topics.ihf）を参照してください。

チェックボックス（Checkbox）

チェックボックスをクリックすると、選択状態が変わり、プログラマーが指定したプロシージャを実行することができます。

チェックボックスはグローバル変数に接続することができます。
チェックボックスは、ラジオボタンのように表示し、動作するように設定することができます。

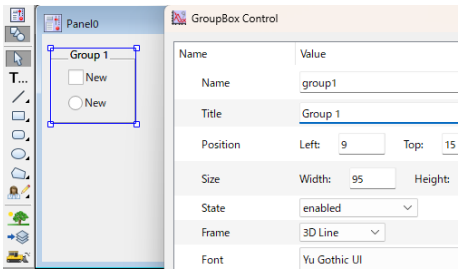


カスタムコントロール（CustomControl）

これは Add Controls メニューにはありません。
カスタムコントロールは、プログラマーがカスタムメイドするまったく新しいタイプのコントロールを作成するために使います。
カスタムコントロールの外観や動作のすべての側面を定義し、コントロールすることができます。

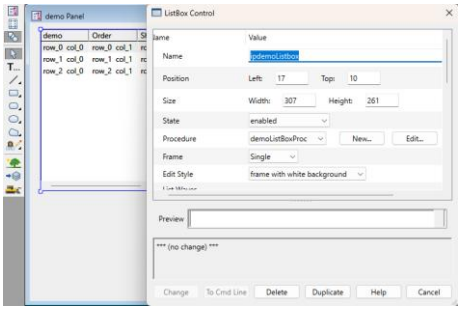
グループボックス（GroupBox）

GroupBox コントロールは、整理や装飾のための要素です。
複数のコントロールを視覚的にグループ化するために使います。
ボックスまたはセパレーターラインを描画でき、オプションでタイトルを付けることもできます。



リストボックス（ListBox）

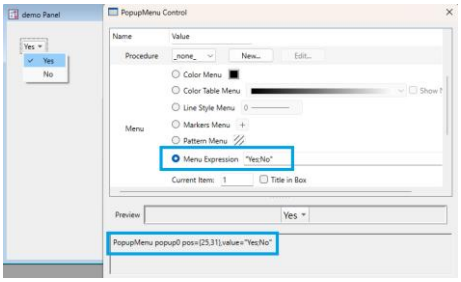
ListBox コントロールは、表示または選択用の 1 つまたは複数の列からなるアイテムのリストを表示することができます。
ListBox は、さまざまな選択モードに設定することができます。
リスト内のアイテムは編集可能にすることができ、チェックボックスとして設定することもできます。



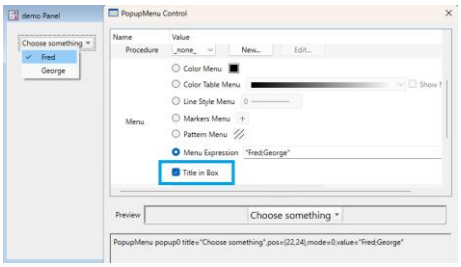
ポップアップメニュー

このコントロールには２つの形式があります。
１つは現在の項目がポップアップメニューボックスに表示される形式です。

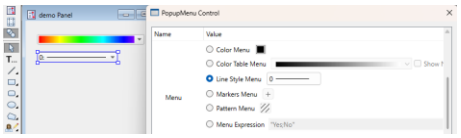
もう１つは、現在のアイテムがなく、ボックス内にタイトルが表示される場合です。



最初の形式は通常、複数の項目の中から１つを選択するために使われ、２番目の形式は複数のコマンドの中から１つを実行するために使われます。



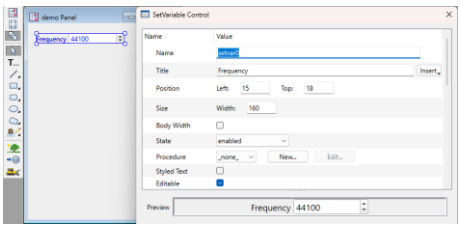
ポップアップメニューは、Igor の色、線種、パターン、マーカーのポップアップメニューと同様に動作するよう設定することもできます。
これらは常に現在の項目を表示します。



SetVariable

SetVariable コントロールも様々な形式を取り、数値を表示することができます。
式の結果を表示する Value Display コントロールとは異なり、SetVariable コントロールは個々のグローバル変数やウェーブ要素に接続され、それらの変数の現在の値を読み出すだけでなく、それらの変数を設定または変更するために使うことができます。
SetVariable コントロールは、グローバル文字列変数やテキストウェーブ要素と組み合わせて、短い１行の文字列を表示または設定することもできます。
SetVariable コントロールは、関連付けられている変数が変更されるたびに自動的に更新されます。
値は内部に保存される場合もあります。

数値変数に接続された場合、これらのコントロールにはオプションで上下矢印を表示でき、プログラマーが指定した量だけ変数の現在の値を増減させます。
また、プログラマーは数値表示の上限と下限を設定できます。



数値変数と文字列変数の両方に、コントロールに直接入力することで新しい値を設定できます。
編集領域内をクリックすると、コントロールが編集モードになります。

その後、カット、コピー、ペーストなどの標準的な操作で表示テキストを編集できます。
変更を破棄する場合は、Escape キーを押してください。
変更を確定するには、Return キー、Enter キー、Tab キーを押すか、コントロールの外側をクリックしてください。
Tab キーを押すと現在の値が入力され、次のコントロールがある場合はそこに移動します。
Shift-Tab キーも同様ですが、前のコントロールがある場合にそこに移動します。

コントロールが数値を表示している場合、入力したテキストが数値に変換できないと、値を入力しようとするとビープ音が鳴り、値は変更されません。

入力しようとしている値がプログラマーが設定した制限を超える場合、入力値は最も近い制限値に置き換えられます。

数値コントロールが編集用に選択されている場合、上矢印キーと下矢印キーは、そのコントロールの上下ボタンと同様に動作します。

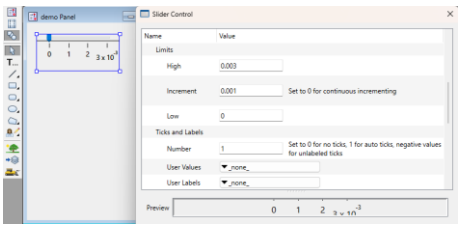
SetVariable コントロールの値を変更すると、プログラマーがプロシージャを指定している場合、そのプロシージャが実行される可能性があります。

SetVariable コントロールとデータフォルダー

グローバル変数またはウェーブ要素に接続された SetVariable コントロールは、変数が存在するデータフォルダーを記憶し、現在のデータフォルダーがコントロール対象の変数と異なる場合でも正常に機能し続けます。
「SetVariable」のセクションを参照してください。

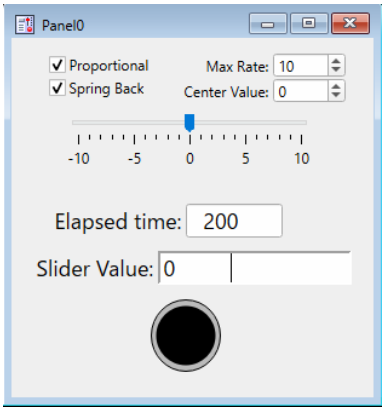
システム変数（K0 から K19）は特定のデータフォルダーに属さず（どのデータフォルダーからでも利用可能）、これらの変数は 1 つのコピーしか存在しません。
現在のデータフォルダーが「aFolder」の状態で K0 をコントロールする SetVariable を作成し、別の SetVariable を現在のデータフォルダーが「bFolder」の状態で K0 をコントロールするように設定した場合、実際には同じ K0 をコントロールしています。

スライダー



スライダーコントロールは、離散値または連続値をグラフィカルに選択するために使用できます。
離散値の選択に使う場合、スライダーはポップアップメニューやラジオボタンのセットに類似しています。
スライダーはライブに動作させることができ、ユーザーがスライダーをドラッグする時に変数を更新したりプロシージャを実行したりできます。
あるいは、ユーザーが操作を完了するまで待機するように設定することも可能です。

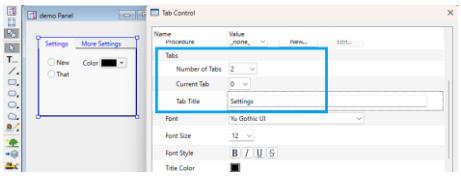
繰り返しスライダー



スライダーは、ユーザーがスライダーを操作している間、アクションプロシージャを繰り返し呼び出すように設定できます。
一定の速度で動作させるか、値に比例した速度で動作させるか設定でき、オプションで離した時に元の値にスプリングバックさせることができます。
この機能は Igor Pro 8.0 で追加されました。

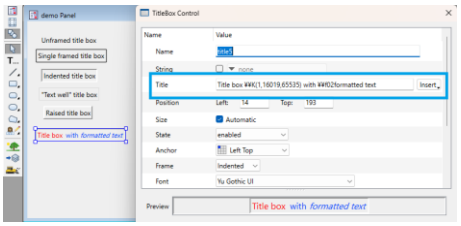
繰り返しスライダーを実装するには、Slider コマンドで repeat キーワードを使います。
デモについては、次のデモエクスペリメントを参照してください：File→Example Experiments→Feature Demos 2→Slider Repeat Demo
Slider Repeat Demo。

TabControl



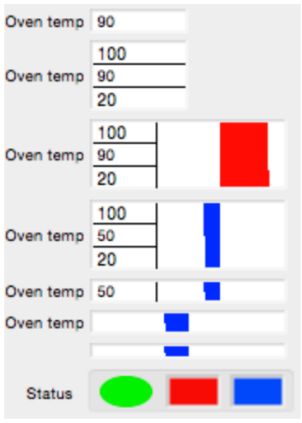
TabControls は、通常では収まらないほど多くのコントロールを含む複雑なパネルを作成するために使われます。
ユーザーがタブをクリックすると、プログラマーが設定したプロシージャが実行され、前のコントロール群を非表示にしつつ新しいコントロール群を表示します。

TitleBox



TitleBox コントロールは主に装飾要素です。
Insert ポップアップメニューで装飾を選択します。
これらは、コントロールパネル内で説明テキストやラベルを提供するために使われます。
また、テキストの結果を表示するためにも使用できます。
テキストは固定でも、グローバル文字列変数の内容でもかまいません。
いずれの場合も、ユーザーが誤ってテキストを変更することはありません。

ValDisplays

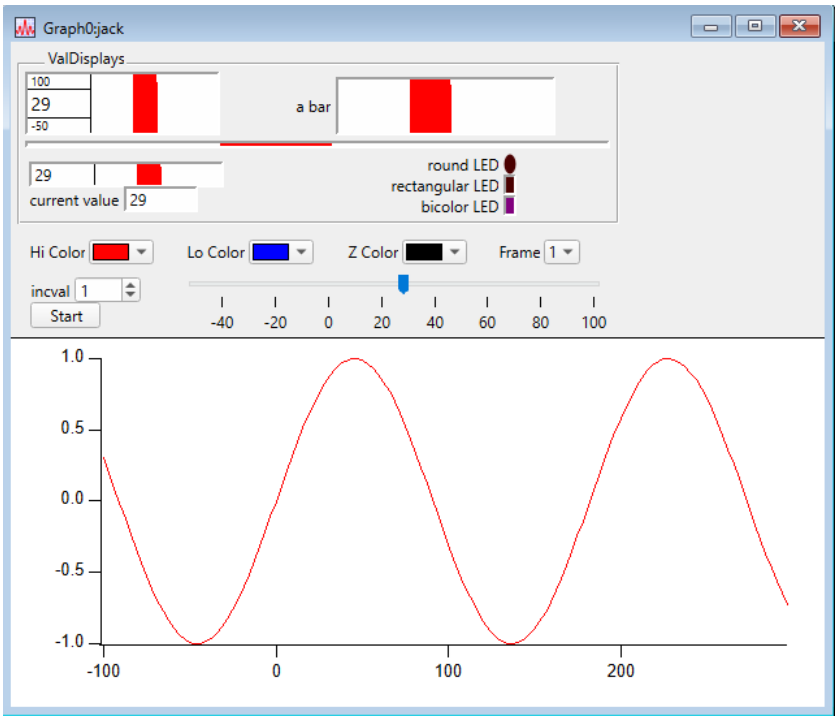


ValDisplay コントロールは、単純な数値表示から温度計バーまで、さまざまな形式で数値または文字列値を表示します。
形式にかかわらず、ValDisplay は単なる表示です。
ユーザーインタラクションはありません。
プログラマが指定した式（通常は数値変数の値）の現在の値を表示します。
ユーザー定義関数や外部関数の呼び出しを含む、あらゆる数値式を指定できます。

右は、ValDisplay コントロールが取り得る形式の例です。

温度計バーが表示される場合、温度計領域の左端はプログラマーが設定した下限値を表し、右端は上限値を表します。
上限値と下限値は右の例の一部で確認できます。
バーはプログラマーが設定した基準値から描画され、現在の値が基準値を超えると赤色に、基準値を下回ると青色に表示されます。
この例では基準値は 60 です。
基準値の数値表示はありません。
基準値が下限値より小さい場合、バーは左から右へ伸びます。
基準値が上限値より大きい場合、バーは右から左へ伸びます。

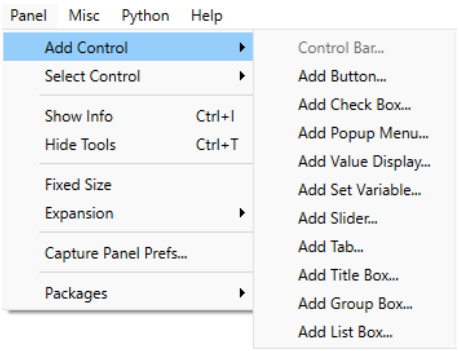
File→Example Experiments→Feature Demos→ValDisplay Demo に使用例があります。



コントロールの作成

コントロールの作り易さはコントロールによって大きく異なります。
簡単なプロシージャが書ける人なら誰でもボタンやチェックボックスを作成できますが、チャートやカスタムコントロールの作成にはより高度な知識が必要です。
ほとんどのコントロールは、Graph メニューまたは Panel メニューの Add Control サブメニュー、あるいは描画モードまたはガイドモードのパネルで使用できるパネルコンテキストメニューの Add Control サブメニューから呼び出すダイアログを使って作成および変更できます。

Add Control メニューまたは Add Control サブメニューは、ガイドモードまたは描画モードが有効な場合にのみ使用できます（これらのモードの詳細については、「操作モード」のセクションを参照してください）。
これを行うには、Graph または Panel メニューから Show Tools を選択し、グラフまたはパネルツールパレットの上から 2 番目のアイコンをクリックします。

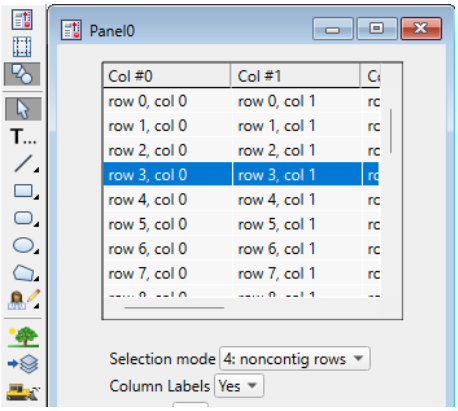


Ctrl+Alt キーを押すと、ツールパレットを表示せずに描画モードで一時的に矢印ツールを使用できます。
これらのキーを押している間、通常は無効な Add Controls サブメニューが有効になります。

描画モード

Panel→Show Tools を選択するか、パネルウィンドウ内で右クリックして Show Tools を選択して、描画モードに入ります。
上から 3 番目のボタンをクリックして描画モードを有効にします。
これにより、描画オブジェクトを作成するためのツールパレットが表示されます。

下部には、描画オブジェクトやコントロールを操作するためのメニューを表示するボタンがあります。
最上部のツールボタンは矢印ツールまたは選択ツールです。



矢印ツールでコントロールをクリックすると、コントロールのサイズ変更が可能な小さなハンドルが表示されます。
一部のコントロールは、この方法でサイズ変更できない場合や、一方向のみサイズ変更できる場合があります。
コントロールのサイズ変更を試みて、サイズが変更できない場合は、このことを理解できるでしょう。
矢印ツールを使って、コントロールの位置を変更することもできます。
コントロールは、Graph または Panel メニューの Select Control サブメニューから名前で選択できます。

矢印ツールを使うと、ほとんどのコントロールをダブルクリックして、そのコントロールを変更または複製するダイアログを表示できます。
チャートとカスタムコントロールはダイアログをサポートしていません。

コントロールを右クリックすると、そのコントロールの種類に応じて異なるコンテキストメニューが表示されます。

描画モード（ツールパレットの３番目のボタン）では、複数のコントロールを選択したり、選択範囲と描画オブジェクトを混在させたり、移動、切り取り、コピー、貼り付け、削除、整列などの操作を実行したりすることができます。
これらの操作は元に戻すことができます。
描画オブジェクトのようにコントロールをグループ化することはできません。
Select All を選択すると、すべての描画オブジェクトとコントロールが選択されます。

コントロールをあるウィンドウから別のウィンドウにコピーしたい場合は、Edit メニューの Copy と Paste を使ってください。
Copy と Paste でコントロールを複製することも可能です。

コントロールをクリップボードにコピーすると、コマンドとコントロール名もテキストとしてコピーされます。
これはプログラミングに便利です。
コントロールを右クリックして Copy Commands を選択すると、そのコントロールを作成するコマンドがクリップボードにコピーされます。
１つ以上のコントロールを選択し、右クリックして Copy Names を選択すると、セミコロン区切りのコントロール名のリストがコピーされます。

Alt キーを押しながら Edit→Copy を選択すると、コピーしたコントロールを作成するための完全なコマンドがコピーされます。

ウィンドウの端や下端からコントロールをコピーした場合、より小さいウィンドウに貼り付けると表示されないことがあります。
表示するには、Mover ツールパレットアイコンの Retrieve サブメニューを使ってください。

ガイドモード

ツールパレットの上から２番目のアイコンを選択すると、ガイドモードに入ります。
ガイドモードでは、コントロールをガイドラインに固定できます。
これにより、コントロールの位置合わせやサイズ調整、所有ウィンドウ内での適切なレイアウト維持が可能になります。
詳細は「ガイドモードでのコントロール配置」のセクションを参照してください。

コントロールのガイドモードは、サブウィンドウのレイアウトにガイドを使う機能と似ています。
ヘルプ Guides Mode for Laying Out Subwindows を参照してください。

ガイドモードでは、ガイドを使って複数のコントロールをレイアウト用に選択できます。
描画オブジェクトはガイドと連動せず、ガイドモードでは選択できません。

一般的なコマンドの構文

すべてのコントロールコマンドは、次の一般的な構文を使います。

```
ControlOperation Name [,keyword[=value] [,keyword[=value]]...]
```

Name はコントロールの名前です。
このコントロールを含むウィンドウ内のコントロール間で一意である必要があります。
Name がまだ使われていない場合、新しいコントロールが作成されます。
同じ名前のコントロールが既に存在する場合、そのコントロールが変更され、同じ名前を使う複数のコマンドが 1 つのコントロールのみを結果とするようになります。
これは、多くのキーワードを必要とするコントロールを作成するのに便利です。

すべてのキーワードはオプションです。
すべてのコントロールがすべてのキーワードを受け入れるわけではなく、一部のコントロールはキーワードを受け入れますが、実際にはその値を使いません。
あるタイプのコントロールで使われるキーワードの値は、別のタイプのコントロールで使われる同じキーワードの値とは異なる形式になる場合があります。
詳細は、Igor Pro リファレンスの各コントロールコマンドに関するドキュメントを参照してください。

一部のコントロールは、フォーマット文字列を設定するためにフォーマットキーワードを利用します。
フォーマット文字列は、 1 つの数値を期待する printf 形式の任意のフォーマットで指定できます。
出力は次のコマンドの結果であると考えてください。

```
Printf formatString, value_being_displayed
```

printf 形式文字列の詳細については、printf コマンドのヘルプを参照してください。
形式文字列の最大長は 63 バイトです。
この形式は数値を表示するコントロール要素にのみ使われます。

ほとんどのコントロールは、マウスやキーボードと対話できます。
このようなコントロールは、ユーザーのアクションに応じて、オプションでユーザー定義関数を呼び出すことができます。
このような関数をアクションプロシージャと呼びます。
コントロールのアクションプロシージャは、アクションプロシージャの呼び出しをトリガーしたアクションに関する情報をカプセル化した特別な構造体である 1 つのパラメーターを受け取ります。
各コントロールには、それぞれ独自の構造体があります。
コントロールの構造体に関する情報は、各コントロールタイプのドキュメントを参照してください。

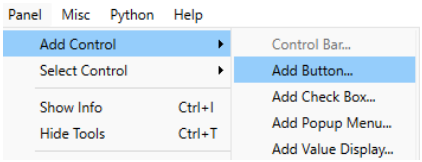
古いバージョンでは、コントロール用のアクションプロシージャは複数のパラメーターを受け取り、各パラメーターが 1 ビットの情報を保持していました。
こうしたアクションプロシージャは現在も文書化されており、動作もしますが、使用は非推奨です。
新しいコードでは、構造体ベースのアクションプロシージャを使ってください。
これらははるかに高性能で、WaveMetrics が新機能を追加しやすくなっています。
旧式のアクションプロシージャは、既存コードの動作を維持するためだけに維持されています。

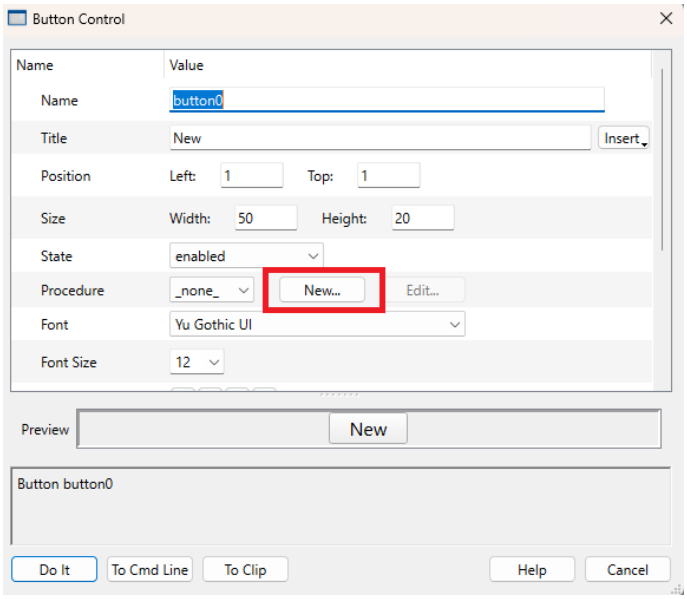
各コントロールの作成ダイアログでは、適切なパラメーターを持つ空のユーザー関数を作成できます。

ボタンコントロールの作成

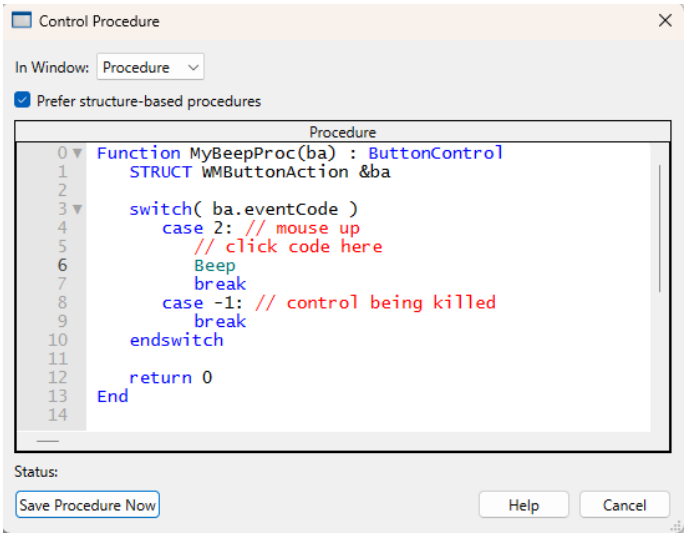
Button コマンドでは、タイトルテキストがボタン中央に配置された標準またはカスタムボタンを作成または変更します。
デフォルトのフォントはオペレーティングシステムに依存しますが、フォント、フォントサイズ、テキストカラーを変更でき、注釈のようなエスケープコードを使用できます（ヘルプ Annotation Escape Codes を参照）。
テキストの量によってボタンのサイズは変化せず、任意のサイズに設定できます。

ここでは、押すとビープ音を鳴らすだけのシンプルなボタンを作成します。
グラフまたはパネルのウィンドウを表示し、Graph または Panel メニューから Add Button を選択し、Button Control ダイアログを呼び出します。





Procedure の New ボタンをクリックすると、編集・名前変更・保存が可能なプロシージャテンプレートを含むダイアログが表示されます。
ここでは標準の ButtonControl テンプレートから開始し、デフォルト名を MyBeepProc に置き換え、Beep コマンドを追加しました。



これらのコントロール機能は、２つの形式でプロシージャと連携できます。
Igor 3 と 4 で使われていた従来のプロシージャ形式と、Igor 5 で導入された「構造ベース」形式です。

Prefer structure-based procedures チェックボックスは、新しいコードには構造ベースのプロシージャが推奨されるため、デフォルトでチェックされています。
テンプレート編集前にこのチェックボックスをオフにすると、Igor はテンプレートを旧式のプロシージャ形式に切り替えます。
旧形式の使用は推奨されません。

ヘルプをクリックすると、ボタンの操作に関するヘルプが表示されます。
ヘルプの詳細セクションでは、WMButtonAction 構造体のメンバーに関する情報を確認できます。

ダイアログ内のコントロールのアクションプロシージャを作成できるという事実から、そのプロシージャはボタンとともに保存されていると誤解されるかもしれません。
しかし、それは事実ではありません。
実際には、そのプロシージャは Procedure ウィンドウに保存されています。
これにより、複数のコントロールに対して同じアクションプロシージャを使うことが可能になります。
コントロールの構造には、コントロール名と、そのコントロールを所有するウィンドウの名前が含まれており、同じプロシージャを使う個々のコントロールを区別することができます。

アクションプロシージャの使用に関する詳細は、「コントロール構造」、ヘルプ Button、Using Structures with Windows and Controls を参照してください。

ボタンコントロールの例

以下は、タイトルが「Start」と「Stop」で交互に切り替わるボタンの作成方法です。

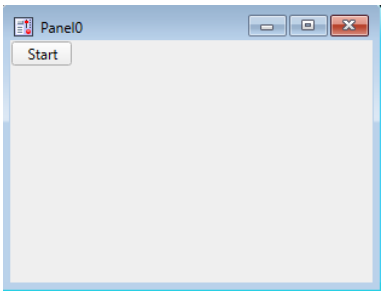
Procedure ウィンドウに以下を入力します。

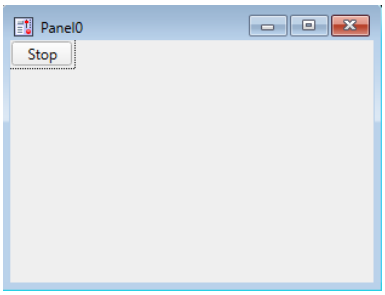
```
Function MyStartProc()
	Print "Starting"
End

Function MyStopProc()
	Print "Stopping"
End

Function StartStopButton(ba) : ButtonControl
	STRUCT WMButtonAction &ba

	switch (ba.eventCode)
		case 2:                                                    // マウス UP
			if (CmpStr(ba.ctrlName,"bStart") == 0)
				Button $ba.ctrlName,title="Stop",rename=bStop
				MyStartProc()
			else
				Button $ba.ctrlName,title="Start",rename=bStart
				MyStopProc()
			endif
			break
	endswitch

	return 0
End
```

そして、次を実行します。

```
NewPanel
Button bStart,size={50,20},proc=StartStopButton,title="Start"
```

ボタンをクリックすると、Start／Stop が切り替わります。





カスタムボタンコントロールの例

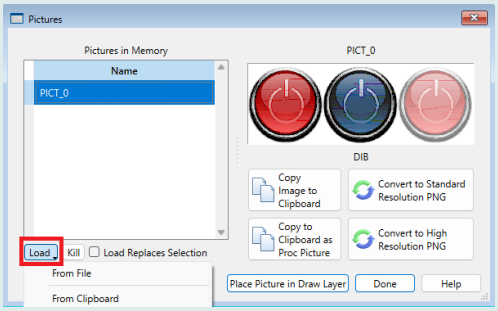
以下の手順に従ってカスタムボタンを作成できます。

1. **画像編集プログラムを使って、ボタンが通常（リラックス）状態、押下状態、無効状態を示す画像を作成してください。**
   画像の各部分は同じサイズである必要があります。

   ボタンが背景に溶け込む場合、パネルで使う背景上でボタンを作成すると見栄えが良くなります。
   さらに良いのは背景を透明にした状態で画像を作成することです。

2. **画像をクリップボードにコピーします。**

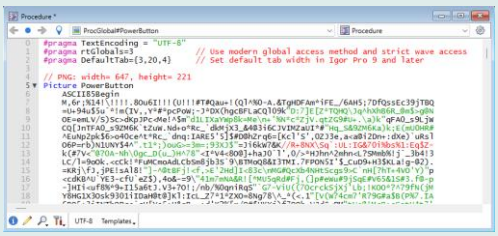
3. **Igor に切り替え、Misc→Pictures を選択し、Load→From Clipboard をクリックします。**



4. **Copy to Clipboard as Proc Picture をクリックすると、クリップボードに Proc Picture のテキストが作成されます。**

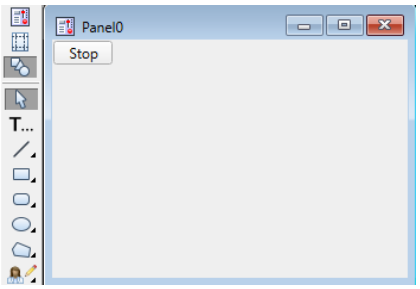
   Done をクリックします。

5. **Procedure ウィンドウを開き、テキストを貼り付け、画像に適切な名前を付けます（ここでは PowerButton と付けました）。**

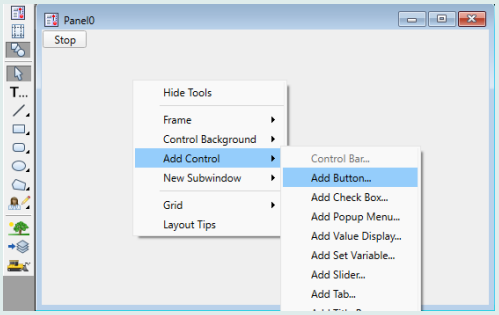


6. **既存のコントロールパネルを有効にするか、新しいコントロールパネルを作成します。**

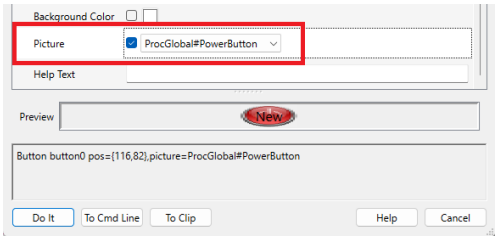
   ツールパレットが表示されていない場合は、Panel→Show Tools を表示を選択してください。



7. **Panel→Add Control→Add Button を選択するか、パネル内で右クリックして Add Control→Button を選択すると、Button Control ダイアログが表示されます。**

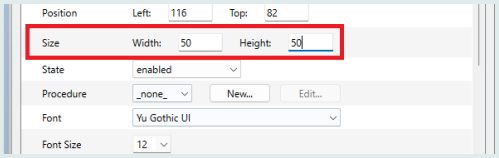


8. **ダイアログ内の Picture 設定を見つけ、チェックボックスをオンにし、ポップアップメニューからプロシージャの画像（ここでは PowerButton）を選択します。**



9. **ダイアログ内の Size 設定を見つけ、ボタンに適切なサイズを設定してください。**

   ボタンサイズは縦横同じサイズです。
   設定したサイズに合わせて画像が歪みます。



10. **Do It をクリックします。**

    ボタンが作成され、マウスで押下すると切り替わります。

チャートコントロールの作成

Chart コマンドはチャートコントロールを作成または変更します。
チャートコントロールにはダイアログによるサポートがありません。
機能的なチャートコントロールを作成するには、少なくとも中級レベルの Igor プログラミングスキルが必要です。

詳細は、ヘルプ FIFOs and Charts を参照してください。

チェックボックスコントロールの作成

CheckBox コマンドは、チェックボックス、ラジオボタン、または開示コントロールを作成または変更します。

CheckBox コントロールは、高さおよび幅の両方で自動的にサイズが調整されます。
オプションで、グローバル変数またはウェーブ内の要素に接続できます。

例えばチェックボックスをラジオボタンとして使う方法については、CheckBox のリファレンスドキュメントを参照してください。

Checkbox コントロールプロシージャは次の形式です。

```
Function CheckProc(cba) : CheckBoxControl
        STRUCT WMCheckboxAction &cba

        switch (cba.eventCode)
                case 2:                         // マウス UP
                        Variable checked = cba.checked
                        break
                case -1:                        // コントロールを Kill
                        break
        endswitch

        return 0
End
```

チェックされた構造体メンバーは、新しいチェックボックス値（0 または 1）に設定されます。

チェックボックスの状態は ControlInfo コマンドで読み取れるため、チェックボックスに対してプロシージャを定義する必要はほとんどありません。

カスタムボタンと同様の手順（カスタムボタンコントロールの例を参照）でカスタムチェックボックスを作成できます。
ただし、画像には 3 つではなく 6 つの状態が並列に存在します。
チェックボックスの状態は以下の通りです。

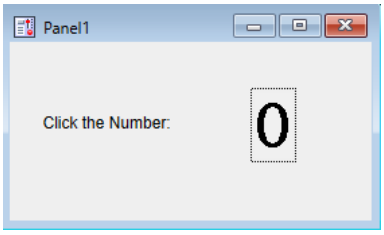
画像の順序	**コントロール状態**
Left	選択解除が有効
	選択解除が有効化され、クリックダウン状態（選択されようとしている）
	選択解除が無効
	選択が有効
	選択が有効化され、クリックダウン状態（選択解除されようとしている）
Right	選択が無効

カスタムコントロールの作成

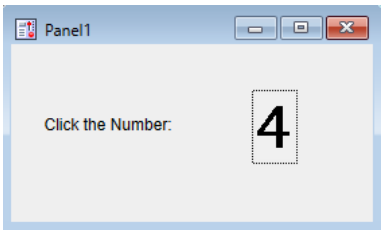
CustomControl コマンドは、そのすべての側面が完全にプログラマーによって定義されるカスタムコントロールを作成または変更します。

このセクションの例は、Custom Controls Demo（File→Example Experiments→Feature Demos 2）でも見ることができます。

カスタムコントロールで作成できるものは、クリックするとインクリメントするこのカウンターのように、かなりシンプルなものもあります。



４回クリックすると、

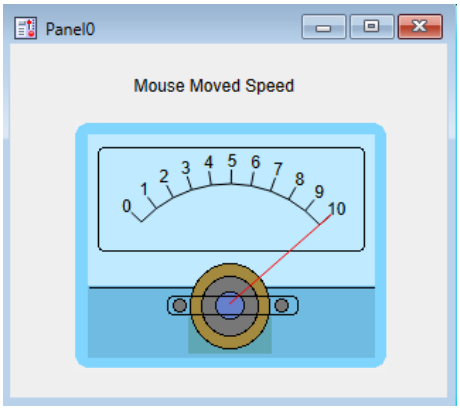


次のコードは、kCCE_frame イベントを使って上記のカウンターカスタムコントロールを実装します。
パネル上で数字をクリックするとカウンターがインクリメントされます。
また、コントロールの外側をクリックしたままドラッグしてみてください。

```
static constant kCCE_mouseup= 2
static constant kCCE_frame= 12

// PNG: 幅 = 280, 高さ = 49
Picture Numbers0to9
        ASCII85Begin
        M,6r;%14!¥!!!!.8Ou6I!!!$:!!!!R#Qau+!00#^OT5@]&TgHDFAm*iFE_/6AH5;7DfQssEc39jTBQ
        =U"5QO:5u`*!m@2jnj"La,mA^'a?hQ[Z.[.,Kgd(1o5*(PSO8oS[GX%3u'11dTl)fII/"f-?Jq*no#
        Qb>Y+UBKXKHQpQ&qYW88I,Ctm(`:C^]$4<ePf>Y(L¥U!R2N7CEAn![N1I+[hTtr.VepqSG4R-;/+$3
        IJE.V(>s0B@E@"n"ET+@5J9n_E:qeR_8:Fl?m1=DM;mu.AEj!)]K4CUuCa4T=W)#(SE>uH[A4¥;IG/
        e]FqJ4u,2`*p=N5sc@qLD5bH89>gIBdF-1i6SF28oH@"3c2m)bDr&,UB$]i]/0bA.=qbR2#¥-D9E?O
        2>3D>`($p(Kn)F8aF@)LYiXn[h2K):5@^kF?94)j*1Xtq1U2oFZmY.te?0G)EQ%5,RVT-c)DVa+%mP
        %+bS*_hN$hC*8uCJuIWqTHJR.U?32`_B)(g_8e#*YXa>=faEdJsF]6iJlrQ@QAX7huJUmXj8:PBTb2
        Y:DYf*Sci'Q"3_;@RDQA:A/([2sO8r$hW)¥B$XBGASJ:6OpC+GL<FjVfeNm20U<l<9J%cndX3'HP+k
        R.IV?U>ns*¥_;Zt[]6G6"Rb-*'Nm-E8]LXXXo7Ub>A**7Bm5cS*">HbQ&_RhmUe]$iu@T?Cci:e-¥_`k
        sE+H.GRSMT(9to;IZuH`T4%Yt<jF$+W?Yh6Q*_`C4sGig=L@DKoT%.H=#e_H"QEeeBVNTWBSMYr3dj
        O=T%d&4kT9#cWPHS>kAG;3=or2(IK*IBF$^qK,+m0NSDK_!+e0#3fAI>HfKa<sk0641u¥W@r+Y:$.i
        i$grCPR#&6,;+>nTs_IKS6XcYR)A$fJiC6Z_d2S!$R>_ZH+[<p:JI0ub]¥BhE(0RP@((KTRTGo;#SY
        LT^9;D7X#km%UV20?$RS"FZoIF!(`FY-iL?n$%#o;-Wj(¥PaBS6ZRQe@:kC>%ULrhTWLNM=n@fUbRp
        SKkLe¥kJ)Sd]u7!?pRJk-!XL[/MZX'"n4?a?JIKO0k'KUm1IZ+roB=:Bq'$&E<#$Krp%p,E"4sI>[-
        0F#^ff5SN':2fO)LNC?L4(2ga=!aLm8)tVbGAM?L`l^=$D_YP7Z(sOFs)BL5er5G95p3?m%hM^lSr'
        *E^O@8=u6hL`L$mPcq!Bl-iHuGA6hiip%`cFjl9>W?'E-&5T%Y.]i2A@1i%p8XJ5[khb:&"JXYSC¥r
        10Ss8<Ye;S^"Nc0%-DFouAiPQ9OemnR!"sHH$JKt@!"d0E"'M(P%:`p'15_10`!<nVt"TALQ>PF8WL
        Z:#f!!!!j78?7R6=>B
        ASCII85End
End

Structure CC_CounterInfo
        Int32 theCount          // 10 フレーム連続の数値列における現在のフレーム
EndStructure

Function MyCC_CounterFunc(s)
        STRUCT WMCustomControlAction &s

        STRUCT CC_CounterInfo info

        if( s.eventCode==kCCE_frame )
                StructGet/S info,s.userdata
```

```
                s.curFrame= mod(info.theCount+(s.curFrame!=0),10)
        elseif( s.eventCode==kCCE_mouseup )
                StructGet/S info,s.userdata
                info.theCount= mod(info.theCount+1,10)
                StructPut/S info,s.userdata             // コントロールに出力
        endif

        return 0
End

Window Panel_CustomControl() : Panel
        PauseUpdate; Silent 1                           // ウィンドウを構築
        NewPanel /W=(69,93,271,194)
        SetDrawLayer UserBack
        DrawText 26,62,"Click the Number:"
        CustomControl cc2,pos={152,26},proc=MyCC_CounterFunc,picture={ProcGlobal#Numbers0to9,10}
EndMacro
```

さらに高度なコントロール、例えば、次のような電圧計コントロールを作成することもできます。



Custom Controls Demo を開いて、このコントロールを試すとともに、それを実装するコードを確認してください。

グループボックスコントロールの作成

GroupBox コマンドは、グループボックスコントロールを作成または変更します。
関連するコントロール群の視覚的なコンテナを作成します。

リストボックスコントロールの作成

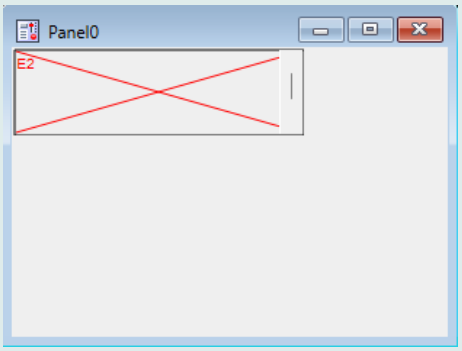
ListBox コマンドは、リストボックスコントロールを作成または変更します。

リストボックスの作成例を説明します。

1. リストボックス コントロールを含むパネルを作成します。

```
NewPanel
ListBox list0 size={200,60},mode=1
```

最も単純な機能を持つリストボックスには、リスト項目を格納するためのテキストウェーブが少なくとも１つ必要です。
テキストウェーブがない場合、リストボックスコントロールにはリスト項目が存在しません。
この状態では、リストボックスはコントロール上に赤い×印が表示された状態で描画されます。



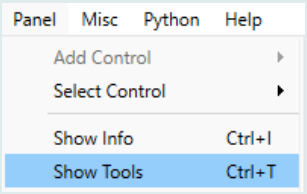
2. リスト項目を格納するテキストウェーブを作成します。

```
Make/O/T textWave0 = {"first item in list", "second item in list", "etc..."}
```

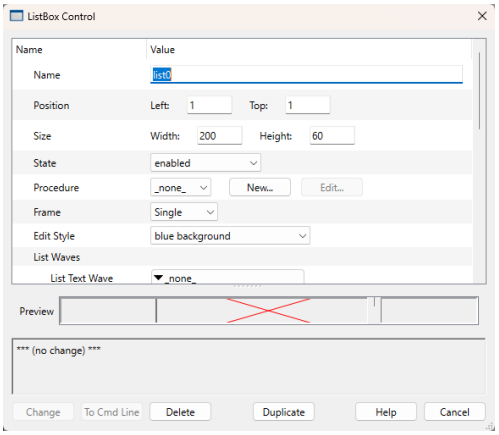
3. Panel→Show Tools を選択します。

３つのツールボタンの下端が選択されていることを確認してください。

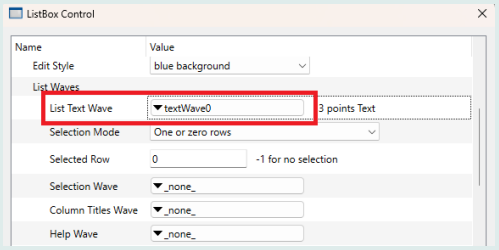
これによりパネルが描画モードになり、コントロールを変更できるようになります。



4. リストボックスコントロールをダブルクリックして、ListBox Control ダイアログを呼び出します。

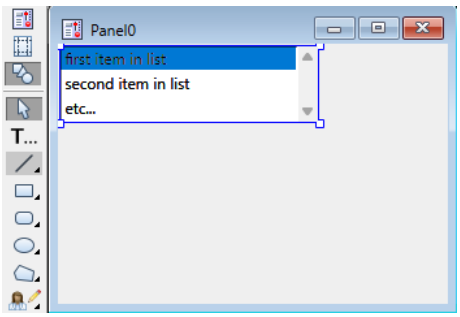


5. List Text Wave プロパティで、作成したウェーブ（textWave0）を選択し、リストのテキストウェーブとして割り当てます。

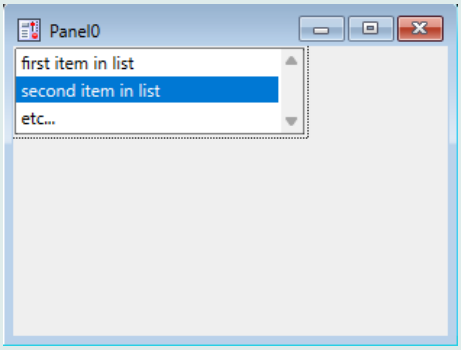


6. **Change をクリックします。**

   基本的なリストボックスコントロールが作成されました。



7. **操作（上部）アイコンをクリックするか、Panel→Hide Tools を選択すると、リストを編集するのではなく使用できるようになります。**

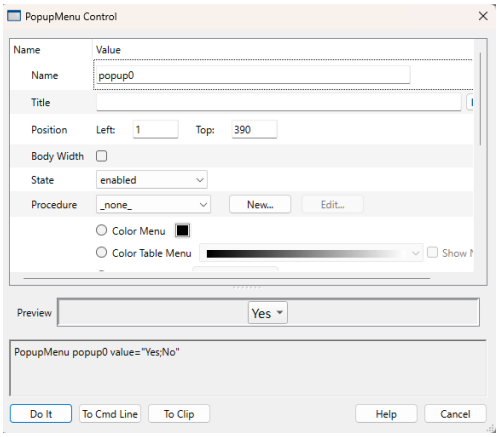


この例では、１つの単一選択リストを作成しました。
ControlInfo を呼び出し、出力変数 V_Value を確認することで、その選択内容を照会できます。

リストボックスコントロールには多くの異なるオプションとバリエーションがあります。
リストボックスコントロールのデモエクスペリメントもあります。
File→Example Experiments→Feature Demos 2→ListBox Demo

ポップアップメニューコントロールの作成

PopupMenu コマンドは、ポップアップメニューコントロールを作成または変更します。
ポップアップメニューは通常、テキスト項目の選択肢を提供するために使われますが、色、カラーテーブル、線種、パターン、マーカーを表示することもできます。



このコントロールは、タイトルまたは現在選択されているメニュー項目に応じて自動的にサイズを調整します。
bodyWidth キーワードを指定すると、ポップアップメニューの本体（タイトル以外の部分）を固定サイズに強制できます。
これにより、幅を均等に揃えて整然と配置されたポップアップメニュー群を実現したり、非常に長いメニュー項目を含むメニューコントロールの幅を制限したりできます。
bodywidth キーワードは、テキスト以外のポップアップメニューにも影響します。

font と fsize キーワードは、ポップアップメニューのタイトルにのみ影響します。
ポップアップメニュー本体は標準のシステムフォントを使います。

色、カラーテーブル、線種、パターン、マーカーのポップアップメニューとは異なり、テキストのポップアップメニューコントロールは、モードキーワードの値によって設定される２つの異なるモードで動作できます。

mode キーワードの引数がゼロ以外の場合、それは初期の現在の項目であるメニュー項目の番号と見なされ、ポップアップメニューボックスに現在の項目を表示します。

これが selector mode です。
現在の項目の値は ControlInfo コマンドを使っていつでも読み取れるため、アクションプロシージャは通常不要です。

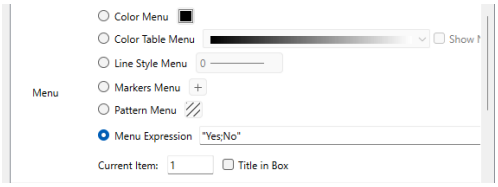
モードがゼロの場合、タイトルはポップアップメニューボックス内に表示されるため、このモードは「タイトルインボックスモード」と呼ばれます。
このモードは通常、アクションプロシージャが実行するコマンドを選択するために使われます。
現在の項目は、ポップアップメニューがアクティブ化され、選択された項目がアクションプロシージャに渡される場合を除き、意味を持ちません。

コントロールがクリックされた時に表示されるポップアップメニューは、value キーワードの引数として渡す文字列式によって決定されます。
例えば：

```
PopupMenu name value="Item 1;Item 2;Item 3;"
```

色、カラーテーブル、線種、パターン、またはマーカーのポップアップメニューを作成するには、文字列式を次の固定値のいずれかに設定します。

```
"*COLORPOP*"
"*COLORTABLEPOP*"
"*COLORTABLEPOPNONAMES*"
"*LINESTYLEPOP*"
"*MARKERPOP*"
"*PATTERNPOP*"
```



テキストポップアップメニューの場合、文字列式はセミコロンで区切られた項目のリストとして評価されなければなりません。
これは固定リテラル文字列または動的に計算された文字列です。
例えば：

```
PopupMenu name value="Item 1;Item 2;Item 3;"
PopupMenu name value="¥_none¥_;" + WaveList("*",";","")
```

メニュー項目には、特定の特殊効果を適用することが可能です。
例えば、項目を無効化したり、項目にチェックマークを付けるといった操作です。
詳細はヘルプ Special Characters in Menu Item Strings を参照してください。

文字列式の評価結果ではなく、文字列式のリテラルテキストがコントロールとともに保存されます。
Igor は、PopupMenu value=<value> コマンドの実行時に式を評価し、ユーザーがポップアップメニューボックスをクリックするたびに式を再評価します。
この再評価により、上記の WaveList の例で作成されたような動的メニューは、ポップアップメニューコントロールが作成された時点の状態ではなく、クリック時の状態を反映します。

ユーザーがクリックし、Igor が文字列式を再評価する時点で、ポップアップメニューを作成したプロシージャは既に実行終了しています。
その結果、そのローカル変数は存在しなくなるため、文字列式はそれらを参照できなくなります。
値式にローカル変数の値を組み込むには、Execute コマンドを使います。

```
String str = <code that generates item list>   // str はローカル変数
Execute "PopupMenu name value=" + str
```

Igor は文字列式をコマンドラインで入力されたかのように評価します。
ユーザーがポップアップメニューをクリックした時点で、現在のデータフォルダーがどうなるかは予測できません。
したがって、特定のデータフォルダー内のオブジェクトを参照したい場合は、絶対パスを使う必要があります。
例えば：

```
PopupMenu name value=#"func(root:DF234:wave0, root:gVar)"
```

クリック時の再評価のため、文字列式値が変更されてもポップアップメニューは自動更新されません。
通常これは問題になりませんが、ControlUpdate コマンドを使ってポップアップメニューの更新を強制できます。
次に例を挙げます。

```
NewPanel/N=PanelX
String/G gPopupList="First;Second;Third"
PopupMenu oneOfThree value=gPopupList          // ポップアップは "First" を表示
gPopupList="1;2;3"                             // ポップアップは変わらない
ControlUpdate/W=PanelX oneOfThree              // ポップアップは "1" を表示
```

場合によっては、文字列式が PopupMenu コマンドのコンパイル時にコンパイルできないことがあります。
これは、まだ存在しないグローバルオブジェクトを参照しているためです。
この場合、次の特別な構文を使うことでコンパイル時のエラーを回避できます。

```
PopupMenu name value= #"pathToNonExistentGlobalString"
```

遅延式に引用符が含まれる場合、それらをバックスラッシュでエスケープする必要があります。

```
PopupMenu name value= #"¥"¥_none¥_;¥"+UserFunc(¥"foo¥")"
```

オプションのユーザー定義アクションプロシージャは、ユーザーがポップアップメニューから選択を行った後に呼び出されます。
ポップアップメニュープロシージャは次の形式を持ちます。

```
Function PopMenuProc(pa) : PopupMenuControl
        STRUCT WMPopupAction &pa

        switch (pa.eventCode)
                case 2:                                         // マウス UP
                        Variable popNum = pa.popNum             // 1-ベースのアイテム番号
                        String popStr = pa.popStr               // 選択したアイテムのテキスト
                        break
                case -1:                                        // コントロールを Kill
                        break
        endswitch

        return 0
End
```

pa.popNum はアイテム番号（1 から始まる）であり、*pa.popStr* は選択されたアイテムのテキストです。

カラーポップアップメニューにおいて、選択された色を判定する最も簡単な方法は、ControlInfo コマンドを使うことです。

メニュー項目を指定する様々な方法の例については、PopupMenu コマンドのヘルプを参照してください。

変数の設定コントロールの作成

SetVariable コマンドは、SetVariable コントロールを作成または変更します。
SetVariable コントロールは、これらの値の表示と設定の両方に役立ちます。

SetVariable コントロールは、数値または文字列のグローバル変数、ウェーブの 1 つの要素、またはコントロール自体に保存された内部値に紐付けられます。
煩雑さを最小限に抑えるため、ほとんどの場合、内部値を使うべきです。

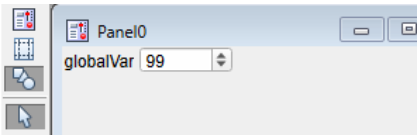
数値変数で使う場合、Igor は値を増減させるためにユーザーが使用できる上下矢印を描画します。

コントロールの幅は設定できますが、高さはフォントとフォントサイズによって決定されます。
表示領域の幅は、コントロールの幅からタイトルと上下矢印の幅を差し引いた値です。
ただし、bodyWidth キーワードを使って、コントロールの本体（タイトル以外の部分）に固定幅を指定できます。

例えば、次のコマンドを実行すると：

```
Variable/G globalVar=99
SetVariable setvar0 size={120,20},frame=1,font="Helvetica", value=globalVar
```

右の SetVariable コントロールを作成します。



SetVariable コントロールを、SetVariable が実行される時点で現在のデータフォルダーに存在しない変数に関連付けるには、データフォルダーのパスを使う必要があります。

```
Variable/G root:Packages:ImagePack:globalVar=99
SetVariable setvar0 value=root:Packages:ImagePack:globalVar
```

ポップアップメニューコントロールとは異なり、SetVariable コントロールは SetVariable コマンドの実行時に現在のデータフォルダーを記憶します。
したがって、同等のコマンドのセットは次のとおりです。

```
SetDataFolder root:Packages:ImagePack
Variable/G globalVar=99
SetVariable setvar0 value=globalVar
```

「SetVariable コントロールとデータフォルダー」のセクションも参照してください。

数値表示のスタイルは、format キーワードでコントロールできます。
例えば、文字列「%.2d」は小数点以下 2 桁で値を表示します。
数値を読み戻す必要があるため、表示にテキストを含めるためにフォーマット文字列を使わないでください。
表示値に接尾辞を追加することは可能ですが、接頭辞は機能しません。
文字列変数と併用する場合、フォーマット文字列は使われません。

多くの場合、ControlInfo を使って値を照会するだけで十分であり、アクションプロシージャは必要ありません。
値が変更されるたびに何か処理を行いたい場合は、以下の形式でアクションプロシージャを作成する必要があります。

```
Function SetVarProc(sva) : SetVariableControl
        STRUCT WMSetVariableAction &sva

        switch (sva.eventCode)
                case 1:                                         // マウス UP
                case 2:                                         // キー入力
                case 3:                                         // Live Update
                        Variable dval = sva.dval
                        String sval = sva.sval
                        break
                case -1:                                        // コントロールを Kill
                        break
        endswitch

        return 0
End
```

値が文字列の場合、sva.sval はその内容を含みます。
数値の場合、sva.dval がその値を含みます。
sva.isStr は数値に対して 0 を返し、文字列に対しては 0 以外を返します。

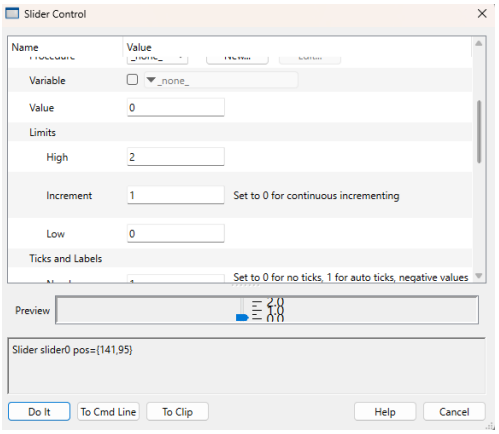
ユーザーが上向きまたは下向きの矢印をクリックして押し続けると、変数の値は増分値によって徐々に変化します。
デフォルトでは、ユーザーがマウスボタンを離すまでアクションプロシージャは呼び出されません。

SetVariable キーワード live=1 を使うと、増分ごとにアクションプロシージャが呼び出されます。
この機能を使う場合は、アクションプロシージャが迅速に実行されることが重要です。

WMSetVariableAction 構造体の情報を使うことで、マウスクリックがコントロールのどの位置で発生したかを把握できます。
これにより、非線形な増分やコンテキストメニューの表示といった特殊効果を作成することが可能になります。

スライダーコントロールの作成

Slider コマンドは、スライダーコントロールを作成または変更します。

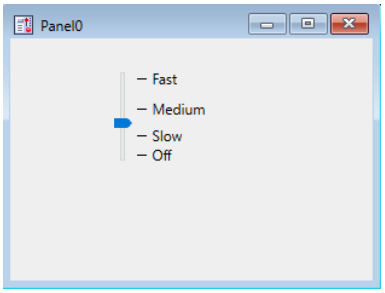


スライダーコントロールは、数値のグローバル変数、またはコントロール自体に格納された内部の数値に紐付けられます。
煩雑さを最小限に抑えるため、ほとんどの場合、内部値を使うべきです。
値は、コントロールのスライダーをドラッグするか、目盛上で目的の値をクリックすることで変更されます。

数値範囲のラベル付けには、目盛の数を設定するなど、多くの選択肢があります。

カスタムラベルは２つのウェーブで指定できます。
１つ目は数値、２つ目は対応するテキストラベルです。

例えば：




```
NewPanel
Make/O tickNumbers= {0,25,60,100}
Make/O/T tickLabels= {"Off","Slow","Medium","Fast"}
Slider speed,pos={86,28},size={74,73}
Slider speed,limits={0,100,0},value= 40
Slider speed,userTicks={tickNumbers,tickLabels}
```

多くの場合、ControlInfo を使って値を照会するだけで十分であり、アクションプロシージャは必要ありません。
値が変更されるたびに何かを実行したい場合、または繰り返しスライダーを実装したい場合は、アクションプロシージャを作成する必要があります。

ユーザーがスライダーをドラッグしたとき、スライダーをクリックしたとき、スライダーの両側をクリックしたとき、およびプロシージャがスライダーのグローバル変数（存在する場合）を変更したときに、Igor はアクションプロシージャを呼び出します。
繰り返しスライダーの場合、ユーザーがスライダーをクリックしている間、定期的にアクションプロシージャを呼び出します。

詳細については、Slider コマンドのヘルプ、「繰り返しスライダー」のセクションを参照してください。

スライダーイベントの処理

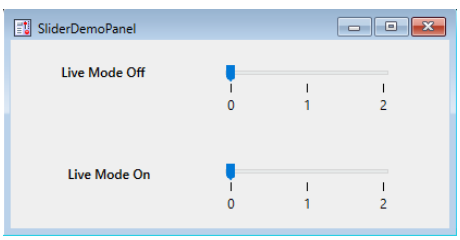
スライダーは、マウスが離された時（ライブモードオフ）にのみアクションプロシージャを呼び出すか、マウスが押されている間スライダーの位置が変化するたびに呼び出す（ライブモードオン）ことができます。
このセクションでは、両方の種類のスライダーを作成する方法を示します。

下のコードを新しいエクスペリメントのプロシージャウィンドウに入力してください。

```
Window SliderDemoPanel() : Panel
	PauseUpdate; Silent 1			// ウィンドウを構築
	NewPanel /W=(262,115,665,287)

	TitleBox Title0,pos={46,21},size={139,15},fSize=12,frame=0,fStyle=1
	TitleBox Title0,title="Live Mode Off"
	Slider slider0,pos={197,23},size={150,44},proc=Slider0Proc
	Slider slider0,limits={0,2,0},value=0,live=0,vert=0

	TitleBox Title1,pos={52,114},size={135,15},fSize=12,frame=0,fStyle=1
	TitleBox Title1,title="Live Mode On"
	Slider slider1,pos={197,113},size={150,44},proc=Slider1Proc
	Slider slider1,limits={0,2,0},value=0,live=0,vert=0
EndMacro

Function Slider0Proc(sa) : SliderControl		// slider0 のアクションプロシージャ
	STRUCT WMSliderAction &sa

	switch(sa.eventCode)
		case -3:				// コントロールがキーボードフォーカスを受けとった
		case -2:				// コントロールがキーボードフォーカスを失った
		case -1:				// コントロールを Kill
			break
		default:
			if (sa.eventCode & 1)	// 値のセット
				Printf "Slider value = %g, event code = %d¥r", sa.curval, sa.eventCode
			endif
			break
	endswitch

	return 0
End

Function Slider1Proc(sa) : SliderControl		// slider1 のアクションプロシージャ
	STRUCT WMSliderAction &sa

	switch(sa.eventCode)
		case -3:				// コントロールがキーボードフォーカスを受けとった
		case -2:				// コントロールがキーボードフォーカスを失った
		case -1:				// コントロールを Kill
			break
		default:
			if (sa.eventCode & 1)	// 値のセット
				Printf "Slider value = %g, event code = %d¥r", sa.curval, sa.eventCode
			endif
			if (sa.eventCode & 8)	// マウスが移動、または矢印キーがスライダーを移動
				Printf "Mouse moved or arrow key, value = %g, event code = %d¥r",
sa.curval, sa.eventCode
			endif
			break
	endswitch

	return 0
End
```

その後、コマンドラインで次を実行してください。

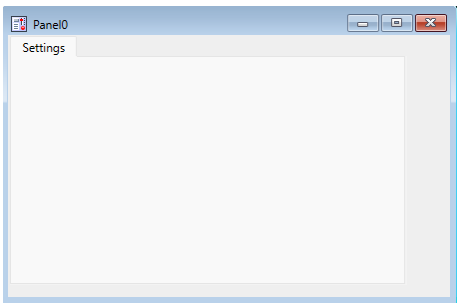
```
SliderDemoPanel()
```



タブコントロールの作成

TabControl コマンドは、タブコントロールの作成または変更を行います。
タブは、コントロールを可視グループと非表示グループにグループ化するために使われ、異なる目的のコントロール群がコントロールパネル内の同じ領域を占めることを可能にします。

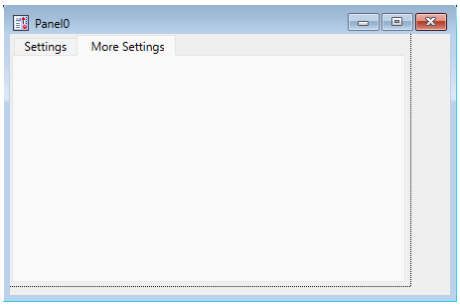
タブには番号が付けられています。
最初のタブは tab 0、2 番目は tab 1、以下同様です。



デフォルトのタブコントロールには 1 つのタブがあります。

```
NewPanel /W=(150,50,650,400)
TabControl tb, tabLabel(0)="Settings", size={400,250}
```

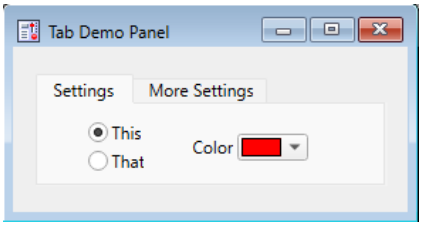
コントロールにタブを追加するには、追加のタブラベルを指定します。

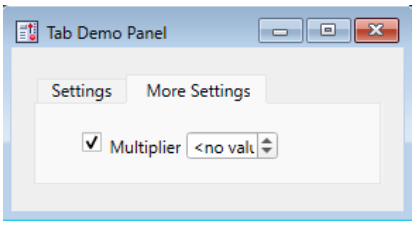



```
TabControl tb, tabLabel(1)="More Settings"
```

タブをクリックすると、コントロールのアクションプロシージャがクリックされたタブの番号を受け取ります。

コントロールの表示と非表示は、アクションプロシージャによって行われます。
次の例では、Setting タブがクリックされると、[This]、[That]、[Color] コントロールが表示され、[Multiplier] チェックボックスは非表示になります。
[More Settings] タブがクリックされると、アクションプロシージャによって逆の動作が行われます。

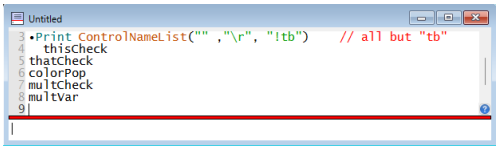




タブ付きインターフェイスを作成する最も簡単な方法は、タブコントロールの外側にすべてのコントロールが見えるように配置した、大きめのパネルを作成することです。
コントロールを互いに相対的なおおよその位置に配置します。
このようにコントロールを配置することで、満足のいく状態になるまで各コントロールをより簡単に修正できます。

タブコントロールにコントロールを配置する前に、タブコントロール以外のコントロール名のリストを取得します。

```
Print ControlNameList("" ,"¥r", "!tb") // "tb"以外のすべて

thisCheck
thatCheck
colorPop
multCheck
multVar
```



どのコントロールをどのタブに表示するかを決定します。

Tab 0: Settings	**Tab 1: More Settings**
thisCheck	multCheck
thatCheck	multVar
colorPop	

タブコントロールの表示と非表示を行うプロシージャを記述します。

```
Function TabProc(tca) : TabControl
        STRUCT WMTabControlAction &tca

        switch (tca.eventCode)
                case 2:                                                 // マウス UP
                        Variable tabNum = tca.tab                       // アクティブなタブ番号
                        Variable isTab0 = tabNum==0
                        Variable isTab1 = tabNum==1

                        ModifyControl thisCheck disable=!isTab0         // Tab 0 ではない時は隠す

                        ModifyControl thatCheck disable=!isTab0         // Tab 0 ではない時は隠す
                        ModifyControl colorPop disable=!isTab0          // Tab 0 ではない時は隠す

                        ModifyControl multCheck disable=!isTab1         // Tab 1 ではない時は隠す
                        ModifyControl multVar disable=!isTab1           // Tab 1 ではない時は隠す
                        break
        endswitch

        return 0
End
```

より洗練された方法は、各タブ内のコントロールを、そのタブ固有の接頭辞または接尾辞を用いて体系的に命名することです。
例えば、tab0_thisCheck、tab0_thatCheck、tab1_multVar などです。
その後、ModifyControlList コマンドを使ってコントロールの表示と非表示を切り替えます。
例については ModifyControlList コマンドのヘルプを参照してください。

アクションプロシージャをタブコントロールに割り当てます。

```
TabControl tb, proc=TabProc
```

アクションプロシージャが、タブをクリックするたびにコントロールを正しく表示および非表示にすることを確認してください。
これが正しく動作したら、コントロールをタブコントロール内の最終的な位置に移動します。

このプロセスでは、「一時選択」ショートカットが便利です。
操作モード中に Ctrl+Alt を押すと、一時的に選択モードに切り替わり、コントロールを選択してドラッグできるようになります。

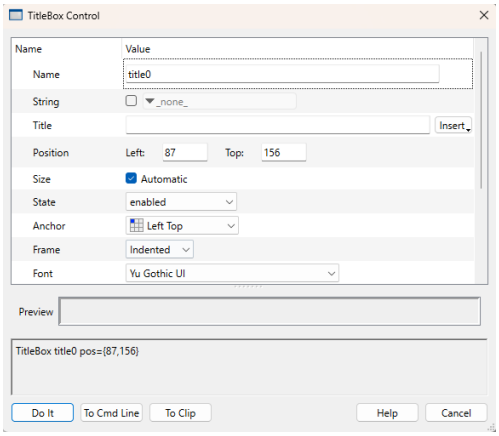
パネルを再作成マクロとして保存し（Windows→Control→Window Control）、最終的なコントロール位置を記録します。
マクロを、最初にパネルを作成する関数として書き換えます。

```
Function CreatePanel()
        KillWindow/Z TabPanel
        NewPanel/N=TabPanel/W=(596,59,874,175) as "Tab Demo Panel"
        TabControl tb,pos={15,19},size={250,80},proc=TabProc
        TabControl tb,tabLabel(0)="Settings"
        TabControl tb,tabLabel(1)="More Settings",value= 0
        CheckBox thisCheck,pos={53,52},size={39,14},title="This"
        CheckBox thisCheck,value= 1,mode=1
        CheckBox thatCheck,pos={53,72},size={39,14},title="That"
        CheckBox thatCheck,value= 0,mode=1
        PopupMenu colorPop,pos={126,60},size={82,20},title="Color"
        PopupMenu colorPop,mode=1,popColor= (65535,0,0)
        PopupMenu colorPop,value= #"¥"*COLORPOP*¥""
        CheckBox multCheck,pos={50,60},size={16,14},disable=1
        CheckBox multCheck,title="",value= 1
        SetVariable multVar,pos={69,60},size={120,15},disable=1
```

```
        SetVariable multVar,title="Multiplier",value=multiplier
End
```

タブコントロール内のコントロールを管理する別の方法として、各タブに対してパネルサブウィンドウを作成する方法があります。
アクションプロシージャでは、SetWindow コマンドで hide キーワードを使って、選択されたタブのサブウィンドウを表示し、他のサブウィンドウを非表示にします。
コントロールのレイアウト時にサブウィンドウの表示／非表示を切り替えることができます。
この方法の例は、「ガイドモードでのコントロールのレイアウト」のセクションで後述します。

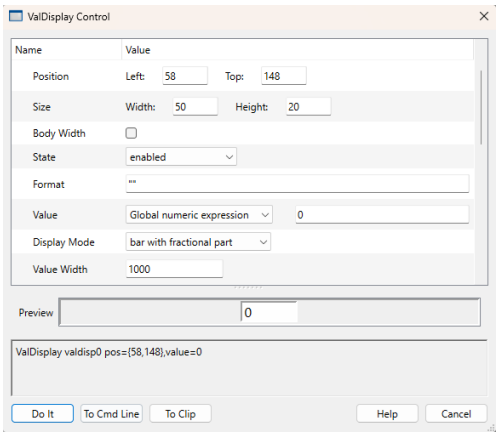
タイトルボックスコントロールの作成



TitleBox コマンドは、タイトルボックスコントロールを作成または変更します。
コントロールのテキストは静的でも、グローバル文字列変数に紐付けることもできます。
グローバル文字列変数を使うと、より長い文字列が可能になります。
静的テキストは一行のコマンド行内に収まる必要があるためです。
完全な説明と例については、TitleBox コマンドのヘルプを参照してください。

デフォルトでは、タイトルボックスコントロールはテキストに合わせて幅を調整します。
固定サイズを設定するには fixedSize キーワードを使います。
これにより、動的で長くなる可能性のあるコンテンツが他のコントロールに干渉するのを防げます。

値表示コントロールの作成



ValDisplay コマンドは、値表示コントロールを作成または変更します。

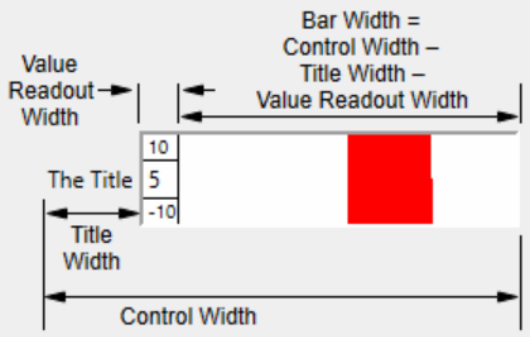
値表示は非常に柔軟で多面的なコントロールです。
単純な数値表示から温度計バー、あるいはその両方のハイブリッドまで、さまざまな形をとることができます。
値表示コントロールは、value キーワードの引数として指定する数値式に関連付けられます。
数値式が依存する何かが変更されるたびに、Igor は自動的にコントロールを更新します。

値表示コントロールは、root データフォルダーのコンテキストで値式を評価します。
root にないデータオブジェクトを参照するには、データフォルダーパス（例: root:Folder1:var1）を使う必要があります。

ValDisplay コマンドから抜粋したいくつかの選択キーワードは以下の通りです。

```
size={width,height}
barmisc={lts, valwidth}
limits={low,high,base}
```

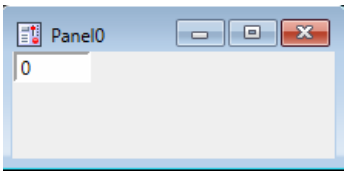
値表示コントロールのサイズと外観は、主に valwidth および size パラメーターとタイトルの幅によって決まります。
ただし、bodyWidth キーワードを使って、コントロールの本体（タイトル以外の部分）の固定幅を指定することができます。
基本的に、各要素のスペースは左から右に割り当てられ、タイトルが最優先されます。
タイトルでコントロール幅がすべて使われていない場合、値表示幅は valwidth ポイントと残りの幅のうち小さい方になります。
コントロール幅が使い切られていない場合、バーは残りのコントロール幅内に表示されます。



値表示コントロールには多くの形態があります。
以下にその一部を示します。
これらの例の中には、その前の例を修正したものもあります。
例えば、２番目のバーのみの例は、最初のバーのみの例で作成された valdisp1 コントロールを修正したものです。
これらの例を試すには、それらを配置するためのパネルが必要です。

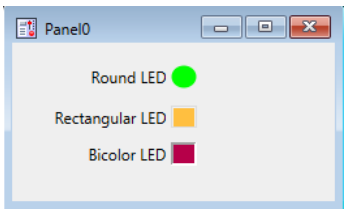
数字の表示のみ

```
// デフォルトの表示幅（1000）は、デフォルトのコントロール幅（50）以上
ValDisplay valdisp0 value=K0
```



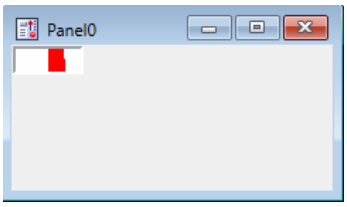
LED 表示

```
// ３つの LED 形式を作成
ValDisplay led1,pos={67,17},size={75,20},title="Round LED"
ValDisplay led1,limits={-50,100,0},barmisc={0,0},mode=1
ValDisplay led1,bodyWidth= 20,value= #"K1",zeroColor=(0,65535,0)

ValDisplay led2,pos={38,48},size={104,20},title="Rectangular LED"
ValDisplay led2,frame=5,limits={0,100,0},barmisc={0,0},mode=2
ValDisplay led2,bodyWidth= 20,value= #"K2"
ValDisplay led2,zeroColor= (65535,49157,16385)

ValDisplay led3,pos={60,76},size={82,20},title="Bicolor LED"
ValDisplay led3,limits={-40,100,-100},barmisc={0,0},mode= 2
ValDisplay led3,bodyWidth= 20,value= #"K3"
```



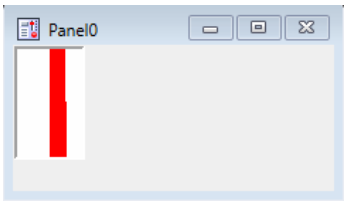
バーのみ

```
// 表示幅 = 0
ValDisplay valdisp1,frame=1,barmisc={12,0},limits={-10,10,0},value=K0
K0= 5            // 0 の基底から 10 の上限までの半分の位置
```



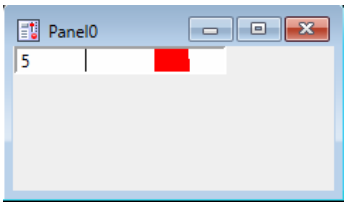
バーのみの ValDisplay の良い点は、高さを 5～200 ポイントに設定できることです。
一方、数値表示では、表示部と印刷された目盛りのフォントサイズによって高さが決まります。

```
// コントロール高さ = 80 に設定
ValDisplay valdisp1, size={50,80}
```



数値表示とバー

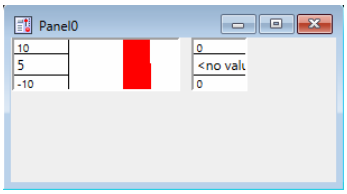
```
// 0 < 表示幅 (50) < コントロール幅 (150)
ValDisplay valdisp2 size={150,20},frame=1,limits={-10,10,0}
ValDisplay valdisp2 barmisc={0,50},value=K0    // 制限なし表示
```



オプションの制限

数値表示が可視状態のときは、オプションの設定限界値も表示されることがあります。

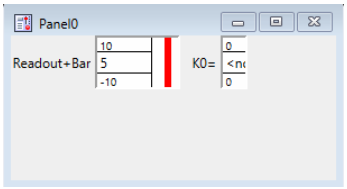
```
// フォントサイズを 10 ポイントに設定。表示幅は変更しない
ValDisplay valdisp2 barmisc={10,50}
ValDisplay valdisp0 barmisc={10,1000}
```



オプションのタイトル

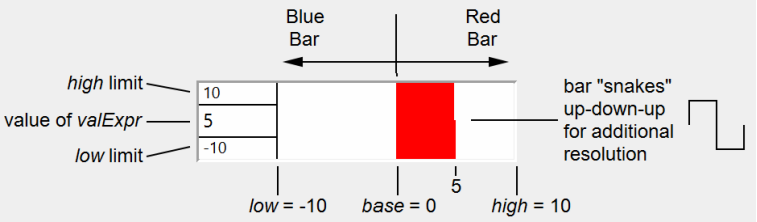
コントロールのタイトルが数値表示とバーから横方向のスペースを奪い、それらを右側に押しやっています。
表示が消えないようにするには、コントロールの幅を大きくする必要があるかもしれません。

```
// タイトルを追加。表示幅、コントロール幅は変更なし
ValDisplay valdisp2 title="Readout+Bar"
ValDisplay valdisp0 title="K0="
```



限界値 low、high、base、valExpr の値は、バー（存在する場合）の描画方法をコントロールします。
バーは、base 値に対応する開始位置から、valExpr、low、high の値によって決定される終了位置まで描画されます。
low はバーの左側に対応し、high は右側に対応します。
base 値に対応する位置は、low と high の間で直線補間されます。

例えば、low = -10、high = 10、base = 0 の場合、valExpr の値が 5 であれば、バー領域の中心（0 は -10 と 10 の間に中心位置）から右方向へ、バー領域の中心から右端までの半分の位置（5 は 0 から 10 までの半分の位置）まで描画されます。

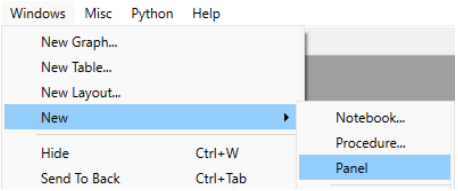


mode=3 を指定することで、小数部を含む棒の描画を強制的に無効にできます。

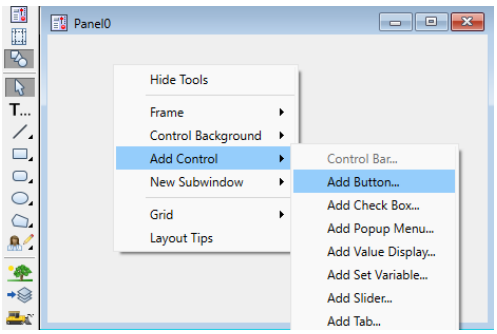
ガイドモードでのコントロールのレイアウト

サブウィンドウとサブウィンドウ配置用のガイドは、Igor で長年利用可能でした。
Igor Pro バージョン 10 では、ガイドがコントロール配置にも拡張されました。
これにより、ガイドの動作方式にいくつかの変更が加えられています。
変更点について、およびガイドモードでのサブウィンドウ配置について学ぶには、ヘルプ Guides Mode for Laying Out Subwindows を参照してください。
レイアウトガイドの基本を学ぶために、今すぐこのヘルプを参照することをお勧めします。

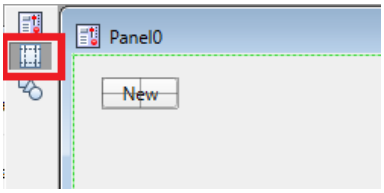
新しいパネルを作成するには、Windows→New→Panel を選択します。
デフォルトでは、新しいパネルにはツールパレットが表示されており、すぐに作業を開始できます。
ツールパレットが表示されていない場合は、Panel→Show Tools を選択します。



パネルを右クリックし、Add Control→Button を選択します。
Do It をクリックします。
右クリックした位置に「New」というタイトルのボタンが作成されることに注目してください。



ツールパレットで、2 番目のボタンをクリックしてガイドモードに入ります。
ボタンをクリックして選択します。
ガイドモードで選択すると、Igor はコントロールの端と中央に灰色の線を追加します。

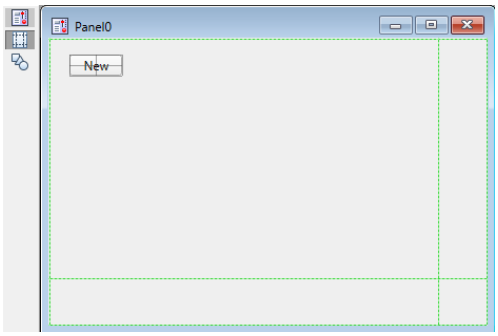


これらの線は、コントロールを所有するウィンドウ内のレイアウトガイドにコントロールを固定するためのアンカーです。
コントロールパネルのウィンドウサイズが変更された時、そのボタンをコントロールパネルの右下隅付近に保持したい場合があります。

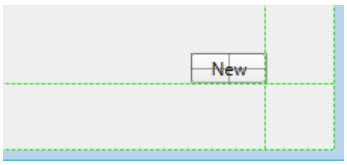
ウィンドウの縁の近くにコントロールを配置

Alt キーを押したまま、（少し見えにくいですが）ウィンドウの右端にある緑色の点線の上にマウスを移動させ、マウスのカーソルが二重線の左右矢印に変わるまで待ちます。

Alt キーを押したまま、左方向に約 1cm ドラッグします。
新しいユーザー定義ガイドが作成され、マウスでドラッグされます。
同じ操作で、ウィンドウの下端から新しいガイドを上にドラッグします。
完了すると、ウィンドウは右のような状態になります。



ボタンをクリックして選択します。
ウィンドウがガイドモードのため、ボタンのアンカー線が表示されます。
マウスポインターを右端のアンカー線に合わせ、左右矢印カーソルに変わっていることを確認してください。
スペースが限られているため注意が必要です。
正しいカーソルが表示されたら、ボタンの右端アンカーラインをクリックし、垂直ガイドまでドラッグします。
ガイドに近づくとスナップします。
次にマウスをボタンの下端に移動し、上下矢印カーソルを探します。
ボタンをクリックして水平ガイドまでドラッグします。



ガイドモードでコントロールを選択すると、そのコントロールにガイドへのアタッチメントがある場合、アタッチされたアンカーラインが赤色で表示されます。

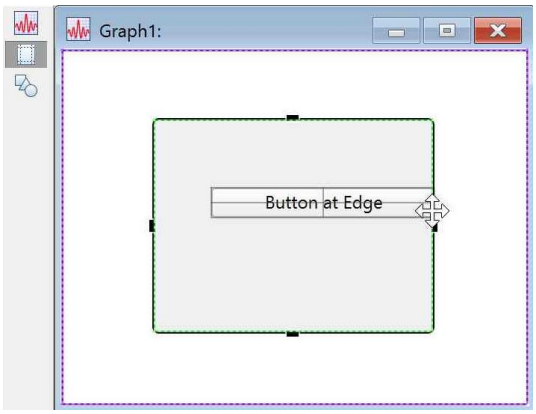
ウィンドウ枠の右下隅をドラッグしてウィンドウのサイズを変更します。
ボタンがガイドに沿って移動し、ウィンドウの端から一定の距離を保つ様子を確認してください。

コントロールをアタッチされたガイドから切り離すには、アンカーライン上にないコントロール本体内の任意の場所をクリックしてドラッグします。
マウスカーソルの状態から動作がわかります。
両矢印カーソルの場合は接続線をドラッグします。
アタッチされたガイドをドラッグすると、そのガイドが切り離されます。

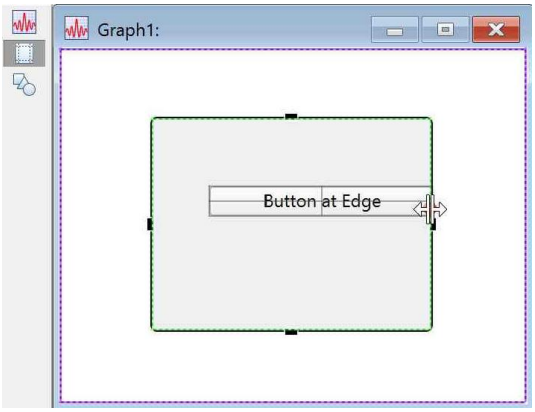
サブウィンドウの端にあるコントロールアンカーを変更できない場合

右は、サブウィンドウ内のコントロールパネルのサブウィンドウで、ボタンの右端アンカーがサブウィンドウ内のフレーム右ガイドにアタッチされている状態を示す画像です。

マウスがボタンの右端アンカーの上にホバーしているのは、そのアンカーを移動させたいためです。
しかし実際には、マウスカーソルはボタンのアンカーを選択して移動するのではなく、サブウィンドウのサイズ変更が行われることを示しています。



その位置にカーソルを合わせた状態で Alt キーを押すと、代わりに左右の矢印が付いた二重線カーソルが表示され、ボタンの右端アンカーを移動できる状態であることを示します。

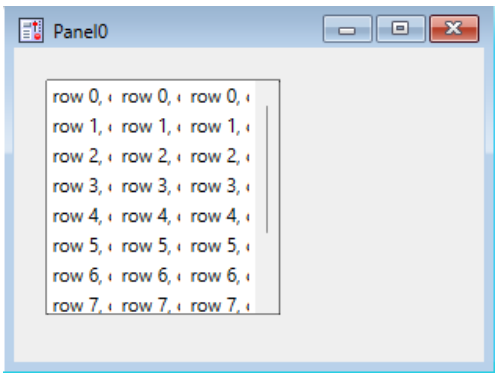


リストボックスを含んでいるパネルでサイズ変更を可能にする

リストボックスコントロールを含むパネルを作成するには、次のコマンドを実行します。

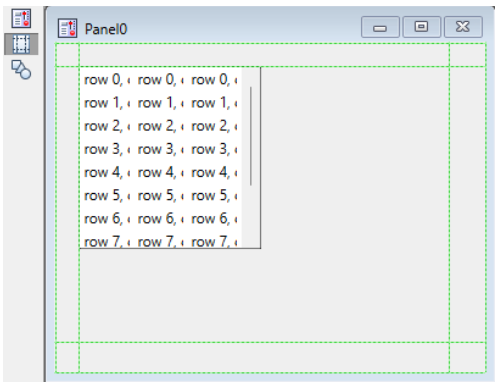
```
NewPanel
Make/T/N=(10,3)/O listwave="row "+num2str(p)+", col "+num2str(q)
Make/N=(10,3)/O selwave=0
ListBox list0 pos={20,20},size={150,150},ListWave=listwave,SelWave=selwave,mode=8
```

これは次のような表示になります。



このリストボックスは、パネルウィンドウのサイズ変更時にパネル全体に広がるようにすると便利です。
そのためには、まさにその動作を実現する４つのユーザー定義ガイドを作成します。

前のセクションと同じ方法を使って、ウィンドウの四辺すべて近くにガイドを作成します：

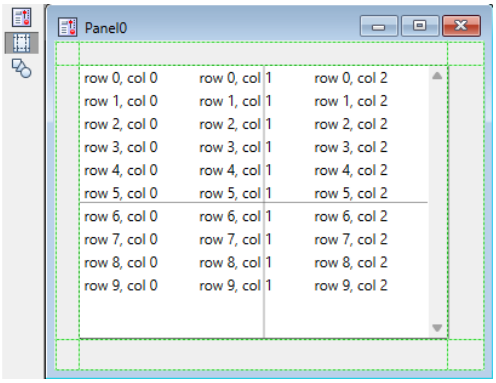


リストボックスコントロールをクリックして選択し、アンカー線を表示します。
上部のアンカー線をドラッグして、上部のガイドにスナップするまで移動します。
特定の方向にガイドにアンカーが既にアタッチされていない場合、アンカーをドラッグするとコントロールのサイズを変更せずに移動することに注意してください。

左アンカーを左ガイドにドラッグするなどして、４辺すべてが適切なガイドにアタッチされるまで操作を続けます。

特定の方向に１つのアンカーが固定されると、その方向で別のアンカーをドラッグするとコントロールのサイズが変更されます。
最初に固定されたアンカーがコントロールの位置を制限するため、アンカーのドラッグに対応する唯一の方法はサイズ変更です。

最終の結果です。



パネルのサイズを変更すると、リストボックスコントロールはウィンドウに合わせて拡大・縮小します。

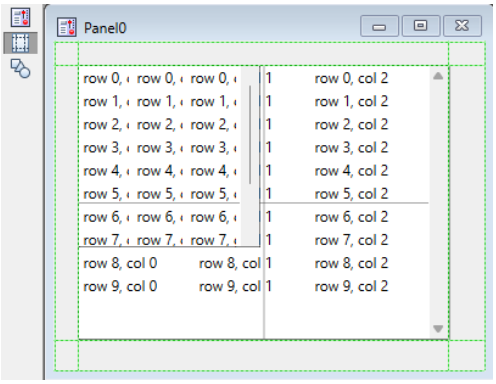
この方法は、タブコントロールを持っているウィンドウと共にサイズ変更させる場合にも役立ちます。

２つのリストボックスコントロールがウィンドウを均等に共有

前のセクションのターゲットと同じパネルウィンドウで、このコマンドを実行して２つ目のリストボックスコントロールを作成します。

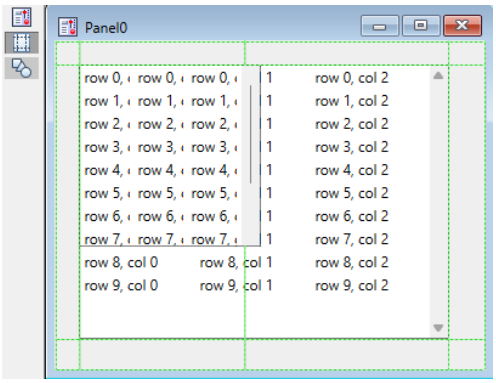
```
ListBox list1 pos={20,20},size={150,150},listWave=listwave,selwave=selwave,mode=8
```

実際のアプリケーションでは、２つのリストボックスで同じウェーブを再利用することは非常に珍しいです。
これは混乱を招くためです。



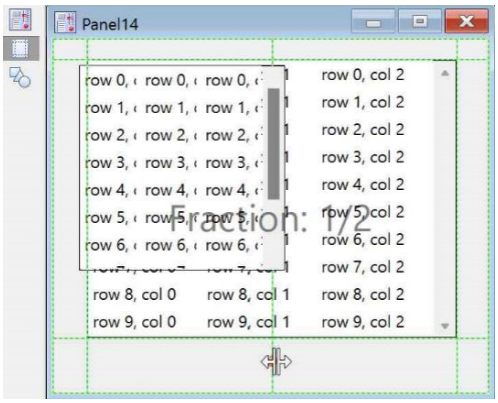
２つのリストボックスコントロールを並べて配置し、等しいスペースを占めるようにします。
また、ウィンドウのサイズ変更時に等しいスペースを維持するようサイズ変更します。
そのためには、他の２つのガイドを参照する相対ガイドを作成します。

Alt キーを押したまま、左側のガイドから新しいユーザー定義ガイドをクリックしてドラッグします。
既存のガイドから Alt キーを押しながらドラッグすると、新しいガイドを作成できます。



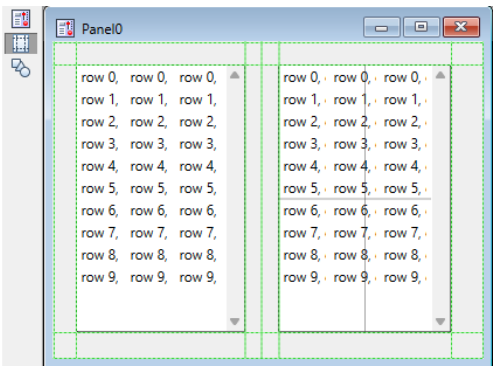
新しいガイドを右クリックし、Make relative to→UGV2 を選択します（ガイド上にマウスカーソルを置くと、ガイド名、所有ウィンドウ名、ガイドに関する情報が表示されるツールチップが表示されます）。
ガイドの作成順序によっては、このガイドが UGV0/1 のような別のものとなる場合があります。
相対ガイドは、基準となる 2 つのガイド間の距離を比例的に維持します。

そのガイドを UGV0（左のガイド）と UGV1（右のガイド）のちょうど中間点に配置したいと考えています。
今回は Alt キーを押しながらドラッグしないでください。
代わりに Shift キーを押しながらドラッグしてください。
これによりガイドの位置が特定の分数に制限されます。

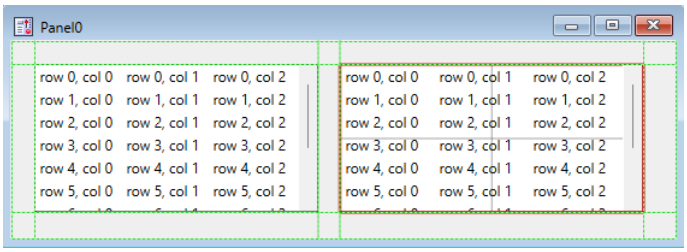


右側のリストボックスの左端と新しいリストボックスの右端をそのガイドに固定できます。
新しいリストボックスは他のガイドにもアタッチする必要があります。
これでウィンドウのサイズ変更時にもリストボックス間の関係が維持されます。

しかし実際には、リストボックスの間に少しスペースが必要です。
中央のガイドから新しいガイドを少し左に、もう一つを少し右にドラッグします。
最終的な見た目は次のようになります。



そしてウィンドウを横方向に引き伸ばし、縦方向に縮小すると、次のようになります。

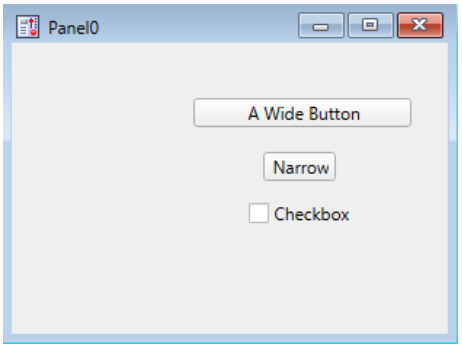


中央揃えのコントロールの列

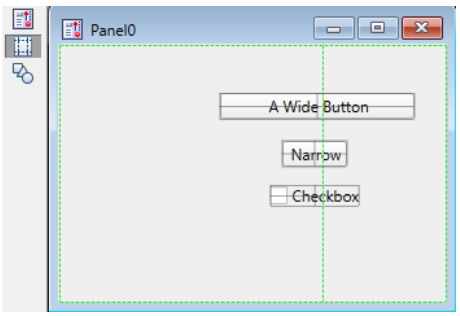
例えば、これまでの例では、コントロール上に十字を作る中央ガイドアンカーは使っていません。
これらのアンカーは、コントロールの中心に垂直または水平に取り付けられます。
これらのアンカーを使うと、ウィンドウ内の他のコントロールや機能に対して、コントロールを中央に配置したままにできます。

細いボタン、広いボタン、チェックボックスで構成されるパネルを作成します。

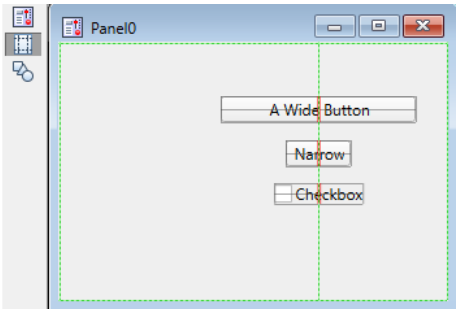
```
NewPanel
Button button0,pos={124.50,38},size={150,20},title="A Wide
Button"
Button button1,pos={172.50,74.50},size={50,20},title="Narrow"
CheckBox
s,pos={163.75,109},size={67.50,16.50},title="Checkbox",value=0
```



次に、パネルの右側から新しいガイドを作成します。
３つのコントロールすべてをドラッグ選択します。
パネルは右のようになります。



コントロールを作成したコマンドを操作し、各コントロールの水平方向の中心がほぼ揃うように調整しました。
いずれかのコントロールの垂直方向のアンカーを、ガイドにスナップするまでドラッグしてください。
複数のコントロールを選択した状態で１つのコントロールのアンカーをドラッグすると、すべてのコントロールが移動します。
ドラッグしたアンカーが他のコントロールの同じアンカーとほぼ揃っている場合、すべてが同時にガイドにスナップします。
最終の結果は右のようになります。



ウィンドウのサイズを変更しても、それらのコントロールはウィンドウの右端付近に配置されたままとなり、中央揃えの列に配置された状態を維持します。

コントロールを過剰に制約することはできない

ガイドアタッチメントがコントロールに適用する制約に対応する方法は２つしかありません。
１つのガイドアンカーがアタッチされた場合に発生するコントロールの移動、または２つのアンカーがアタッチされた場合に発生するコントロールのサイズ変更です。
しかし各方向には３つのアンカーが存在します。
３つ全てがアタッチされた場合はどうなるのでしょうか？
Igor ではそのような操作は許可されません。

一部のコントロールのサイズは、フォントサイズやタイトルの長さなどの要素によって設定されます。
このようなコントロールでは、２つのガイドアタッチメントがサイズ変更を試みることになり、コントロールの推奨サイズと競合します。
このようなコントロールでは、１つのアタッチメントのみが許可されます。
一部のコントロールには、サイズ変更の可否を変更するプレファレンスがあるため、アタッチメントの数はコントロールの設定に依存します。
以下の表に許可される内容を示します。

コントロール	**水平方向サイズ変更**	**水平方向サイズ変更**
Button	可能	可能
Chart	可能	可能
CheckBox	不可能：タイトル長に依存	不可能：タイトルフォントに依存
CustomControl	可能*	可能*
GroupBox	可能	可能
ListBox	可能	可能
PopupMenu	不可能：内容／*bodyWidth* に依存	不可能：フォントサイズに依存
SetVariable	可能：*bodyWidth* が未使用の時	不可能：フォントサイズに依存
水平 Slider	可能	不可能：軸の設定に依存
垂直 Slider	不可能：軸の設定に依存	可能
TabControl	可能	可能
TitleBox	*fixedSize* がゼロ以外の場合のみ	*fixedSize* がゼロ以外の場合のみ
ValDisplay	*BodyWidth* がゼロ以外の場合のみ	不可能：機能のサイズに依存

*しかし、サイズ変更を適切に処理するよう記述されていない場合、それは間違いかもしれません。

SetVariable、TitleBox、ValDisplay は、*bodyWidth* 設定、または TitleBox の場合は *fixedSize* 設定によって、サイズ変更可能または不可となる場合があります。
したがって、これらのコントロールタイプのうちいずれかが、２つの垂直ガイドのアタッチを許可する状態から１つに制限される状態に変化する可能性があります。
そのようなコントロールに２つのガイドがアタッチされており、それが不正な状態となった場合、いずれかのガイドが切断されます。

ガイドモードは描画オブジェクトには適用されない

描画オブジェクト（描画ツールで作成された線、長方形など）はガイドでレイアウトできません。
その結果、ガイドモードでは描画オブジェクトを選択することができません。
選択できるのはコントロールとサブウィンドウのみです。

描画モードでは、描画オブジェクトとコントロールの両方を選択できます。
選択したコントロールは、クリックしてドラッグすることで移動およびサイズ変更が可能です。
コントロールがレイアウトガイドにアタッチされている場合、任意の移動やサイズ変更に対応できないことがあります。
その理由を示すために、描画モードでコントロールが選択され、レイアウトガイドにアタッチされている場合、描画モード中はガイドを操作できませんが、それらのガイドは表示されます。

ガイド付きでコピー＆ペースト

ガイドのコピーとペーストをウィンドウ間で実行する機能は、一部サポートされています。
詳細はヘルプ Copy and Paste Guides で説明されています。

コントロールの削除

プロシージャ内から KillControl コマンドを使ってコントロールを削除できます。
これは、他の設定に応じて外観を変更するコントロールパネルの作成に役立ちます。

描画モード中に、矢印ツールまたは Select Control サブメニューでコントロールを選択し、Delete キーを押すことで、対話的にコントロールを削除できます。

コントロールについての情報を取得

ControlInfo コマンドを使うと、指定されたコントロールに関する情報を取得できます。
これは、チェックボックスの現在の状態やポップアップメニューの現在の設定などの情報を取得するのに便利です。

ControlInfo は通常、実行ボタンまたは同等の機能を持つコントロールパネルで使われます。
ユーザーがボタンをクリックすると、そのアクションプロシージャが ControlInfo を呼び出し、関連する各コントロールの状態を照会し、それに応じて動作します。

ControlInfo は、通常、各コントロールのアクションプロシージャがそのコントロールがクリックされた直後に実行されるタイプのパネルには使われません。

コントロールの更新

ControlUpdate コマンドを使うと、指定されたコントロールを現在の値で再描画させることができます。
このコマンドは、コントロールの値や外観を変更した後、通常更新が行われる前に変更内容を表示するために、プロシージャ内で使います。

ユーザー定義コントロール用ヘルプテキスト

各コントロールタイプにはヘルプテキストプロパティがあり、help キーワードを使って設定することで、ヘルプヒントを追加できます。
Igor Pro 9.0 以降では、ヒントは 1,970 バイトの制限を持っています。
以前のバージョンでは 255 バイトに制限されていました。

次がその例です。

```
Button button0 title="Beep", help={"This button beeps."}
```

ユーザーがコントロール上にマウスを移動すると、ヒントが表示されます。
ユーザー定義のコントロールからヘルプヒントを無効にする組み込みの方法は存在しません。
コントロールのヘルプヒントを使うインターフェイスでは、ユーザーがツールチップを簡単に無効にできる方法を提供することが推奨されます。
画面の小さいユーザーは、ツールチップを無効にしたい場合がよくあります。

書式設定には、限定された HTML タグを使用できます。
ヘルプ HTML Tags in Tooltips を参照してください。

コントロールの変更

コントロールコマンドは、名前パラメーターがウィンドウ内の既存のコントロールと一致しない場合、新しいコントロールを作成します。
コマンドは、名前がウィンドウ内のコントロールと一致する場合、既存のコントロールを変更しますが、コントロールの種類が操作と一致しない場合はエラーを生成します。

例えば、パネルに既に「button0」という名前のボタンコントロールが存在する場合、Button button0 コマンドでそのボタンを変更できます。

```
Button button0 disable=1              // 隠す
```

ただし、ボタンではなくチェックボックスを使うと、「button0 is not a Checkbox（button0 はチェックボックスではありません）」というエラーが発生します。

ModifyControl コマンドと ModifyControlList コマンドを使えば、コントロールの種類を知らなくてもコントロールを変更できます。

```
ModifyControl button0 disable=1     // 隠す
```

タブコントロールと組み合わせて使う場合に特に便利です。

コントロールの無効化と非表示

すべてのコントロールはキーワード「disable=d」をサポートしており、d には以下の値が指定可能です：

0： 通常の操作
1： 非表示
2： ユーザー入力が不可能
3： 非表示かつユーザー入力が不可能

Chart と ValDisplay は、読み取り専用であるため、disable=2 の場合でも外観は変化しません。

SetVariables には noedit キーワードも存在します。
これは disable=2 モードとは異なり、noedit では上下矢印キーによるユーザー入力が可能ですが、disable=2 では不可能です。

背景色のコントロール

コントロールパネルウィンドウの背景色および ControlBar コマンドによって確保されるグラフの端部の領域は、オペレーティングシステムの見た目に合わせるために選択されたグレーの濃淡です。
このグレーは、ModifyGraph cbRGB または ModifyPanel cbRGB で設定されたコントロールバーの背景色が、赤、緑、青の成分がすべて 65535 のデフォルトの純白の場合に使われます。
純白以外の cbRGB 設定は、すべてそのまま反映されます。
ただし、一部のコントロールまたはコントロールの一部はオペレーティングシステムによって描画されるため、別の背景色を選択すると不自然に表示される場合があります。

特別な目的のために、個々のコントロールに対して labelBack キーワードを使って背景色を指定できます。
詳細については、個々のコントロールタイプのヘルプを参照してください。

コントロールの構造体

コントロールプロシージャは、構造ベース形式と推奨されない旧形式の 2 つの形式のいずれかを取ります。
このセクションでは、構造ベース形式を使っていることを前提としています。

コントロールのアクションプロシージャは、関数のパラメーターとして事前定義された組み込み構造体を使います。
このプロシージャは次の形式を取ります。

```
Function ActionProcName(s)
        STRUCT <WMControlTypeAction>& s        // <WMControlTypeAction> は以下の構造体のいずれか
        ...
End
```

各種コントロール構造の名前は以下の通りです。

コントロール形式	構造体名
Button	WMButtonAction
CheckBox	WMCheckboxAction
CustomControl	WMCustomControlAction
ListBox	WMListboxAction
PopupMenu	WMPopupAction
SetVariable	WMSetVariableAction
Slider	WMSliderAction
TabControl	WMTabControlAction

アクション関数は、文書化された eventCode 値のみに応答すべきです。
他のイベントコードや追加フィールドが将来追加される可能性があります。
戻り値は現在使用されていませんが、アクション関数は常にゼロを返す必要があります。

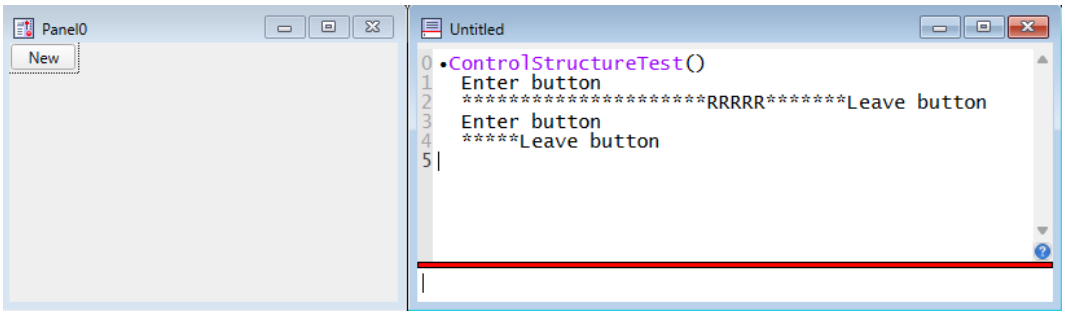
構造体 char 配列のサイズを指定するために使われる定数は Igor Pro 内部のものであり、変更される可能性があります。

同じタイプの異なるコントロールに対して、同じプロシージャを使用できます。
例えば、１つのウィンドウ内のすべてのボタンに対して使用できます。
構造体の ctrlName フィールドを使ってコントロールを識別し、win フィールドを使ってそのコントロールを含むウィンドウを識別します。

コントロール構造体の例

この例は、ボタンコントロールで利用可能な拡張イベントコードを示しています。
この関数は、ボタン領域内で実行した操作に応じて、履歴領域またはコマンドウィンドウに様々なテキストメッセージを出力します。
（実行するには、下記のコードをプロシージャウィンドウにペーストし、コマンドラインで ControlStructureTest() を実行します。）

```
Function ControlStructureTest()
	NewPanel
	Button b0,proc= NewButtonProc
End

Structure MyButtonInfo
	Int32 mousedown
	Int32 isLeft
EndStructure

Function NewButtonProc(s)
	STRUCT WMButtonAction &s
	STRUCT MyButtonInfo bi
	Variable biChanged= 0
	StructGet/S bi,s.userdata
	if( s.eventCode==1 )
		bi.mousedown= 1
		bi.isLeft= s.mouseLoc.h < (s.ctrlRect.left+s.ctrlRect.right)/2
		biChanged= 1
	elseif( s.eventCode==2 || s.eventCode==3 )
		bi.mousedown= 0
		biChanged= 1
	elseif( s.eventCode==5 )
		print "Enter button"
	elseif( s.eventCode==6 )
		print "Leave button"
	endif
	if( s.eventCode==4 )                              //  マウスの移動
		if( bi.mousedown )
			if( bi.isLeft )
				printf "L"
			else
				printf "R"
			endif
		else
			printf "*"
		endif
	endif
	if( biChanged )
		StructPut/S bi,s.userdata                 // コントロールに書き出し
	endif
	return 0
End
```



コントロール構造体 eventMod のフィールド

eventMod フィールドは、各コントロールタイプ用の組み込み構造体に存在します。
これは以下の通り定義されたビットフィールドです。

Int32 eventMod
- Bit 0： マウスボタンが押下
- Bit 1： Shift キーが押下
- Bit 2： Alt キーが押下
- Bit 3： Ctrl キーが押下
- Bit 4： コンテキストメニューのクリックが発生

ビット設定の詳細については、ヘルプ Setting Bit Parameters を参照してください。

コントロール構造体 blockReentry のフィールド

かつて、Mac 版 v6 以前では、コントロールアクションプロシージャの実行に長い時間がかかる場合、アクションを繰り返し、前の呼び出しが終了する前にアクションプロシージャを再度開始することが可能でした（これは再入可能性と呼ばれます）。
つまり、ボタンが長い計算（例えば、時間のかかるカーブフィッティングなど）を開始した場合、最初のフィッティングが完了する前に、ボタンを再度クリックしすぎると、新しいフィッティングを開始することが可能でした。
これをコントロールするため、「blockReentry」メンバーが追加されました。

Igor 7 以降では、再入を防止するコードが追加されたため、blockReentry フィールドは廃止されました。
後方互換性のために残されていますが、効果はありません。

コントロール用のユーザーデータ

userdata キーワードを使って、コントロールに任意のデータを保存できます。

各コントロールには、デフォルトで使われるプライマリの無名のユーザーデータ文字列があります。
また、各ユーザーデータ文字列に名前を指定することで、無制限に追加のユーザーデータ文字列を保存できます。
名前は、Igor で有効な標準的な名前であれば何でも指定できます。

デフォルトのユーザーデータから情報を取得するには、ControlInfo コマンドを使います。
このコマンドは、S_UserData 文字列変数にそのような情報を返します。
名前付きユーザーデータを取得するには、GetUserData コマンドを使い、objID 入力にコントロールの名前を指定する必要があります。
ユーザーデータ文字列は単なるバイト列として扱われます。

内容に対して文字列コマンドを適用しない限り、任意のバイナリデータを格納できます。
小規模なデータの場合、扱いやすさとデータの脆弱性低減のため、可読テキストを使うのが最適でしょう。

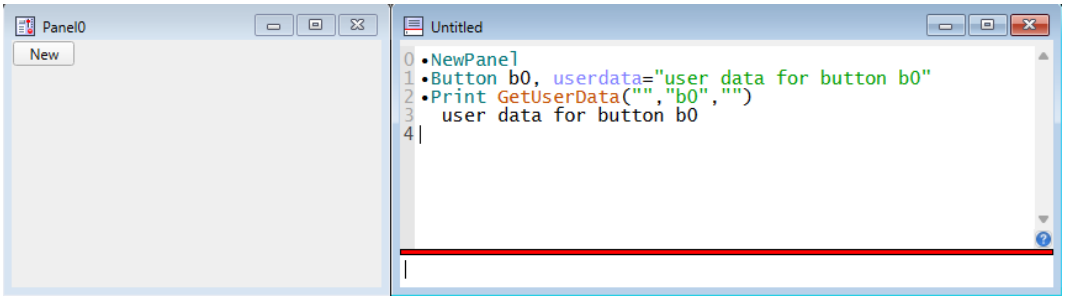
ユーザーデータの保存容量に制限はありませんが、エクスペリメント保存時にはウィンドウの再構築マクロの一部として生成される必要があります。
したがって、巨大なユーザーデータ文字列はエクスペリメントの保存と読み込みを遅くする可能性があります。

ユーザーデータは、コントロールに関連する状態情報を維持するためのグローバル変数の使用を置き換えるか、その使用を減らすことを目的としています。

コントロール用ユーザーデータの例

ユーザーデータを持つボタンの簡単な例を以下に示します。

```
NewPanel
Button b0, userdata="user data for button b0"
Print GetUserData("","b0","")
```

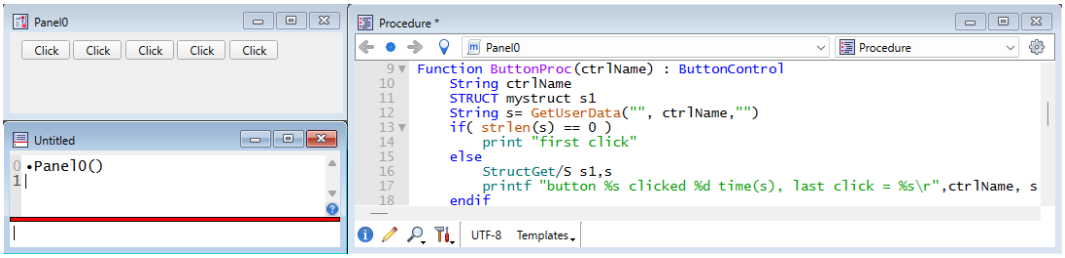


より複雑な例を以下に示します。

以下のコードを新規エクスペリメントのプロシージャウィンドウにコピーし、コマンドラインで Panel0() を実行してください。
その後、ボタンをクリックします。

```
Structure mystruct
        Int32 nclicks
        double lastTime
EndStructure

Function ButtonProc(ctrlName) : ButtonControl
        String ctrlName
        STRUCT mystruct s1
        String s= GetUserData("", ctrlName,"")
        if( strlen(s) == 0 )
                print "first click"
        else
                StructGet/S s1,s
                printf "button %s clicked %d time(s), last click = %s¥r",ctrlName, s1.nclicks,
Secs2Date(s1.lastTime, 1 )+" "+Secs2Time(s1.lastTime,1)
        endif
        s1.nclicks += 1
        s1.lastTime= datetime
        StructPut/S s1,s
        Button $ctrlName,userdata= s
End

Window Panel0() : Panel
        PauseUpdate; Silent 1                   // ウィンドウを構築
        NewPanel /W=(150,50,493,133)
        SetDrawLayer UserBack
```

```
        Button b0,pos={12,8},size={50,20},proc=ButtonProc,title="Click"
        Button b1,pos={65,8},size={50,20},proc=ButtonProc,title="Click"
        Button b2,pos={119,8},size={50,20},proc=ButtonProc,title="Click"
        Button b3,pos={172,8},size={50,20},proc=ButtonProc,title="Click"
        Button b4,pos={226,8},size={50,20},proc=ButtonProc,title="Click"
End
```

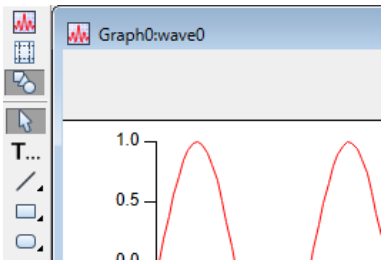


グラフ内のコントロール

コントロールとグラフの組み合わせにより、データをいじり回すのに適したインターフェイスが提供されます。
このようなインターフェイスは、コントロールをグラフに埋め込むか、グラフをコントロールパネルに埋め込むことで作成できます。
このセクションでは前者の手法について説明しますが、通常は後者の手法（グラフをコントロールパネルに埋め込む）が推奨されます。
詳細はヘルプ Embedding and Subwindows を参照してください。

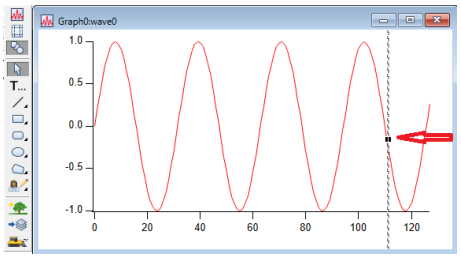
コントロールはグラフ内の任意の位置に配置できますが、グラフウィンドウの端にコントロール専用の領域を確保することが推奨されます。
これらの専用領域に配置されたコントロールは、よりスムーズに動作します。

ControlBar コマンドを使うと、グラフの任意の端に沿って埋め込みコントロール領域の高さを設定できます。
Control Bar ダイアログは、グラフ上部にこの領域を作成するためのインターフェイスを提供します。



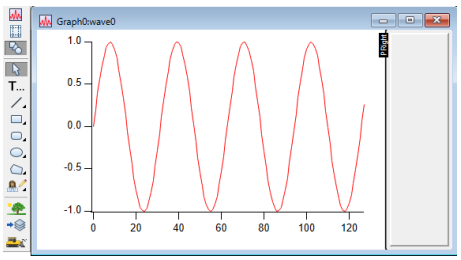

```
Make/O wave0
wave0=sin(x/5)
Display/K=0 root:wave0
ControlBar 50
```

パネルを追加する最も簡単な方法は、描画ツールパレットを表示して描画モードを選択し、グラフの端付近をクリックしてコントロール領域をドラッグすることです。

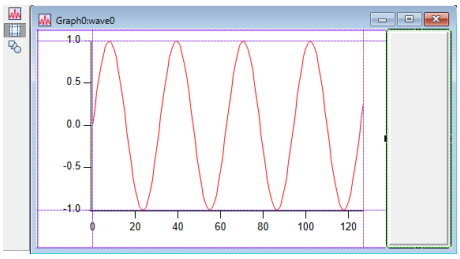


グラフの右上、右下、左上、左下の端をクリックしてください。
この時、ウィンドウ枠を掴まないように注意してください。

PRIGHT は、結果として生成される埋め込みパネルのサブウィンドウの名前です。
「操作」モードではラベルは表示されません。



ガイドモード（ヘルプ Guides Mode for Laying Out Subwindows を参照）では、サブウィンドウの枠をクリックしハンドルをドラッグして、埋め込みウィンドウの位置を調整します。
破線はプロット領域とグラフ領域の境界を表し、サブウィンドウの枠はそれらにスナップして固定されます。



コントロール領域または埋め込みパネルの背景色は、背景を右クリックし、コンテキストメニューのポップアップカラーパレットから色を選択することで設定できます。
詳細は「背景色のコントロール」のセクションを参照してください。

コンテキストメニューは、パネルの周囲の枠のスタイルを調整します。

同じコンテキストメニューを使って、埋め込まれたパネルを削除できます。
これにより、下部のコントロール領域のみが残ります。
コントロール領域を削除するには、内側の端をグラフの外側の端までドラッグします。

描画の制限

グラフのコントロールバー領域では描画ツールを使用できません。
描画ツールは、グラフの端からドラッグして表示されるパネルサブウィンドウ（「グラフ内のコントロール」のセクションを参照）、グラフサブウィンドウを含むパネル、またはグラフを親とする通常のパネルサブウィンドウで使用できます。

コントロールパネル

コントロールパネルは、コントロールを収容するために設計されたウィンドウです。
NewPanel コマンドはコントロールパネルを作成します。

描画ツールはパネルウィンドウで使用でき、コントロールパネルの装飾に活用できます。
コントロールパネルには、コントロールの背後に配置される 2 つの描画レイヤー（UserBack と ProgBack）と、コントロールの前に配置される 1 つのレイヤー（Overlay）があります。
詳細はヘルプ Drawing Layers を参照してください。

パネルの背景色は、背景を右クリックし、表示されるカラーパレットから色を選択することで設定できます。
詳細は「背景色のコントロール」のセクションを参照してください。

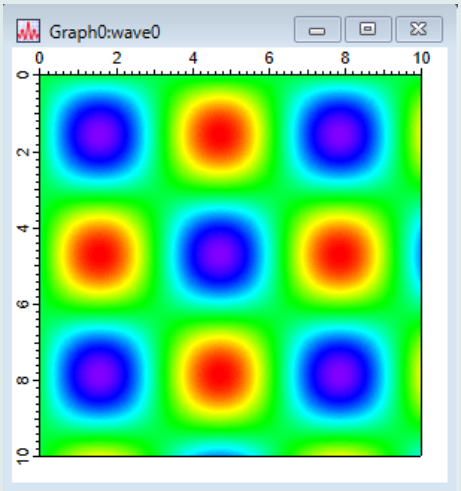
コントロールパネルへの埋め込み

グラフ、テーブル、ノートブック、または別のパネルをコントロールパネルウィンドウに埋め込むことができます。
詳細については、ヘルプ Embedding and Subwindows を参照してください。
この方法は、グラフにコントロール領域を追加するよりもすっきりしています。
また、コントロール付きの 1 つのウィンドウに複数のグラフを埋め込むこともできます。

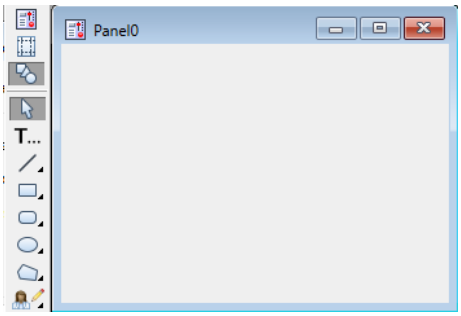
描画モード中にコンテキストメニューを使って、埋め込みウィンドウを追加します。
サブウィンドウのサイズと位置を調整するには、ガイドモードに入り、サブウィンドウの枠をクリックします。
以下に例を挙げます。

1. **埋め込む画像を作成します。**

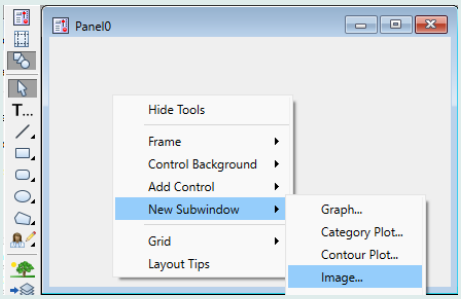
```
Make/N=(256,256,256) wave0
SetScale x,0,10,wave0
SetScale y,0,10,wave0
wave0[][][0]=sin(x)*sin(y)
NewImage/K=0 root:wave0
```



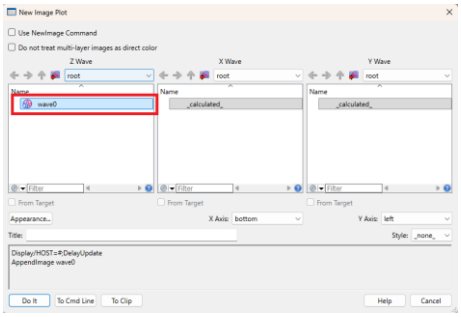
2. **メニュー Windows→New→Panel を選択します。**



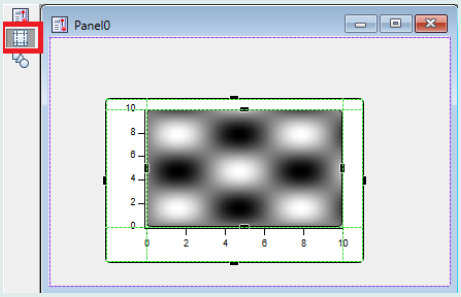
3. **コンテキストメニューから New Subwindow→Image を選択します。**



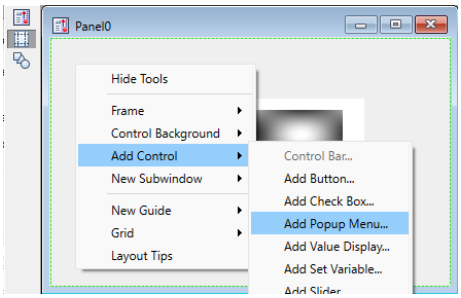
4. **Z Wave で wave0 を選択します。**



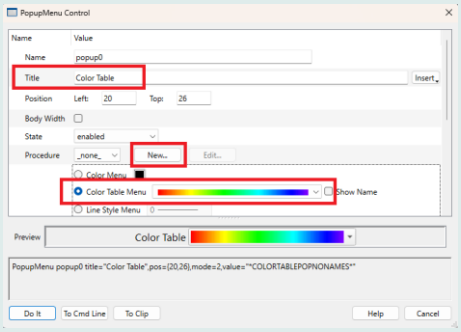
5. **ガイドモードに切り替え、サブウィンドウの大きさを調整します。**



6. **コンテキストメニューから Add Control→Add Popup Menu を選択します。**

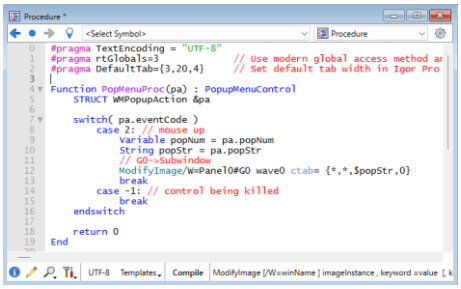


7. **Popup Menu ダイアログで、Title に「Color Table」と入力し、Color Table Menu ラジオボタンにチェックを入れ、配色を Rainbow にします。**



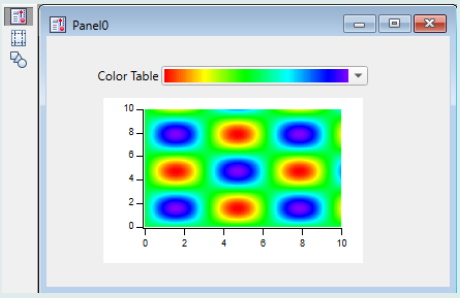
8. **Procedure セクションの New ボタンを押して、プロシージャを次のように書き換えます。**

```
Function PopMenuProc(pa) : PopupMenuControl
    STRUCT WMPopupAction &pa

    switch( pa.eventCode )
        case 2: // mouse up
            Variable popNum = pa.popNum
            String popStr = pa.popStr
            // G0->Subwindow
            ModifyImage/W=Panel0#G0 wave0
ctab= {*,*,$popStr,0}
            break
        case -1: // control being killed
            break
    endswitch
    return 0
End
```



Compile ボタンを押します。

9. **ツールを操作モードに戻して、メニューから配色を選択すると、下の画像の色が変わります。**



コントロールパネル内のノートブックサブウィンドウを使って、ステータス情報を表示したり、長文のユーザー入力を受け付けたりできます。
詳細はヘルプ Notebooks as Subwindows in Control Panels を参照してください。

サブウィンドウのレイアウトに関する詳細は、ヘルプ Subwindow User-Interface Concepts を参照してください。

「ガイドモード」を使うと、複雑なサブウィンドウの配置が多少容易になります。
ガイドモードは、パネル内のコントロールのレイアウトにも活用できます（「ガイドモードでのコントロールのレイアウト」のセクションを参照）。

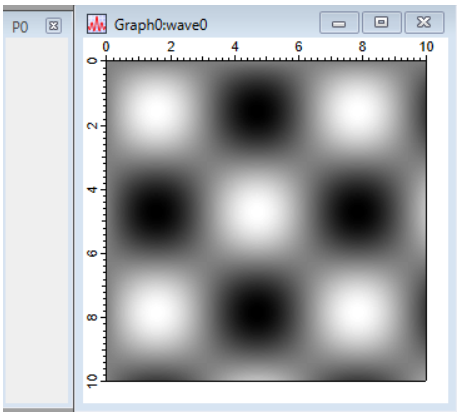
外部コントロールパネル

外部サブウィンドウは、サブウィンドウのように機能するパネルですが、ホストウィンドウにアタッチされた独自のウィンドウ内に存在します。
ホストウィンドウは、グラフ、テーブル、パネル、または Gizmo プロットであることができます。
ホストウィンドウとその外部サブウィンドウは一緒に移動し、一般的には 1 つのウィンドウとして動作します。
外部サブウィンドウは、ホストウィンドウの動作を妨げないという利点があり、通常のサブウィンドウとは異なり、ホストウィンドウによってサイズが制限されません。

外部サブウィンドウパネルを作成するには、/HOST と組み合わせて /EXT フラグ付き NewPanel を使います。

例えば、画像ウィンドウ Graph0 にアタッチしてパネルを作成するには次のコマンドを実行します。

```
NewPanel/HOST=Graph0/EXT=1.0
```

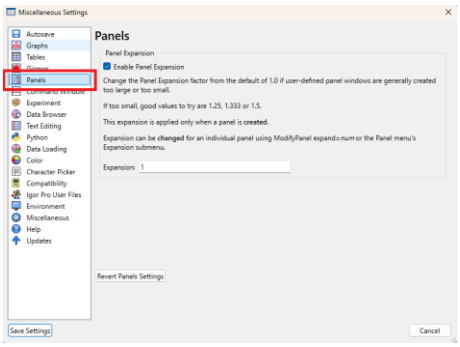


フローティングコントロールパネル

フローティングパネルは、ダイアログを除く他のすべてのウィンドウの上に浮遊します。
フローティングパネルを作成するには、/FLT フラグ付きで NewPanel コマンドを使います。

コントロールパネルの拡大率

Igor Pro 9.0 以降では、Panel メニューの Expansion サブメニュー、または右クリックで呼び出すパネルのコンテキストメニューを使って、コントロールパネルの拡大率を設定できます。
Miscellaneous Settings ダイアログの Panels セクションでは、すべてのパネルに対して設定できます。
以下で説明するように、ほとんどの場合、すべてのパネルに対して設定することを推奨します。



製品出荷時のコントロールパネル拡大率は 1.0 です。
これはコントロールパネルとそのコントロールが「標準」サイズで表示されることを意味します。
標準サイズは、使っている画面のピクセルサイズと様々な複雑なソフトウェア要因によって決まります。
好みに応じて小さすぎたり大きすぎたりする場合があります。
その場合は、コントロールパネル拡大率を使って調整できます。

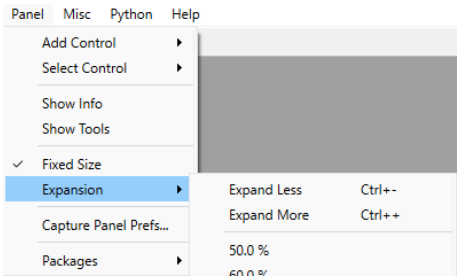
拡大係数を変更すると、パネルウィンドウのサイズおよびパネル内のコントロールやサブウィンドウのサイズがそれに応じて変更されます。
さらに、描画ツールを使ってパネル内に描画された要素も同様に拡大されます。

通常のサイズが理想的でない場合、通常は特定のコントロールパネルではなく、すべてのコントロールパネルの拡大率を変更したいでしょう。
これは、Miscellaneous Settings ダイアログの Panels セクションでデフォルトのコントロールパネル拡大率を変更することで実現できます。
ほとんどの用途では、特定のコントロールパネルの拡大率を変更するよりも、デフォルトを変更することを推奨します。

プログラムでコントロールパネルを作成する場合、Igor Pro 9.0 で追加された NewPanel /EXP=<係数> フラグを使って拡大係数を設定できます。
ただし、ほとんどの用途では /EXP を省略し、ユーザーのデフォルトのコントロールパネル拡大係数が適用されるようにすべきです。

コントロールパネルが再作成される場合（例えばコントロールパネル付きのエクスペリメントを開くとき）、再作成マクロに /EXP=<factor> フラグが存在すると、Igor は指定された拡大係数をパネルに適用します。
/EXP フラグが省略された場合、Igor はユーザーのデフォルトのコントロールパネル拡大係数を適用します。

Igor がコントロールパネルの再作成マクロを生成する場合、コントロールパネルが Miscellaneous Settings ダイアログの Panels セクションで設定されたデフォルトの拡大率以外のものを使っている場合、Igor は NewPanel /EXP フラグを使って目的の拡大率を指定します。
エクスペリメントが別の PC で開かれた場合、特定の拡大係数が適用されます。
これは、ユーザーのデフォルトのコントロールパネル拡大係数を上書きします。
特定のパネル用に設定する代わりに、デフォルトのコントロールパネル拡張を使うことで、ユーザーのプレファレンスを上書きすることを回避できます。



ほとんど必要ありませんが、Igor Pro 9.0 で追加された expand キーワード付きの ModifyPanel コマンドを使って、コントロールパネルウィンドウまたはサブウィンドウの拡大係数をプログラムで設定できます。
メインコントロールパネルウィンドウの場合、これは Expansion サブメニューで設定するのと同じ効果があります。
コントロールパネルサブウィンドウの場合、コントロールのサイズには影響しますが、サブウィンドウ自体のサイズには影響しません。
上記の理由から、通常はプログラムによる設定を控え、ユーザーのデフォルト設定が反映されるようにするのが最善です。

ControlBar コマンドを使ってコントロールバーを作成することで、グラフにコントロールを追加できます。
デフォルトのコントロールパネル拡大係数は、グラフ内のコントロールバーのサイズと、コントロールバー内のコントロールのサイズに影響します。

コントロールパネルの単位

画面上のコントロールパネルウィンドウのサイズと位置は、NewPanel /W=(*left,top,right,bottom*) フラグで設定されます。
コントロールパネル内のコントロールのサイズと位置は、Button や TitleBox などのコントロールコマンドにおける pos={*left,top*} および size={*width,height*} キーワードで設定されます。

歴史的な理由により、これらのパラメーターの解釈はオペレーティングシステムによって異なります。
詳細はヘルプ Control Panel Resolution on Windows を参照してください。

パラメーターはポイントとして解釈されます。
ただし、画面解像度が 96 DPI の場合、既存のエクスペリメントやプロシージャとの互換性を確保するため、ピクセルとして解釈されます。

ユーザーがコントロールパネルのサイズとその内容をコントロールできるようにするため、Igor Pro 9.0 ではコントロールパネル拡張機能が導入されました。
このセクションの残りの部分では、コントロールパネル拡大が NewPanel /W とコントロールコマンドのパラメーターの解釈に与える影響について説明します。

通常のコントロールパネル単位（前述のポイントまたはピクセル）で表現され、Igor がコントロールパネル拡大係数を適用するパラメーターは、*control panel units* で表現されていると言われます。

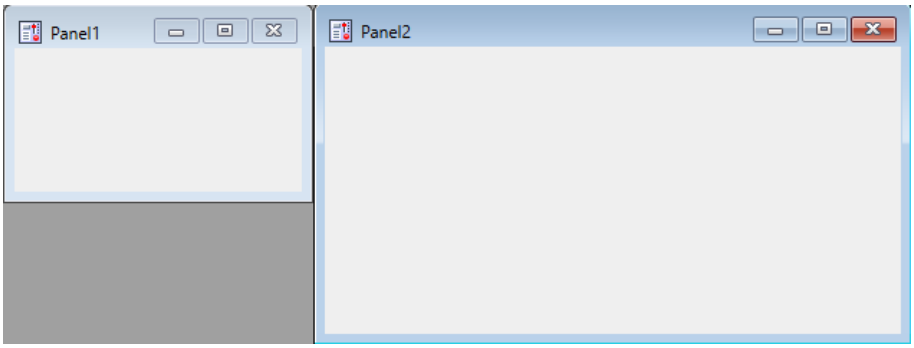
NewPanel 座標の解釈

まず、NewPanel /W=(*left,top,right,bottom*) フラグについて検討します。

コントロールパネルの拡大率が製品出荷時のデフォルト値 1.0 以外の場合、Igor はパネルの幅（左右方向として計算）と高さ（上下方向として計算）を拡大率でスケーリングしますが、左方向と上方向の解釈は変更しません。

デフォルトのコントロールパネル拡大率が 1.0 であると仮定した場合、例えば以下の例を考えてみます。

```
NewPanel/N=Panel1/W=(100,100,300,200)
NewPanel/N=Panel2/EXP=2/W=(100,100,300,200)
```



最初の NewPanel コマンドは、指定された座標にパネルを作成します。
この座標は、前のセクションで説明したように、ポイントまたはピクセルとして解釈されます。

２番目のコマンドでは、/EXP フラグにより拡大率が 2.0 となります。
これは *left* と *top* パラメーターで指定されたウィンドウの位置には影響しません（上図では見やすいように移動しています）。
ただし、Igor が width=*right-left* と height=*bottom-top* から計算するパネルの幅と高さに影響します。
幅と高さを計算した後、Igor は拡大係数を適用します。
その結果、Panel2 の左上隅の位置は Panel1 と同じですが、Panel2 の幅と高さは Panel1 の２倍になります。
両ケースとも最終的な幅=200、高さ=100 となりますが、パネル拡大により Panel2 は Panel1 の２倍の大きさになります。

left、*top*、*right*、*bottom* パラメーターは通常のパネル単位で指定されますが、NewPanel は内部で計算された幅と高さをコントロールパネル単位（つまり、コントロールパネルの拡大が適用される通常の単位）として解釈します。

/I（インチ）または /M（センチメートル）を /W の前に使うと、パラメーターの解釈が変わります。
これらはインチまたはセンチメートルからポイントに変換され、その結果は Display コマンドと同様にポイントとして扱われます。

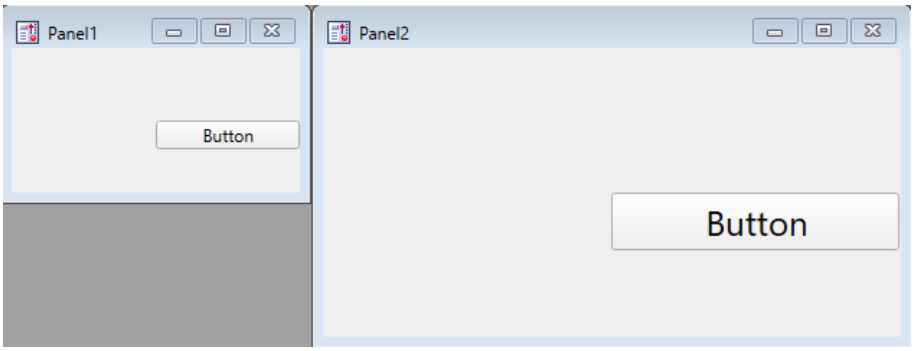
コントロールコマンド座標の解釈

次に、Button や TitleBox などのコントロールコマンドにおける pos={*left,top*} と size={*width,height*} キーワードについて検討します。

これらのパラメーターはすべて、コントロールパネルの単位として扱われます。
つまり、まず前述のようにポイントまたはピクセルとして扱われ、その後、対象パネルの拡大係数が 1.0 以外の場合に適用されます。

前のセクションの Panel1 と Panel2 を使った以下の例を考えてみます。

```
AutopositionWindow /R=Panel1 Panel2                    // パネルを並べる
Button button0 win=Panel1, pos={100,50}, size={100,20}, title="Button"
Button button0 win=Panel2, pos={100,50}, size={100,20}, title="Button"
```

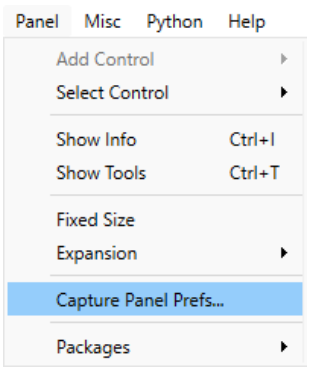


パネルの半分の幅のボタンを作成し、ボタンの左上隅を、上から下への距離の半分、左から右への距離の半分に位置させました。
どちらの場合も、同じパラメーターを使ってこれを行いました。
Button コマンドは、パラメーターをコントロールパネルの単位で表現されていると解釈するため、結果が異なります。

描画座標の解釈

コントロールパネルで描画ツールや描画操作を使用する場合、座標はボタンなどのコントロールコマンドと同様に、コントロールパネル単位として解釈されます。

コントロールパネルのプレファレンス



コントロールパネルのプレファレンスでは、新しいコントロールパネルを作成したときの動作をコントロールできます。
プレファレンスを行うには、パネルを作成し、好みに合わせて設定します。
これをプロトタイプパネルと呼びます。
次に、Panel メニューから Capture Panel Prefs を選択します。

プレファレンスは通常、手動操作に対してのみ有効であり、Igor プロシージャからの自動操作には適用されません。
詳細はヘルプ Preferences で説明しています。

Igor を最初にインストールすると、すべてのプレファレンスは製品出荷時のデフォルト値に設定されます。
ダイアログには、変更したプレファレンス項目が表示されます。

プレファレンスは新規パネルの作成にのみ影響します。

Show Tools カテゴリのチェックボックスを選択すると、新しいパネル作成時に描画ツールパレットが初期状態で表示されるか非表示になるかが決定されます。

コントロールのショートカット

パネルまたはグラフのツールパレットを表示または非表示にする

Ctrl+T を押します。

ツールパレットを使わずにユーザー定義コントロールを移動またはサイズ変更する

Ctrl+Alt キーを押しながらコントロールをクリックします。Ctrl+Alt キーを押したまま、ドラッグまたはサイズ変更を行います。

ツールパレットを使わずにユーザー定義コントロールを追加する

Ctrl+Alt キーを押しながら、Panel または Graph メニューの Add Control サブメニューから選択します。

ユーザー定義コントロールを変更する

Ctrl+Alt キーを押しながらコントロールをダブルクリックします。

これにより、コントロールのあらゆる側面を変更できるダイアログが表示されます。コントロールが既に選択されている場合は、Ctrl+Alt キーを押す必要はありません。

ユーザー定義コントロールのアクションプロシージャを編集する

パネルを編集モード（ツール表示、上から 2 番目のアイコンが選択された状態）にして、コントロールを右クリックします。これによりコンテキストメニューが表示され、Go to <action procedure> 項目が表示されます。

パネル内に埋め込みグラフまたはテーブルを作成する

パネルを編集モードにして、パネルの背景を右クリックします。表示されるコンテキストメニューの New サブメニューからサブウィンドウの種類を選択します。

ユーザー定義コントロールのアクションプロシージャを編集する

パネルを編集モードにして、Alt キーを押しながらコントロールをダブルクリックします。これにより、アクションプロシージャを含むプロシージャウィンドウが表示されます。アクションプロシージャが存在しない場合はビープ音が鳴ります。

埋め込みウィンドウの境界線のスタイルを変更する

パネルを編集モードにして、埋め込みウィンドウを右クリックします。表示されるコンテキストメニューの Frame と Style サブメニューから境界線のスタイルを選択します。

埋め込みウィンドウを削除する

パネルを編集モードにして、埋め込みウィンドウを右クリックします。表示されるコンテキストメニューから Delete を選択します。

グラフの端にあるコントロール領域を除去する

パネルを編集モードにして、または Ctrl+Alt キーを押しながら、コントロール領域の内側端をクリックし、グラフの外側端までドラッグします。

ユーザー定義コントロールの位置を微調整する

コントロールを選択し、矢印キーを押します。
Shift キーを押すと素早く移動できます。