CONTENTS

コマンド

DoIgorMenu [/C /OVRD] *MenuNameStr*, *MenuItemStr*

DoIgorMenu コマンドを使うと、Igor プログラマーは Igor Pro の組み込みメニュー項目を呼び出すことができます。

パラメーター

MenuNameStr “File”、”Graph”、”Load Waves” など、Igor Pro のメニューまたはサブメニューの名前です。

MenuItemStr メニュー項目の表示テキストです。たとえば、（Edit メニューの）”Copy” や、（Windows メニューの）”New Graph”、あるいは（Load Waves サブメニューの）” Load Igor Binary” など。

/C を指定した場合、*MenuItemStr* は “” になることがあります。

フラグ

1 つのコマンド内で /C フラグと /OVRD フラグを同時に使うことはできません。

/C 確認用です。メニュー項目は実行されませんが、その項目が有効だった場合は V_flag が 1 に、無効だった場合は 0 に設定されます。

Igor Pro 9.01 以降では、*MenuItemStr* が ”” の場合、V_Flag、V_isInvisible、S_value はメニュー項目ではなく、メニュー全体に関連付けられます。

/OVRD *MenuNameStr* と *MenuItemStr* で指定されたメニューコマンドを実行する前に、通常行われるチェックをスキップするように指示します。実行時に存在する条件下で、呼び出すメニューコマンドが適切であることを確認する必要があります。

/OVRD フラグの主な用途は、上級プログラマーが、サブウィンドウを操作する時に、現在非表示になっているメニューのコマンドを実行できるようにすることです。例えば、操作モードにあるコントロールパネル内にグラフのサブウィンドウがある場合、メニューバーには Graph メニューが表示されません。通常、ユーザーは Graph メニューの項目を呼び出すことはできません。 しかし、Modify Trace Appearance などの項目を呼び出したい場合があるかもしれません。

/OVRD を使うと、メニューコマンドを実行できますが、その操作が適切であるかどうかを確認する必要があります。Modify Trace Appearance の例では、アクティブなウィンドウまたはサブウィンドウが、少なくとも 1 つのトレースを含むグラフである場合にのみ、メニューコマンドを実行してください。

このフラグは、Igor Pro 7.0 で追加されました。

詳細

メニュー名とメニュー項目のテキストはすべて英語です。
これにより、Igor Pro のローカライズ版向けに開発されたコードが、すべてのバージョンで実行可能になります。
なお、*MenuItemStr* では末尾に “...” は使われません。

対応するメニュー項目が有効化された場合、V_flag は 1 に設定されます。
これは通常、そのメニュー項目が正常に選択されたことを意味します。
それ以外の場合は、V_flag は 0 になります。
V_flag は、表示されたダイアログ（ある場合）の成否を反映するものではありません。

Igor Pro 7 で追加された V_isInvisible は、対応するメニュー項目が非表示だった場合に 1 に設定されます。
これはかなり稀なケースであり、ほとんどの場合、V_isInvisible は 0 に設定されます。
非表示のメニュー項目とは、File→Adopt All のようなもので、ユーザーがメニューを呼び出す時に Shift キーを押さない限り非表示になります。
これは、HideIgorMenus によって非表示にされたメニュー項目とは異なります。
後者の場合、V_isInvisible は 0 に設定されます。

Igor 9.0 で追加された S_value には、切り替わるメニュー項目のテキストが設定されます。
これは、Show Tools や Hide Tools のように、2 つの状態を切り替える常時有効なメニュー項目に役立ちます。

メニュー項目の選択により、コマンドを生成するダイアログが表示された場合、Do It ボタンをクリックすると、あたかも Execute/Z コマンドが使われたかのように、コマンドラインを使わずにそのコマンドが即座に実行されます。
To Cmd Line ボタンをクリックすると、コマンドが先頭に挿入されるのではなく、コマンドラインの末尾に追加されます。

DoIgorMenu コマンドは、カーブフィッティング中や、動的メニュー項目の関数が実行中の間は、メニューの選択を試みません。
もちろん、DoIgorMenu を使うべきではない状況は他にもあるでしょう。

File メニューの一部の項目のテキストは、アクティブなウィンドウの種類によって変化します。
このような場合、*MenuItemStr* パラメーターとして汎用テキストを指定する必要があります。
Save Notebook や Save Procedure などの代わりに、Save Window、Save Window As、Save Window Copy、Adopt Window、Revert Window を使ってください。
Page Setup For All Graphs などの代わりに、Page Setup を使ってください。
Print Graph などの代わりに、Print を使ってください。

Edit→Insert File メニューは、以前は Insert Text という名前でした。
互換性を確保するため、*MenuItemStr* には Insert File または Insert Text のいずれかを指定することで、この項目を呼び出すことができます。

参照

SetIgorMenuMode、ShowIgorMenus、HideIgorMenus、Execute コマンド

SetIgorMenuModeProc.ipf という WaveMetrics プロシージャファイルには、すべてのメニューとメニュー項目に対する SetIgorMenuMode コマンドが含まれています。
これを読み込むには、次を記述します。

```
#include <SetIgorMenuModeProc>
```

DoUpdate [/E=*e* /W=*targWin* /SPIN=*ticks*]

DoUpdate コマンドは、ウィンドウと依存オブジェクトを更新します。

フラグ

/E=*e*　　/W オプションと組み合わせて使うと、/E=1 を指定することで、ユーザーコードの実行中にマウスイベントを受け入れることができる進行状況ウィンドウとしてウィンドウを指定します。現在、進行状況ウィンドウとして使用できるのはコントロールパネルウィンドウのみです。

別のウィンドウが進行状況ウィンドウとして指定されている場合、このコマンドによりそのウィンドウの指定が解除されます。

/W=*targWin*　指定されたウィンドウのみを更新します。依存関係は更新されず、その他の更新処理も行われません。

現在、/W フラグが有効なのはグラフウィンドウとパネルウィンドウのみです。

V_Flag は、ウィンドウが存在するかどうかを示す真偽値として設定されます。V_Flag のその他の値については、ヘルプ Progress Windows を参照してください。

/SPIN=*ticks*　コントロールプロシージャの開始から「スピニングビーチボール」が表示されるまでの遅延時間を設定します。*ticks* はティック単位（1 秒の 60 分の 1）での遅延時間です。/W フラグと併用しない限り、/SPIN は単に遅延時間を設定するだけで、更新は行われません。

詳細

プロシージャから DoUpdate を呼び出すことで、更新が必要なオブジェクトの更新を強制します。
更新が必要なウィンドウに加え、前回の更新以降に変更された他のオブジェクトに依存するオブジェクト（文字列変数、数値変数、ウェーブ、コントロール）も更新します。
ページレイアウトのウィンドウは、すぐには更新されない場合があります。
ページレイアウトの更新に関する詳細については、ヘルプ Automatic Updating of Layout Objects を参照してください。

以下の条件を満たす場合、Igor Pro は自動的に更新を実行します：

- 実行中のユーザープロシージャがない
- インタプリター型プロシージャ（マクロ、プロシージャ、Window タイプのプロシージャ）が実行中であり、PauseUpdate または DelayUpdate が有効になっていない

ユーザー定義関数の実行中は自動的に DoUpdate を実行しません。
ユーザー定義関数内から DoUpdate を呼び出すことで、更新を強制することができます。

参照

DelayUpdate、PauseUpdate、ResumeUpdate コマンド
ヘルプ Progress Windows

DoWindow [/B[=*bname*]/C/D/F/H/HIDE=*h* /K/N/R/W=*targWin*] [*windowName*]

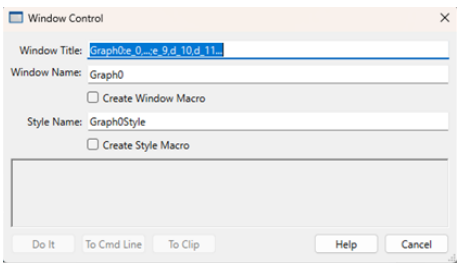
DoWindow コマンドは、ウィンドウのさまざまなパラメーターや特性をコントロールします。
/S または /T フラグを使う場合、DoWindow には追加の形式があります（下記を参照）。

DoWindow はサブウィンドウ構文をサポートしていません。

パラメーター

windowName は、最上位のグラフ、テーブル、ページレイアウト、ノートブック、パネル、Gizmo、カメラ、または XOP ターゲットウィンドウの名前です。
windowName はサブウィンドウのパスにはできません。

ウィンドウの名前は、そのタイトルとは異なります。
タイトルはウィンドウのタイトルバーに表示されます。
名前は、コマンドからウィンドウを操作するために使われます。
Window Control ダイアログ（Windows メニューの Control サブメニュー内）を使って、名前とタイトルの両方を確認できます。



フラグ

/B[=*bname*] 指定したウィンドウを最下層（デスクトップの一番下）に移動するか、ウィンドウ *bname* の後ろに移動します。

/C ターゲットウィンドウの名前を指定された名前に変更します。指定された名前は、既存のウィンドウマクロの名前である場合を除き、他のいかなるオブジェクトにも使ってはいけません。

/C/N 対象ウィンドウの名前を変更し、そのウィンドウ用の新しいマクロを作成します。ただし、マクロや関数が実行中の場合、/N は何も行いません。/N はノートブックには適用できません。

/D ウィンドウに関連付けられているファイルがある場合は、それを削除します（ノートブックのみ）。

/F 指定された名前のウィンドウを最前面（デスクトップの最上部）に表示します。

/H 操作の対象としてコマンドウィンドウを指定します。/H を使う場合、*windowName* を指定してはならず、/B と /HIDE フラグのみが有効となります。

/H オプションを使うと、コマンドウィンドウを最前面（デスクトップの最上部）に表示できます。

/H/B を使うと、コマンドウィンドウをデスクトップの最下部に移動できます。

/H/HIDE を使うと、コマンドウィンドウを非表示または表示できます。

/HIDE=*h* ウィンドウの非表示状態を設定します。

h=0： 表示
h=1： 非表示
h=?： 変数 V_flag を次のように設定します：
0： ウィンドウが存在しない
1： ウィンドウが表示
2： ウィンドウが非表示

また、GetWindow を使って非表示の状態を読み取り、SetWindow を使って設定することもできます。

/K 指定された名前のウィンドウを Kill します。

DoWindow/K の代わりに KillWindow を使うことを推奨します。

/N 指定された名前のウィンドウに対して、新しいウィンドウマクロを作成します。ただし、マクロや関数が実行中の場合、/N は何も行いません。/N はノートブックには適用できません。

/R 指定されたウィンドウのウィンドウマクロを置き換える（更新する）か、まだ存在しない場合は作成します。ただし、マクロや関数が実行中の場合、/R は何も行いません。/R はノートブックには適用できません。

/R/K 指定されたウィンドウのウィンドウマクロを置き換える（更新する）か、まだ存在しない場合はそれを作成してから、そのウィンドウを終了させます。ただし、マクロや関数が実行中の場合、/R は何も行いません。/R はノートブックには適用されません。

/W=*targWin* *targWin* をターゲットウィンドウとして指定します。また、*windowName* の指定も必要です。これは主に、常に最前面に表示されるフローティングパネルで使います。外部サブウィンドウのサブウィンドウ指定は、/T フラグを指定する場合、またはフラグを一切指定しない場合にのみ使用できます。

詳細

DoWindow は、DoWindow の実行後に指定された名前のウィンドウが存在した場合は変数 V_flag を 1 に、そのようなウィンドウが存在しなかった場合は 0 に、ウィンドウが非表示の場合は 2 に設定します。

windowName を引数として指定し、フラグを指定せずに DoWindow を呼び出すことで、ウィンドウを変更することなく、そのウィンドウが存在するかどうかを確認できます。
より適切な方法は、サブウィンドウをサポートしている WinType を使うことです。

/N フラグと併用する場合、*windowName* は他のいかなるオブジェクトの名前とも重複してはいけません。
/C フラグと併用する場合、*windowName* は他のいかなるオブジェクトの名前とも重複してはいけませんが、既存のウィンドウマクロの名前である場合はこの限りではありません。

マクロや関数の実行中に /R と /N フラグを指定しても、何の効果もありません。
これは、プロシージャの実行中にその内容を変更すると、予期せぬ望ましくない結果が生じるためです。
ただし、Execute/P コマンドを使えば、プロシージャの実行終了後に DoWindow コマンドを実行させることができます。
例えば：

```
Function SaveWindowMacro(windowName)
    String windowName                       // 最上位のグラフまたはテーブルに対して ""

    if (strlen(windowName) == 0)
        windowName = WinName(0, 3)          // 最上位のグラフまたはテーブルの名前
    endif

    String cmd
    sprintf cmd, "DoWindow/R %s", windowName
    Execute/P cmd
End
```

/D フラグを /K フラグと組み合わせて使ううと、ノートブックウィンドウを強制終了し、関連するファイルがある場合はそれを削除することができます。
/D フラグは、その他の種類のウィンドウには何の影響も与えず、/K フラグが指定されていない場合は何の効果もありません。

例

```
DoWindow Graph0             // Graph0 ウィンドウが存在する場合、V_flag を 1 に設定する
DoWindow/F Graph0           // Graph0 を最前面／ターゲットウィンドウにする
DoWindow/C MyGraph          // ターゲットウィンドウ（Graph0）の名前を MyGraph に変更する
DoWindow/H/B                // コマンドウィンドウを後ろに回す
DoWindow/D/K Notebook2      // Notebook2 を Kill し、そのファイルを削除する
```

参照

RenameWindow、MoveWindow、MoveSubWindow、SetActiveSubwindow、KillWindow コマンド

HideProcedures、IgorInfo コマンド

DoWindow /T *windowName*, *windowTitleStr*

DoWindow/T コマンドは、指定されたウィンドウのタイトルを、指定されたタイトルに設定します。

詳細

タイトルはウィンドウのタイトルバーに表示され、対応する Windows のサブメニューにも表示されます。
ウィンドウのコマンドには引き続きウィンドウ名が使われるため、例えば、ウィンドウ名（windowName がグ

ラフまたはテーブルの場合）は、New Layout ダイアログにはタイトルではなくウィンドウ名として表示されます。

Window Control ダイアログ（Windows メニューの Control サブメニュー内）を使って、名前とタイトルの両方を確認できます。

windowName は、ウィンドウの名前、または kwTopWin や kwFrame といった特別なキーワードです。

windowName が kwTopWin の場合、DoWindow は最上位のターゲットウィンドウのタイトルを変更します。

windowName が kwFrame の場合、DoWindow は、Windows 環境でのみ持つ「フレーム」または「アプリケーション」ウィンドウのタイトルを変更します。
これは、Igor Pro のメニューとステータスバーが表示されるウィンドウです。

Window Control ダイアログでは、kwFrame はサポートされていません。
フレームのタイトルは、Igor Pro を終了するか、例えば例に示すように復元されるまで保持されます。
windowTitleStr を "" に設定すると、通常のフレームタイトルが復元されます。

例

```
DoWindow/T MyGraph, "My Really Neat Graph"
DoWindow/T kwFrame, "My Igor-based Application"
DoWindow/T kwFrame, ""              // 通常のフレームタイトルに戻す
```

参照

次の DoWindow コマンドのセクション

DoWindow /N/S=*styleMacroName windowName*

DoWindow /R/S=*styleMacroName windowName*

DoWindow/S コマンドは、指定されたスタイルマクロ名を使って、指定されたウィンドウ用の新しいスタイルマクロを作成します。
指定されたウィンドウのウィンドウマクロを作成したり置き換えたりすることはありません。

フラグ

/N/S=*styleMacroName*　指定されたウィンドウに基づいて、指定された名前を持つ新しいスタイルマクロを作成します。

/R/S=*styleMacroName*　指定された名前のスタイルマクロを、指定されたウィンドウに基づいて作成または置き換えます。

詳細

/R または /N フラグは、/S フラグの前に指定する必要があります。

/S フラグが指定されている場合、DoWindow コマンドでは、指定されたウィンドウのウィンドウマクロは作成または置換されません。

マクロや関数の実行中に /R と /N フラグを実行しても、何の効果もありません。
これは、プロシージャの実行中にその内容を変更すると、予期せぬ望ましくない結果が生じるため、必要な措置です。

参照

前の DoWindow コマンドのセクション

DoXOPIdle

DoXOPIdle コマンドは、開いているすべての XOP に対して IDLE イベントを送信します。
このコマンドは非常に特殊で、通常はあまり行われません。
一般的に、このコマンドを使う必要があるのは、その XOP の作成者のみです。

詳細

一部の XOP（External Operation コードモジュール）では、特定のタスクを実行するために IDLE イベントが必要となります。

Igor Pro プログラムの実行中は、XOP に対して IDLE イベントを自動的に送信しません。
ユーザー定義プログラムから DoXOPIdle を呼び出すことで、イベントの送信を強制することができます。

DPSS [*flags*] *numPoints*, *numWindows*

DPSS コマンドにより、Slepian の離散長軸楕円体列（Slepian's Discrete Prolate Spheroidal Sequences）が生成されます。

DPSS コマンドは、Igor Pro 7.0 で追加されました。

フラグ

/DEST=*destWave*　DPSS を、*destWave* で指定されたウェーブに保存します。対象のウェーブがすでに存在する場合、そのウェーブは上書きされます。

ユーザー関数内で、宛先ウェーブに対するウェーブ参照を作成します。詳細は、ヘルプ Automatic Creation of WAVE References を参照してください。

/DEST を省略した場合、このコマンドの結果は現在のデータフォルダー内のウェーブ M_DPSS に保存されます。

/EV=*evWave*　*evWave* で指定されたウェーブにおいて、最初の *numWindows* 個の固有値を保存します。固有値は、対称な三対角行列に対して計算されます。これらは実数で、正の値であり、1 に近い値となります。これらは、マルチテーパー計算におけるバイアスの推定に使用できます。

/FREE　出力ウェーブをフリーウェーブとして生成します。

/FREE はユーザー定義関数内でのみ使用可能であり、コマンドラインやマクロ内では使用できません。

/FREE を使う場合、*destWave*、*evWave*、*sumsWave* はパスではなく、単純な名前でなければなりません。

フリーウェーブに関する詳細は、ヘルプ Free Waves を参照してください。

/NW=*nw*　時間帯域積を指定します。この値は通常、[2,6] の範囲内である必要があります。時間帯域積 *nw* が与えられた場合、分散効率を最大化するためには、2**nw* 以下のテーパー数を使うことが推奨されます。時間帯域積のデフォルト値は 3 です。

/DTPS=*sumsWave*　*sumsWave* で指定されたウェーブに、生成された DPSS ウィンドウの合計値を保存します。

/Q　履歴への印刷情報を非表示にします。

/Z　エラーを抑制します。変数 V_Flag は、成功した場合は 0 に、それ以外の場合は -1 に設定されます。

詳細

DPSS は、*numPoints* × *numWindows* の次元を持つ 2D 倍精度ウェーブにおいて、Slepian の離散長軸楕円体列を生成します。

/DEST を省略した場合、このコマンドにより、現在のデータフォルダーに出力ウェーブ M_DPSS が作成されます。
シーケンス／テーパーは、出力ウェーブ内で列として配置されます。

例

```
DPSS/DEST=dpss5 1000,5
Display dpss5[][0],dpss5[][1],dpss5[][2],dpss5[][3],dpss5[][4]
ModifyGraph rgb(dpss5#1)=(0,65535,0),rgb(dpss5#2)=(1,16019,65535)
ModifyGraph rgb(dpss5#3)=(65535,0,52428),rgb(dpss5#4)=(0,0,0)

// 異なるシーケンスは直交している
MatrixOp/o aa=col(dpss5,1)*col(dpss5,4)
Integrate/METH=1 aa/D=W_INT
Print W_INT[numpnts(W_INT)-1]
```

参照

MultiTaperPSD、WindowFunction、ImageWindow、Hanning コマンド

参考文献

D. Slepian, "Prolate spheroidal wave functions, Fourier analysis and uncertainty -- V: The discrete case.", Bell Syst. Tech J., vol 57 pp. 1317-1430, May 1978.

DrawAction [/L=*layerName* /W=*winName*] *keyword=value* [, *keyword=value* ...]

DrawAction コマンドは、名前が付けられた描画オブジェクトグループ、または描画レイヤー全体を削除、挿入、読み戻しします。

パラメーター

DrawAction は、１行に複数の keyword=value 形式のパラメーターを受け付けます。

beginInsert [=*index*]
: インデックス位置の前またはその位置、あるいは getgroup または delete パラメーターで指定された位置に描画コマンドを挿入します。それ以外の場合は、位置は 0 となります。

commands [=*start,stop*]
: 開始インデックス値と終了インデックス値の間、getgroup で定義された範囲、あるいはそれ以外の場合はレイヤー全体にわたる描画オブジェクトのコマンドを S_recreation に格納します。変数 V_flag には、含まれる描画要素の数が設定されます。描画要素がない場合、S_recreation の長さは 0 になります。

delete [=*start,stop*]
: *start* インデックス値と *stop* インデックス値の間にある描画オブジェクト、getgroup で定義された範囲内のオブジェクト、あるいはそれ以外の場合はレイヤー全体のオブジェクトを削除します。

extractOutline [=*start,stop*]
: 開始インデックス値と終了インデックス値の間、getgroup で定義された範囲、またはそれ以外の場合はレイヤー全体にわたるポリゴンの輪郭を格納します。ウェーブ W_PolyX と W_PolyY には、NaN を区切り文字とする座標が含まれます。

V_npnts にはオブジェクト数が、V_startPos には開始インデックス値が、V_endPos には終了インデックス値が含まれます。

座標は、最初に検出されたオブジェクトのものです。

endInsert　挿入モードを終了します。

getgroup=*name*　指定されたグループの開始インデックスと終了インデックスを、それぞれ V_startPos と V_endPos に格納します。_all_ を使うと、レイヤー全体を指定できます。グループが存在する場合、V_flag を真に設定します。

フラグ

/L=*layerName*　コマンドの対象となる描画レイヤーを指定します。*layerName* は、SetDrawLayer で指定された描画レイヤーのいずれかです。

/W=*winName*　描画対象として、指定されたウィンドウまたはサブウィンドウを設定します。省略された場合、その操作はアクティブなウィンドウまたはサブウィンドウに適用されます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

S_recreation に格納されているコマンドは、そのウィンドウの再作成マクロの対象となるオブジェクトの範囲に対して生成されるコマンドと同じですが、各オブジェクトの前には次のような形式のコメント行が追加されています。

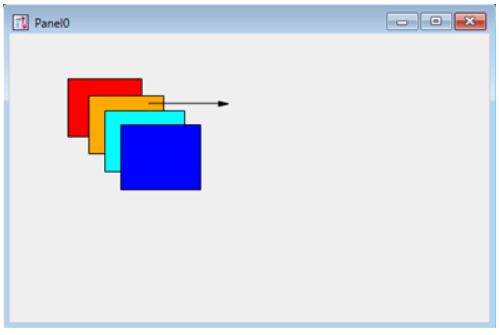
```
// ;ITEMNO:n;
```

ここで、n は描画対象のアイテム番号です。

例

名前付きグループを含む図を作成します。

```
NewPanel /W=(455,124,936,413)
SetDrawEnv fillfgc= (65535,0,0)
DrawRect 58,45,132,103
SetDrawEnv gstart,gname= fred
SetDrawEnv fillfgc= (65535,43690,0)
DrawRect 79,62,154,120
SetDrawEnv arrow= 1
DrawLine 139,70,219,70
SetDrawEnv gstop
SetDrawEnv fillfgc= (0,65535,65535)
DrawRect 95,77,175,138
SetDrawEnv fillfgc= (0,0,65535)
DrawRect 111,91,191,156
```

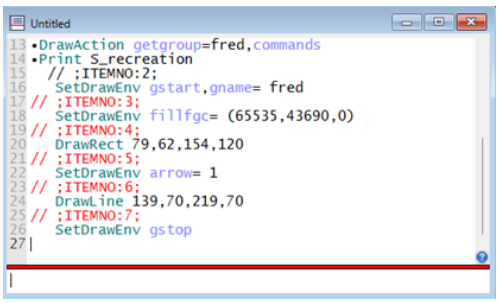


“fred” グループのコマンドを取得して出力します。

```
DrawAction getgroup=fred,commands
Print S_recreation
```

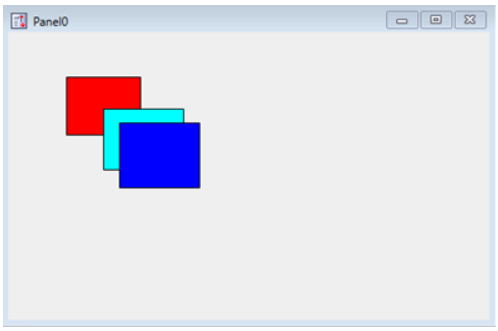
出力は次のようになります。

```
// ;ITEMNO:2;
SetDrawEnv gstart,gname= fred
// ;ITEMNO:3;
SetDrawEnv fillfgc= (65535,43690,0)
// ;ITEMNO:4;
DrawRect 79,62,154,120
// ;ITEMNO:5;
SetDrawEnv arrow= 1
// ;ITEMNO:6;
DrawLine 139,70,219,70
// ;ITEMNO:7;
SetDrawEnv gstop
```



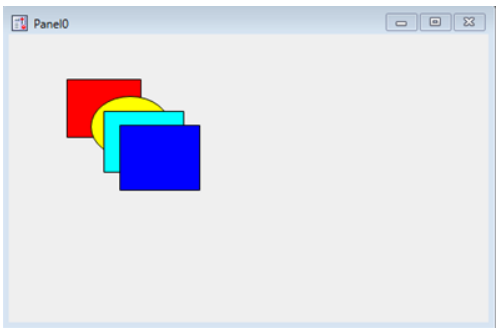
グループ “fred”（オレンジ色の四角形と矢印）を別のオブジェクトに置き換えます。
まず、そのグループを削除し、挿入モードに入ります。

```
DrawAction getgroup=fred, delete, begininsert
```



次に、代替案を描きます。

```
SetDrawEnv gstart,gname= fred
SetDrawEnv fillfgc= (65535,65535,0)
DrawOval 82,62,161,123
SetDrawEnv gstop
```



最後に、挿入モードを終了します。

```
DrawAction endinsert
```

参照

SetDrawEnv、Drawing コマンド

DrawArc [/W=*winName* /X/Y/ORIG] *xOrg*, *yOrg*, *arcRadius*, *startAngle*, *stopAngle*

DrawArc コマンドは、*xOrg* と *yOrg* を中心とする反時計回りの円弧を描画します。

パラメーター

(*xOrg*, *yOrg*) は、現在アクティブな座標系における円弧の中心点を定義します。

角度は、反時計回りに増加する形で度単位で測定されます。
startAngle は円弧の開始角度を指定し、*stopAngle* は終了角度を指定します。
stopAngle が *startAngle* と等しい場合、*stopAngle* に 360 度が加算されます。
したがって、*startAngle=stopAngle* と設定することで、円を描くことができます。

arcRadius は、/X または /Y が指定されていない限り、(*xOrg*, *yOrg*) からのポイント単位の半径距離です。

フラグ

/ORIG　　円弧の始点と終点から、原点 (*xOrg*, *yOrg*) まで線を引きます。矢印がある場合は、円弧の曲線上の始点と終点に引き続き描画されます。
/ORIG は Igor Pro 10 で追加されました。

/W=*winName*　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

/X　現在の X 座標系を使って、円弧の半径（*arcRadius*）を測定します。/Y も指定された場合、円弧は楕円形になることがあります。

/Y　現在の Y 座標系を使って、円弧の半径（*arcRadius*）を測定します。/X も指定された場合、円弧は楕円形になることがあります。

詳細

円弧は、ポリゴンやベジエ曲線と同様に、現在のダッシュパターンや矢印の向き設定を反映します。
実際、円弧はベジエ曲線を使って実装されています。

通常、円弧はプログラムで作成します。
円弧のようなオブジェクトを描画する必要がある場合は、ベジエ曲線を使ったほうがよいでしょう。
ベジエ曲線の方が柔軟性が高く、調整も容易だからです。
しかし、円弧には手描きに便利な機能が 1 つあります。
それは、原点をサポートされている座標系のいずれかに設定でき、半径がポイント単位で指定できるという点です。

しかし、円弧には手描きに便利な機能が 1 つあります。
それは、原点をサポートされている座標系のいずれかに設定でき、半径をポイント単位で指定できるという点です。

弧をインタラクティブに描画するには、ヘルプ Arcs and Circles の手順を参照してください。

参照

Drawing コマンド
SetDrawEnv、SetDrawLayer コマンド
DrawBezier、DrawOval、DrawAction コマンド

DrawBezier [/W=*winName* /ABS] *xOrg*, *yOrg*, *hScaling*, *vScaling*, *xWaveName*, *yWaveName*

DrawBezier [/W=*winName* /ABS] *xorg*, *yorg*, *hScaling*, *vScaling*

DrawBezier /A [/W=*winName*]

DrawBezier コマンドは、曲線の最初の点を *xOrg* と *yOrg* に配置してベジエ曲線を描画します。

パラメーター

(*xOrg*, *yOrg*) は、現在アクティブな座標系におけるベジェ曲線の始点を定義します。

hScaling と *vScaling* は、原点を中心とした水平および垂直の拡大率を設定するもので、1 は 100% を意味します。

xWaveName、*yWaveName* バージョンの DrawBezier は、指定された X と Y ウェーブからデータを取得します。
この接続は維持されるため、いずれかのウェーブに変更が加わると、ベジェ曲線が更新されます。

頂点のリストをリテラルとして受け取るバージョンの DrawBezier を使うには、１行目に任意の数の頂点を配置し、その後、すべての頂点を定義するために必要な数の /A バージョンを使います。

フラグ

/A　　指定された頂点を、現在開いているベジェ曲線（新しく描画されたもの、または現在選択されているもの）に追加します。

/ABS　　デフォルトで最初のデータポイントを残りのデータから差し引く処理を無効にします。

/W=*winName*　　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

ベジェ曲線を定義するウェーブは、1+3*n* 個のデータポイントで構成されなければなりません。
３つおきのデータポイントがアンカーポイントとなり、曲線上に位置します。
その間のポイントはコントロールポイントであり、隣接するアンカーポイントに対する曲線の方向を決定します。

通常、ベジェ曲線は描画ツールを使用して作成、編集するものであり、値を計算するものではありません。
操作方法については、ヘルプ Polygon Tool とヘルプ Editing a Bezier Curve を参照してください。

/ABS フラグを指定することで、最初のデータポイントのデフォルトでの差し引きを無効にすることができます。

データを再指定せずに、原点とスケールのみを変更するには、次のようにします。

```
DrawBezier xOrg, yOrg, hScaling, vScaling,{}
```

ベジエ曲線は、NaN の座標ペアを追加することで、セグメントに分割することができます。
詳細は、ヘルプ Segmented Bezier Curves を参照してください。

参照

Drawing コマンド
SetDrawEnv、SetDrawLayer コマンド
DrawArc、DrawPoly、DrawAction、BezierToPolygon コマンド

DrawLine [/W=*winName*] *x0*, *y0*, *x1*, *y1*

DrawLine コマンドは、対象のグラフ、レイアウト、またはコントロールパネル上に、(*x0*, *y0*) から (*x1*, *y1*) までの線を描画します。

フラグ

/W=*winName*　　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

座標系、線の太さ、色、破線のパターン、その他のプロパティは、現在の作図環境によって決定されます。
線は、SetDrawLayer によって指定された、そのウィンドウの現在の作図レイヤーに描画されます。

参照

Drawing コマンド
SetDrawEnv、SetDrawLayer、DrawAction コマンド

DrawOval [/W=*winName*] *left*, *top*, *right*, *bottom*

DrawOval コマンドは、*left*、*top*、*right*、*bottom* で定義された矩形内に、対象のグラフ、レイアウト、またはコントロールパネルに楕円を描画します。

フラグ

/W=*winName*　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

座標系、楕円の太さ、色、その他のプロパティは、現在の描画環境によって決定されます。
楕円は、SetDrawLayer によって指定された、そのウィンドウの現在の描画レイヤーに描画されます。

参照

Drawing コマンド
SetDrawEnv、SetDrawLayer、DrawAction コマンド

DrawPICT [/W=*winName*][/RABS] *left*, *top*, *hScaling*, *vScaling*, *pictName*

DrawPICT コマンドは、指定された画像をターゲットのグラフ、レイアウト、またはコントロールパネルに描画します。*left* と *top* パラメーターは、画像の左上隅の位置を設定します。*hScaling* と *vScaling* は、水平と垂直方向の拡大率を設定します。1 は 100% を意味します。

フラグ

/RABS　指定された画像を絶対スケーリングを使って描画します。このモードでは、画像は、ポイント (x0, y0) については *left* と *top* で定義される矩形内、ポイント (x1, y1) については *hScaling* と *vScaling* で定義される矩形内に、それぞれ描画されます。

/W=*winName*　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

left と *top* パラメーターの座標系は、現在の描画環境によって決定されます。
図形は、SetDrawLayer によって指定された、そのウィンドウの現在の描画レイヤーに描画されます。

参照

DrawPoly [/W=*winName* /ABS] *xorg*, *yorg*, *hScaling*, *vScaling*, *xWaveName*, *yWaveName*

DrawPoly [/W=*winName* /ABS] *xorg*, *yorg*, *hScaling*, *vScaling*,

DrawPoly [/W=*winName*]/A

DrawPoly コマンドは、対象のグラフ、レイアウト、またはコントロールパネルにポリゴンを描画します。

パラメーター

(*xorg*, *yorg*) は、現在アクティブな座標系におけるポリゴンの始点を定義します。

hScaling と *vScaling* は、それぞれ水平方向と垂直方向の拡大率を設定するもので、1 は 100% を意味します。

DrawPoly の *xWaveName*、*yWaveName* バージョンは、それらの X ウェーブと Y ウェーブからデータを取得します。
この接続は維持されるため、いずれかのウェーブに変更が加わると、ポリゴンが更新されます。

DrawPoly コマンドは多次元に対応していません。
詳細は、ヘルプ Multidimensional Waves、特に Analysis on Multidimensional Waves のセクションを参照してください。

頂点のリストをリテラルとして受け取るバージョンの DrawPoly を使うには、1 行目に任意の数の頂点を記述し、その後、すべての頂点を定義するために必要な数の /A バージョンを使います。

フラグ

/A　指定された頂点を、現在開いているポリゴン（描画したばかりのポリゴンまたは現在選択されているポリゴン）に追加します。

/ABS　デフォルトで最初のデータポイントを残りのデータから差し引く処理を無効にします。

/W=*winName*　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

xorg と *yorg* はポリゴンの始点となる頂点の位置を定義するため、各頂点に定数を足したり引いたりしても何の影響もありません。
{*x0*, *y0*, *x1*, *y1* ...} という頂点リストの最初の XY ペアは、*x0* や *y0* の値にかかわらず (*xorg*, *yorg*) に表示されます。
x0 と *y0* は、単に頂点リストの基準点を設定するためのものです。
それ以降の頂点は、(*x0*, *y0*) を基準としています。

精神的な健康を保つために、(*x0*, *y0*) を (0,0) に指定し、それ以降のすべての頂点をその原点からのオフセットとして扱うことをお勧めします。
そうすれば、(*xorg*, *yorg*) でポリゴンの位置が設定され、リスト内のすべての頂点は、その原点を基準とした相対位置になります。

/ABS フラグを指定することで、最初のポイントの差し引きを無効にすることができます。

現在開いているポリゴンの原点とスケールのみを変更し、データを再指定する必要がない場合は、次のように入力してください。

```
DrawPoly xorg, yorg, hScaling, vScaling,{}
```

座標系、ポリゴンの線幅、色、破線パターン、その他のプロパティは、現在の描画環境によって決定されます。ポリゴンは、SetDrawLayer によって指定された、そのウィンドウの現在の描画レイヤーに描画されます。

NaN の座標ペアを追加することで、ポリゴンをセグメントに分割することができます。
詳細は、ヘルプ Segmented Polygons を参照してください。

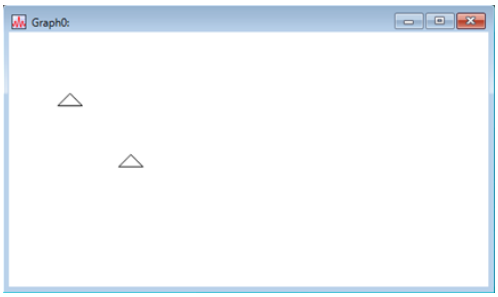
例

ここでは、絶対描画座標（SetDrawEnv を参照）を使って小さな三角形を描画するためのコマンドをいくつか紹介します。

```
Display          // 新しい空のグラフを作成

// 座標 (50, 50) から始まり、スケーリング率 100% で三角形を１つ描画する
SetDrawEnv xcoord= abs,ycoord= abs
DrawPoly 50,50,1,1, {0,0,10,10,-10,10,0,0}

// 同じ大きさ・形の２つ目の三角形を、その右下に描画する
SetDrawEnv xcoord= abs,ycoord= abs
DrawPoly 100,100,1,1, {0,0,10,10,-10,10,0,0}
```



ポリゴンを使った他の例は、ヘルプ Segmented Polygons を参照してください。

参照

SetDrawEnv、SetDrawLayer、DrawBezier、DrawAction、PolygonOp、BezierToPolygon コマンド
ヘルプ DrawPoly and DrawBezier Operations

DrawRect [/W=*winName*] *left*, *top*, *right*, *bottom*

DrawRect コマンドは、*left*、*top*、*right*、*bottom* で定義された矩形内に、対象のグラフ、レイアウト、またはコントロールパネルに矩形を描画します。

フラグ

/W=*winName*　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

座標系、矩形の太さ、色、破線のパターン、その他のプロパティは、現在の描画環境によって決定されます。
矩形は、SetDrawLayer によって指定された、そのウィンドウの現在の描画レイヤーに描画されます。

参照

SetDrawEnv、SetDrawLayer、DrawAction コマンド
ヘルプ Drawing

DrawRRect [/W=*winName*] *left*, *top*, *right*, *bottom*

DrawRRect コマンドは、*left*、*top*、*right*、*bottom* で定義された矩形内に、対象のグラフ、レイアウト、またはコントロールパネルに角が丸い矩形を描画します。

フラグ

/W=*winName*　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

座標系、矩形の丸さ、太さ、色、破線のパターン、その他のプロパティは、現在の描画環境によって決定されます。
矩形は、SetDrawLayer によって指定された、そのウィンドウの現在の描画レイヤーに描画されます。

参照

SetDrawEnv、SetDrawLayer、DrawAction コマンド
ヘルプ Drawing

DrawText [/W=*winName*] *x0*, *y0*, *textStr*

DrawText コマンドは、指定されたテキストをターゲットのグラフ、レイアウト、またはコントロールパネルに描画します。
テキストの位置は、(*x0*, *y0*) と SetDrawEnv によって設定された現在の *textxjust*、*textyjust*、*textrot* の設定によって決定されます。

フラグ

/W=*winName*　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

詳細

座標系、テキストのフォント、サイズ、スタイル、その他のプロパティは、現在の描画環境によって決定されます。
テキストは、SetDrawLayer によって指定された、そのウィンドウの現在の描画レイヤーに描画されます。

参照

SetDrawEnv、SetDrawLayer、DrawAction コマンド
ヘルプ Drawing

DrawUserShape [/W=*winName* /MO=*options*] *x0*, *y0*, *x1*, *y1*, *userFuncName*, *textString*, *privateString*

DrawUserShape コマンドは、形状が描画される時に実行されるユーザー定義関数によって形状が定義される点を除けば、組み込みの描画コマンドと同様です。

DrawUserShape は Igor Pro 7.0 で追加されました。

パラメーター

長方形の形状の場合、(*x0*, *y0*) は現在アクティブな座標系における左上隅を、(*x1*, *y1*) は右下隅を定義します。
線状の形状の場合、(*x0*, *y0*) は線の始点を、(*x1*, *y1*) は終点を指定します。

userFuncName は、組み込みの描画コマンドを使って図形を定義し、その情報を提供するユーザー定義関数を指定します。

textString は、図形の上に描画するテキストを指定します。
テキストが不要な場合は "" を指定します。
エスケープコードを使うことで、テキストのフォント、サイズ、スタイル、色を変更できます。
詳細は、ヘルプ Annotation Escape Codes を参照してください。

privateString は、ユーザー定義関数が任意の目的で使うテキストまたはバイナリデータです。
通常、状態情報の保持に使われ、バイナリをサポートするメソッドを使用して再作成マクロに保存されます。
これが必要ない場合は、"" を指定してください。

既存の図形の変更をサポートするため、上記の最後の 3 つのパラメーターはそれぞれ *NoChange* に設定できます。
変更を行わない場合は、*x0*、*y0*、*x1*、*y1* の各パラメーターをすべてゼロに設定します。
既存の図形がない状態で *NoChange* を使うとエラーとなります。
既存の図形を対象とするには、描画ツールでその図形を選択しておく必要があります。

フラグ

/W=*winName*　指定されたウィンドウまたはサブウィンドウに描画します。省略された場合、操作はアクティブなウィンドウまたはサブウィンドウに対して行われます。プロシージャ、マクロ、またはコマンドラインで使う場合、このフラグは最初に指定する必要があります。

winName を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

/MO=*options*　ユーザー定義の描画関数に渡す整数で、任意の目的で使われます。

詳細

ユーザー定義関数は、以下の形式と構造パラメーターを持つ必要があります。

```
Function MyUserShape(s) : DrawUserShape     // オプションの ": DrawUserShape" は、
                                            // この関数を描画パレットの図形メニューに
    STRUCT WMDrawUserShapeStruct &s         // 追加すべきであることを示す

    Variable returnValue= 1

    StrSwitch(s.action)
        case "draw":
```

```
            Print "Put DrawXXX commands here"
            break

        case "drawSelected":
            Print "Put Draw commands when selected (may be same as normalDraw) here"
            break

        case "hitTest":
            Print "Put code to test if mouse is over a control point here"
            // SetCursor を設定し、必要に応じて cursorCode も設定する
            // コントロールポイントを通過していない場合、または動作モードでボタンの操作を開始したくない
            // 場合は、returnValue を 0 に設定します
            break

        case "mouseDown":
            if( s.operateMode)
                Print "Mouse down in button mode."
            else
                // hitTest と類似または同一のコード
                Print "User is starting a drag of a control point."
            endif
            // 処理や再描画が必要ない場合は、returnValue を 0 に設定する
            break

        case "mouseMove":
            if (s.operateMode)
                Print "Roll-over mode and mouse is inside shape."
            else
                Print "Mouse has been captured and user is dragging a control point."
            endif
            // 処理や再描画が必要ない場合は、returnValue を 0 に設定する
            break

        case "mouseUp":
            if (s.operateMode)
                Print "Mouse button released."
            else
                Print "User finished dragging control point."
            endif
            break

        case "getInfo":
            Print "Igor is requesting info about this shape."
            s.textString= "category for shape"
            s.privateString= "shape name"
            s.options= 0            // またはビットを 0, 1, 2 に設定する
            break
    EndSwitch

    return returnValue
End
```

WMDrawUserShapeStruct は、以下のメンバを持つ組み込み構造体です。

メンバー	**説明**
char action[32]	入力：要求されるアクションを指定します。
Int32 options	入力：/MO フラグからの値です。 出力：アクションが getInfo の場合、ビットを次のように設定します： 図形を単純な線のように振る舞わせる場合は、ビット 0 をセットします。端点のサイズを変更すると、リアルタイムで更新されます。

形状をボタンのように動作させたい場合は、ビット 1 をセットします。通常動作モードでは、マウスダウンイベントが発生します。

ビット 2 をセットすると、ロールオーバー動作が有効になります。hitTest が実行され、1 が返された場合、マウスがキャプチャされます。

Int32 operateMode　入力：0 の場合は形状を編集中、1 の場合は通常動作モード（getInfo の実行時にオプションビット 1 または 2 が設定されていた場合のみ）。

PointF mouseLoc　入力：正規化座標系におけるマウスの位置です。

Int32 doSetCursor　出力：アクションが hitTest の場合、true を設定して次のカーソル番号を使います。ロールオーバーモードでの mouseMoved にも使われます。

Int32 cursorCode　出力：アクションが hitTest であり、doSetCursor が設定されている場合、これを目的のカーソル番号に設定します。

double x0,y0,x1,y1　入力：図形の外接長方形の座標です。

RectF objectR　入力：図形の外接長方形の座標（デバイス単位）です。

char winName[MAX_HostChildSpec+1]
入力：ホストのサブウィンドウへのフルパスです。

座標系に関する情報：

Rect drawRect　デバイス座標系で矩形を描画します。

Rect plotRect　グラフにおいて、これはプロット領域です。

Rect axRect　グラフでは、これは軸の突出分を含んだプロット領域です。

char xcName[MAX_OBJ_NAME+1]
X 座標系の名前です。軸名である場合もあります。

char ycName[MAX_OBJ_NAME+1]
Y 座標系の名前です。軸名である場合もあります。

double angle　入力：回転角度です。テキストを表示する時に使います。

String textString　入力：使うか、無視するか、です。getInfo の特別な出力です。

String privateString　入力と出力：Igor によって管理されますが、ユーザー関数によって定義されます。バイナリである場合もあります。getInfo 用の特別な出力があります。

char 配列のサイズ決定に使われる定数である MAX_HostChildSpec と MAX_OBJ_NAME は、Igor Pro 内部で定義されており、将来のバージョンで変更される可能性があります。

draw と drawSelected アクションで指定する描画コマンドでは、(0,0) から (1,1) までの範囲の正規化座標を使う必要があります。
例えば、長方形を描画するには、次のように指定します。

```
DrawRect 0,0,1,1
```

通常、色や塗りつぶしモードなどの描画環境パラメーターは設定せず、Rectangle などの組み込み図形の場合と同様に、図形のユーザーに任せることになります。

SetDrawEnv や DrawRect などの描画コマンドには、特定のウィンドウまたはサブウィンドウに対して操作を行うよう指定する /W フラグがあります。
ただし、DrawUserShape 関数から呼び出された場合、/W フラグは無視され、すべての描画コマンドは、その図形を含むウィンドウまたはサブウィンドウに対して実行されます。

ユーザーが矢印描画ツールで図形を選択すると、drawSelected アクションが呼び出されます。
単純な図形の場合は、draw アクションと同じコードを使用できますが、図形を編集可能にしたい場合は、黄色い点などのコントロールポイントを描画することができます。
その場合、hitTest アクションや mouseDown アクション内に、ユーザーがコントロールポイントの上を移動したか、クリックしたかを判定するコードを記述します。
例えば、「User Draw Shapes」デモエクスペリメント内（メニュー File→Example Experiments→Programming→User Draw Shapes）の「FatArrows」プロシージャファイルを参照してください。

この関数は、getInfo アクションに対して、textString フィールドにカテゴリ名を、privateString フィールドに図形名を設定して応答する必要があります。
カテゴリ名には「Arrows」、図形名には「Right Arrow」などを指定できます。
これらの名前は、描画ツールパレットの「ユーザー図形」アイコンを右クリックした時に表示されるメニューの内容として使われます。
また、options フィールドのビットを設定することで、特定の特殊アクションを有効にすることもできます。
例えば、「User Draw Shapes」エクスペリメントの「LineCallout」と「button procedure」ウィンドウを参照してください。

ユーザー定義関数は、Igor Pro をクラッシュさせずに中断できない描画処理中に実行されるため、その実行中はデバッガーを呼び出すことができません。
したがって、この関数内に設定されたブレークポイントは無視されます。
代わりに、ヘルプ Debugging With Print Statements に記載の方法を使ってください。

グラフやレイアウトでボタン状の形状をサポートするため、Igor Pro 7.0 では「Overlay」という名前の描画レイヤーが追加されました。
このレイヤーは他のすべての要素の上に描画され、印刷やエクスポートの対象にはなりません。
オーバーレイ描画レイヤー内のオブジェクトは、既存のレイヤーのいずれかを使った場合のようにウィンドウ全体を再描画することなく更新できます。
一貫性を保つため、このレイヤーはコントロールパネルでも利用可能です。

参照

SetDrawEnv、SetDrawLayer、DrawBezier、DrawPoly、DrawAction コマンド
ヘルプ Drawing

DSPDetrend [/F=*function* /M=*maskWave* /P=*polyOrder* /Q] *srcWave*

DSPDetrend コマンドは、srcWave のデータに対して指定された関数を最善の近似として適用することで定義されたトレンドを、*srcWave* から除去します。

フラグ

/A　フィッティングを実行する前に、*srcWave* の平均値を差し引きます。
Igor Pro 7.0 で追加されました。

/DEST=*destWave*　次元に応じて W_Detrend または M_Detrend を置き換えるコマンドによって生成される主なウェーブを指定します。

/F= *function*　この関数は、組み込みのカーブフィッティング関数の１つです。

gauss, lor, exp, dblexp, sin, line, poly (requires /P), hillEquation, sigmoid, power, lognormal, poly2d (requires /P), gauss2d.

関数が指定されていない場合、*srcWave* が 1D であれば line が、2D であれば poly2d がデフォルトとして使われます。

/FREE　*destWave* をフリーウェーブとして作成します。

/FREE は関数内でのみ使用可能であり、かつ /DEST で指定された *destWave* が単純な名前またはウェーブ参照構造体のフィールドである場合に限り使用できます。

詳細は、ヘルプ Free Waves を参照してください。

/FREE フラグは、Igor Pro 10.0 で追加されました。

/M=*maskWave* トレンド除去は、*maskWave* においてゼロ以外の値を持つデータポイントにのみ影響します。なお、*maskWave* は *srcWave* と同じ次元数でなければなりません。

/P=n poly または poly2d 関数の多項式の次数を指定します（詳細は CurveFit コマンドを参照してください）。

1D poly 関数と併用する場合、*n* は多項式の項数を指定します。

デフォルトでは、1D の場合、*n*=3、poly2d の場合は *n*=1 となります。

/Q エラー報告がありません。サイレントモードです。

詳細

DSPDetrend は、処理が成功した場合に V_flag を 0 に設定します。
失敗した場合は、-1 に設定されるか、カーブフィッティングルーチンからのエラーコードが格納されます。
結果は、現在のデータフォルダー内の W_Detrend（1D 入力の場合）または M_Detrend（2D 入力の場合）という名前のウェーブファイルに保存されます。
現在のデータフォルダー内に同名のウェーブファイルがすでに存在する場合、そのファイルは上書きされます。

参照

V_FitQuitReason と組み込みのフィッティング関数に関する詳細は、CurveFit コマンドを参照してください。

DSPPeriodogram [/DB /DBR=*ref* /COHR /DLSG /NODC=*val* /NOR=*value* /Q /SEGN={*ptsPerSegment*, *overlapPts*} /R=[*start*, *end*] /R=(*startX*, *endX*) /WIN=*windowKind* /Z] *srcWave* [, *srcWave2*]

DSPPeriodogram コマンドは、入力ウェーブのピリオドグラム、クロススペクトル密度、またはコヒーレンス度を計算します。
コマンドの結果は、現在のデータフォルダー内のウェーブ W_Periodogram または /DEST フラグを使って指定したウェーブに保存されます。

クロススペクトル密度やコヒーレンス度を計算するには、オプションの *srcWave2* パラメーターを使って 2 番目のウェーブを指定する必要があります。
この場合、W_Periodogram は複素数となり、/DB と /DBR フラグは適用されません。

フラグ

/DB 結果を、最大値を基準として dB 単位で表します。

/DBR=*ref* 指定された基準値を使って、結果を dB 単位で表示します。

/COHR コヒーレンスの度合いを計算します。このフラグは、入力が 2 つのウェーブから構成される場合に適用されます。

/DEST=*destWave* コマンドによって生成される出力ウェーブを指定します。

/DEST フラグは、Igor Pro 8.0 で追加されました。

srcWave と *destWave* の両方に同じウェーブを指定するのはエラーです。

関数内で使う場合、DSPPeriodogram コマンドはデフォルトで、宛先ウェーブに対する実数ウェーブ参照を作成します。
詳細は、ヘルプ Automatic Creation of WAVE References を参照してください。

/DLSG　複数のセグメントを使ってピリオドグラム、クロススペクトル密度、またはコヒーレンス度を計算する場合、デフォルトでは、必要に応じて最後のセグメントの末尾にゼロが埋められます。このフラグを指定すると、不完全な最後のセグメントは除外され、計算には含まれません。

/DTRD　Detrends は、ウィンドウ関数で乗算する前に、各セグメントの線形回帰値を差し引くことでセグメントを処理します。/DTRD はセグメントに影響を与えるため、/NODC=1 とは互換性がありません。/DTRD は Igor Pro 8.0 で追加されました。

/FREE　*destWave* をフリーウェーブとして作成します。

/FREE は関数内でのみ使用可能であり、かつ /DEST で指定された *destWave* が単純な名前またはウェーブ参照構造体のフィールドである場合に限り使用できます。

詳細は、ヘルプ Free Waves を参照してください。

/FREE フラグは、Igor Pro 10.0 で追加されました。

/NODC=*val*　DC 項を抑制します。

val=0：　FFT を使って DC 項を計算します（デフォルト）。

val=1：　処理前とウィンドウ関数を適用する前の信号の平均値を差し引くことで、DC 成分を除去します（以下の /WIN を参照）。

val=2：　DC 項を FFT 配列の 2 番目の項と等しく設定することで、DC 項を抑制します。

/NOR=*N*　ピリオドグラム方程式における正規化係数 N を設定します。デフォルトでは、データポイントの数にウィンドウ関数（ある場合）の二乗ノルムを掛けた値となります。

N=0 または 1： デフォルトの正規化をスキップします。

N のその他の値は、唯一の正規化として使われます。

/PARS　ウィンドウ関数を使う場合でも、Parseval の定理を満たすように正規化を設定します。

/PARS フラグは Igor Pro 8.0 で追加されました。このフラグは /NOR フラグの設定を上書きします。

詳細は、以下の「Parseval の定理を満たす正規化」を参照してください。

/Q　静寂モードです。履歴エリアへの表示を抑制します。

/R=[*startPt*, *endPt*]
　指定されたウェーブの範囲について、ピリオドグラムを計算します。startPt と endPt は、ソースウェーブのポイント番号で表されます。

/R=(*startX*, *endX*)
　指定されたウェーブ数範囲のピリオドグラムを計算します。startX と endX は X 値で指定されます。このオプションでは、指定された X 値がポイント番号に変換されるため、丸め誤差が生じる場合があることに注意してください。

/SEGN={*ptsPerSegment* , *overlapPts*}
　このフラグを使うと、入力ウェーブから抽出した複数のセグメントを平均化して、ピリオドグラム、クロススペクトル密度、またはコヒーレンス度を計算できます。各区間の

サイズは *ptsPerSegment* です。*overlapPts* は、各区間の末尾から次のセグメントに含まれるポイント数を指定します。

/WIN=*windowKind*

/W フラグを省略した場合、このコマンドでは、全ウェーブまたは /R フラグで指定されたデータ範囲に対して長方形のウィンドウが使われます。

以下の定義において、*w(n)* は信号に乗算されるウィンドウ関数の値、*N* は信号ウェーブのデータポイント数（/R が指定されている場合は範囲）、*n* はウェーブのデータポイントインデックスです。

windowKind の選択肢は以下の通りです。

Bartlett, Blackman367, Blackman361, Blackman492, Blackman474, Cos1, Cos2, Cos3, Cos4, Hamming, Hanning, KaiserBessel20, KaiserBessel25, KaiserBessel30, Parzen, Poisson2, Poisson3, Poisson4, Riemann

ウィンドウの方程式や詳細は、FFT コマンドを参照してください。

/Z　エラーを報告しません。エラーが発生すると、V_flag は -1 に設定されます。

詳細

デフォルトのピリオドグラムは次のように定義されます。

$$Periodogram = \frac{\left|F(s)^2\right|}{N}$$

ここで、F(s) は信号 s のフーリエ変換、N はポイント数です。

ほとんどの実用的な状況では、（フーリエ変換を計算する時に）次のような形式をとるウィンドウ関数の使用を考慮に入れる必要があります。

$$Periodogram = \frac{\left|F(s \cdot w)^2\right|}{N_p N_w}$$

ここで、w はウィンドウ関数、Np はポイントの数、Nw はウィンドウ関数の正規化係数です。

信号を（任意のオーバーラップをもって）複数のセグメントに分割し、すべてのセグメントの結果を平均してピリオドグラムを計算する場合、ピリオドグラムの式は次のようになります。

$$Periodogram = \frac{\sum_{i=1}^{M}\left|F(s_i \cdot w)^2\right|}{M N_p N_w}$$

ここで、s_i は i 番目の区間 s、N_s は区間あたりのポイントの数、M は区間の数を表します。

２つのウェーブ s_1 と s_2 のクロススペクトル密度（csd）を計算すると、その結果として複素数値のウェーブが得られます。

$$csd = \frac{F(S_A)[f(S_B)]^*}{N}$$

これは、第１のウェーブ S_A のフーリエ変換と、第２のウェーブ S_B のフーリエ変換の複素共役との正規化された積を含みます。
CSD 計算をセグメント平均化に拡張すると、以下の形となります。

$$csd = \frac{\sum_{i=0}^{M} F\left(S_{A_i}\right)\left[f\left(S_{B_i}\right)\right]^*}{M N_s N_w}$$

ここで、S_{Ai} は第 1 ウェーブの i 番目のセグメント、M はセグメントの数、N_s は 1 つのセグメントに含まれるポイントの数です。

コヒーレンスの度合いは、クロススペクトル密度の正規化された値です。
これは次式で与えられます。

$$\gamma = \frac{\sum_{i=0}^{M} F(s_{Ai})[F(s_{Bi})]^*}{\sqrt{\sum_{i=0}^{M} F(s_{Ai})[F(s_{Ai})]^* \sum_{i=0}^{M} F(s_{Bi})[F(s_{Bi})]^*}}$$

コヒーレンスの度合いにおけるバイアスは、以下の近似式を使って計算されます。

$$B = \frac{1}{M}[1 - |\gamma|^2]^2$$

バイアスはウェーブ W_Bias に格納されています。

/SEGN フラグを使うと、実際のセグメント数が変数 V_numSegments に出力されます。

DSPPeriodogram はウェーブの次元性を検証するものではなく、ウェーブを 1D として扱うことに注意してください。
クロススペクトル密度やコヒーレンス度を計算する時は、 2 つのウェーブの数型、次元、およびスケーリングが一致している必要があります。

Parseval の定理を満たす正規化

/PARS フラグを指定して DSPPeriodogram を実行した後、この関数を使って、正規化が Parseval の定理を満たしていることを確認できます。

```
Function CheckNormalization(srcWave, periodogramWave)
    Wave srcWave                              // 実数値の時系列
    Wave periodogramWave                      // 例： W_Periodogram

    Duplicate/FREE periodogramWave,wp         // オリジナルを保持
    wp[0]/=2                                  // 0 ビンを修正
    wp[numpnts(wp)-1] /=2                     // Nyquist ビンを修正
    MatrixOP/FREE w2=magsqr(srcWave)/numPoints(srcWave)
    Print sum(wp), sum(w2)                    // Parseval: これらは等しくなるはず
End
```

参照

2D ウィンドウ処理アプリケーション向けの ImageWindow コマンド。
ウィンドウ方程式と詳細は FFT コマンドを参照してください。

Hanning、LombPeriodogram、MatrixOp コマンド

参考文献

ウィンドウ関数の使用に関する詳細は、以下を参照してください。

Harris, F.J., "On the use of windows for harmonic analysis with the discrete Fourier Transform", Proc, IEEE, 66, 51-83, 1978.

G.C. Carter, C.H. Knapp and A.H. Nuttall, The Estimation of the Magnitude-squared Coherence Function Via Overlapped Fast Fourier Transform Processing, IEEE Trans. Audio and Electroacoustics, V. AU-21, (4) 1973.

Duplicate [/FREE[=*nm*] /O /R=(*startX*,*endX*)[(*startY*, *endY*)...]]
[*typeFlags*] *srcWaveName*, *destWaveName* [, *destWaveName*]...

WaveMetrics Web サイトの日本語のコマンドのヘルプセクションにある別ファイルを参照してください。

DuplicateDataFolder [/O=*options* /Z] *sourceDataFolderSpec*,
destDataFolderSpec

DuplicateDataFolder コマンドは、ソースのデータフォルダーとその中にあるすべてのファイルのコピーを作成し、そのコピーを指定された場所に指定された名前で配置します。

パラメーター

sourceDataFolderSpec には、現在のデータフォルダー内の子データフォルダーの名前、部分パス（現在のデータフォルダーを基準とした相対パス）と名前、絶対パス（ルートから始まる）と名前、あるいはユーザー定義関数内のデータフォルダー参照（DFREF）を指定できます。

destDataFolderSpec には、データフォルダー名のみ、データフォルダー名を含む部分パス（現在のデータフォルダーを基準とした相対パス）、またはデータフォルダー名を含む絶対パス（ルートから始まる）を指定できます。
データフォルダー名のみを指定した場合、新しいデータフォルダーは現在のデータフォルダー内に作成されます。また、ユーザー定義関数内では、データフォルダー参照（DFREF）が作成されます。

使用する構文によっては、DuplicateDataFolder は *destDataFolderSpec* で指定されたデータフォルダーを上書きする場合もあれば、ソースデータフォルダーを *destDataFolderSpec* で指定されたデータフォルダーにコピーする場合もあります。
詳細は、以下の「DuplicateDataFolder の宛先」のセクションを参照してください。

フラグ

/O=*options*　宛先のデータフォルダーがすでに存在する場合、そのフォルダーを上書きします。

/O フラグは、Igor Pro 8.0 で追加されました。

options=0：データフォルダーを上書きしようとすると、あたかも /O を省略したかのようにエラーが発生します。

options=1：宛先のデータフォルダーを完全に上書きします。宛先のデータフォルダーが存在する場合、DuplicateDataFolder はまず、可能であればそれを削除します。例えば、使用中のウェーブが含まれているなどの理由で削除できない場合、DuplicateDataFolder はエラーを発生させます。削除に成功した場合、DuplicateDataFolder はソースを宛先にコピーします。

options=2：ソースデータフォルダーの内容を、宛先データフォルダーに統合します。ソースデータフォルダー内の項目が宛先データフォルダーにすでに存在する場合、DuplicateDataFolder はその項目を上書きします。それ以外の場合は、その項目をコピーします。ソースデータフォルダーに存在しない宛先データフォルダー内の項目は、宛先データフォルダーに残ります。

options=3：ソースデータフォルダーを宛先データフォルダーにマージし、可能であれば、ソースデータフォルダーに存在しない項目を削除します。ソースデータフォルダー内の項目が宛先データフォルダーに存在しない場合、DuplicateDataFolder はその項目を宛先データフォルダーから削除しようとします。使用中のため削除できない場合でも、エラーは発生しません。

/Z　　エラーは致命的ではありません。エラーが発生しても、プロシージャの実行は継続されます。エラーが発生したかどうかは、出力変数 V_flag を確認することで確認できます。

詳細

宛先がソースデータフォルダー内に含まれている場合、エラーが発生します。

データフォルダー参照変数の一般的な用途として、空きデータフォルダー内にデータフォルダーのコピーを作成することが挙げられます。

```
DFREF dest = NewFreeDataFolder()
DuplicateDataFolder source, dest        // フリーデータフォルダー内のソースのコピー
```

DuplicateDataFolder の宛先

使用する構文によっては、DuplicateDataFolder は *destDataFolderSpec* で指定されたデータフォルダーを上書きする場合もあれば、ソースデータフォルダーを *destDataFolderSpec* で指定されたデータフォルダーにコピーする場合もあります。
これを説明するために、以下のコマンドを実行したと仮定します。

```
KillDataFolder/Z root:
NewDataFolder/O root:DataFolderA
NewDataFolder/O root:DataFolderA:SubDataFolderA
NewDataFolder/O root:DataFolderB
```

これにより、次のようなデータ階層が得られます。

```
root
    DataFolderA
        SubDataFolderA
    DataFolderB
```

以下のコマンドは、そのデータ階層から始まる特定の構文に対して、DuplicateDataFolder がどのような処理を行うかを示しています。

```
// 1. 末尾にコロンがないリテラル dest の場合：
// DataFolderA のコピーを DataFolderB という名前で作成し、DataFolderB を上書きします
DuplicateDataFolder/O=1 root:DataFolderA, root:DataFolderB

// 2. 末尾にコロンが付いたリテラル dest の場合：
// DataFolderA の内容を DataFolderB にコピーします
DuplicateDataFolder/O=1 root:DataFolderA, root:DataFolderB:

// 3. 子フォルダー名を明示したリテラル dest の場合：
// DataFolderA を Child という名前で DataFolderB にコピーします
DuplicateDataFolder/O=1 root:DataFolderA, root:DataFolderB:Child

// 4. 末尾にコロンを付けない DFREF dest の場合：
// DataFolderA の内容を DataFolderB にコピーします
DFREF dest = root:DataFolderB
DuplicateDataFolder/O=1 root:DataFolderA, dest

// 5. 末尾にコロンが付いた DFREF dest の場合：
// エラーが発生します
DFREF dest = root:DataFolderB
DuplicateDataFolder/O=1 root:DataFolderA, dest:    // Error - trailing colon not allowed
```

```
// 6.  子フォルダー名を明示した DFREF dest の場合：
// DataFolderA を、Child という名前で DataFolderB にコピーします
DFREF dest = root:DataFolderB
DuplicateDataFolder/O=1 root:DataFolderA, dest:Child
```

出力変数

DuplicateDataFolder は、以下の出力変数を設定します。

V_flag　　コマンドが成功した場合は 0、宛先のデータフォルダーがすでに存在していた場合は -1、それ以外の場合は 0 以外のエラーコードが返されます。
出力変数 V_Flag は、Igor Pro 8.0 で追加されました。

例

```
DuplicateDataFolder root:DF0,root:DF0Copy          // DF0 のコピーを DF0Copy という名前で作成する
```

参照

MoveDataFolder コマンド
ヘルプ Data Folders、Data Folder References、Free Data Folders

DWT [/I/S/D/P=*num* /T=*type* /N=*num* /V=*value*] *srcWaveName*, *destWaveName*

DWT コマンドは、入力ウェーブ *srcWaveName* に対して離散ウェーブレット変換を実行します。
このコマンドは、各次元の要素数が 2 の冪である場合、または /P フラグが指定されている場合に限り、1 次元以上に対して動作します。

フラグ

/D　　ソースウェーブのノイズを除去します。指定されたウェーブ変換を順方向に行います。その後、変換係数の絶対値が、変換の最大絶対値の % 未満であるものすべてをゼロに設定します（この値は /V フラグで指定されます）。続いて逆変換を行い、その結果を *destWaveName* に格納します。/I フラグは /D フラグと互換性がありません。

/FREE　　*destWaveName* をフリーウェーブとして作成します。

/FREE は関数内でのみ使用可能であり、かつ *destWaveName* が単純な名前またはウェーブ参照構造体のフィールドである場合に限り使用できます。

詳細は、ヘルプ Free Waves を参照してください。

/FREE フラグは、Igor Pro 10.0 で追加されました。

/I　　逆ウェーブレット変換を実行します。/S と /D フラグは、/I フラグと併用できません。

/N=*num*　　ウェーブレット係数の数を指定します。サポートされている組み合わせについては、「詳細」セクションの表を参照してください。

/P=*num*　　*num*=1：　*srcWaveName* の指定された次元のデータ要素数が 2 の冪でない場合、その次元の末尾に、最も近い 2 の冪になるまで 0 を埋めます。

num=2：　変換の計算にはゼロパディングを使いますが、結果として得られるウェーブは入力ウェーブの長さに切り詰められます

/S　　ソースウェーブを平滑化します。これは、指定されたウェーブレット変換を順方向に行います。その後、0 からカットオフ値（/V フラグで%単位で指定）までの範囲にある係数を除く、すべての変換係数を 0 に設定します。続いて、逆変換を行い、その結果を *destWaveName* に格納します。/I フラグは /S フラグと互換性がありません。

/T=*type*　　*type* で指定されたウェーブレット変換を実行します。各変換のタイプコードに対応する変換名と /N フラグで使われる *num* パラメーターの許容値については、「詳細」セクションの表を参照してください。

/V=*value*　　/S と /D フラグでのみ、平滑化の程度を指定します。/S の場合、この値は、その値を超えるデータポイントの割合（%）をカットオフ値として指定し、その値を超える係数はゼロに設定されます。/D の場合、この値は変換の最大振幅に対する割合（%）を指定し、この値未満の係数はゼロに設定されます。

詳細

/T または /N フラグを使う場合、指定された変換に対しては、*type* と *num* の以下の組み合わせが許可されます。

変換	**/T=type**	**/N=num**
Daubechies	1 (デフォルト)	4,6,8,10,12,20
Haar	2	NA
Battle-Lemarie	4	NA
Burt-Adelson	8	NA
Coifman	16	2, 4, 6
Pseudo-Coifman	32	NA
Splines	64	1 (2-2), 2 (2-4), 3 (3-3), 4 (3-7)

destWaveName が存在する場合、DWT はそれを上書きします。
存在しない場合は、DWT がそれを作成します。

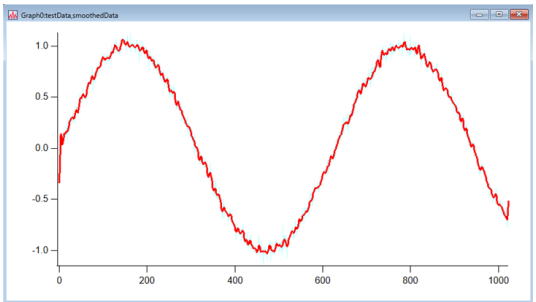
関数内で使う場合、DWT コマンドにより、宛先ウェーブに対するウェーブ参照が自動的に作成されます。
詳細は、ヘルプ Automatic Creation of WAVE References を参照してください。

destWaveName が指定されていない場合、この操作では、1D ウェーブの場合は W_DWT に、それ以上の次元の場合は M_DWT に結果が格納されます。

1D ウェーブを扱う場合、変換結果は、各配列の上半分に詳細成分が、下半分に平滑化成分が格納されるようにパッキングされ、各スケールは下側の要素に順次格納されます。
例えば、ソースウェーブが 128 ポイントを含む場合、最も低いスケールの結果は要素 64～127 に格納され、次のスケール（2 の累乗）は 32～63 に、その次のスケールは 16～31 に、というように格納されます。

例

```
Make/O/N=1024 testData=sin(x/100)+gnoise(0.05)
DWT /S/N=20/V=25 testData, smoothedData
Display testData, smoothedData
ModifyGraph rgb(testData)=(49151,65535,65535)
ModifyGraph lsize(smoothedData)=2
```



参照

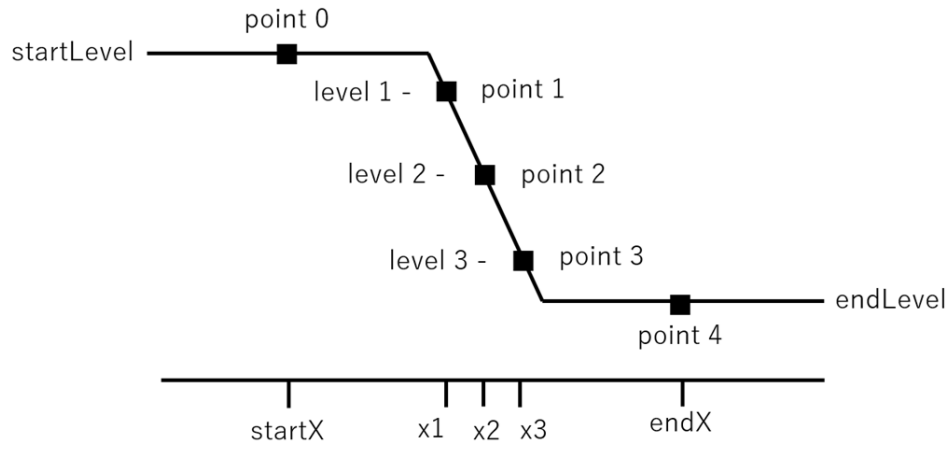
連続ウェーブレット変換を行うには、CWT コマンドを使います。
FFT コマンド

デモ

メニュー File→Example Experimentes→Analysis→Wavelet Demo 2.0

EdgeStats [/A=*avgPts* /B=*box* /F=*frac* /L=(*startLevel*, *endLevel*) /P /Q /R=(*startX*, *endX*) /T=*dx*] *waveName*

EdgeStats コマンドは、単一のエッジが含まれていると予想されるウェーブの領域について、簡単な統計情報を算出します。
複数のエッジが存在する場合、EdgeStats は最初に検出したエッジに対して処理を行います。



フラグ

/A=*avgPts*　*startX* と *endX* を中心とする *avgPts* 個のポイントの平均値を算出し、*startLevel* と *endLevel* を自動的に決定します。デフォルトは /A=1 です。

/B=*box*　移動平均のボックスサイズを設定します。この値は奇数である必要があります。/B=box が省略された場合、または *box* が 1 の場合、平均化は行われません。

/F=*frac*　レベル 1、2、3 を (*endLevel* - *startLevel*) の分数として指定します。

level1 = *frac* * (*endLevel* - *startLevel*) + *startLevel*

level2 = 0.5 * (*endLevel* - *startLevel*) + *startLevel*

level3 = (1 - frac) * (endLevel - startLevel) + startLevel

frac のデフォルト値は 0.1 であり、これにより level1 は 10% レベル、level2 は 50% レベル、level3 は 90% レベルとなります。

frac は 0 から 0.5 の間でなければなりません。

/L=(*startLevel*, *endLevel*)
startLevel と *endLevel* を明示的に設定します。省略した場合は、自動的に決定されます。/A を参照してください。

/P　出力エッジの位置（下記参照）は、ポイント番号として返されます。/P を省略した場合、エッジの位置は X 値として返されます。

/Q　履歴に結果が出力されるのを防ぎ、エッジが見つからない場合のエラーを防止します。

/R=(*startX*, *endX*)　検索対象のウェーブの X 座標範囲を指定します。*startX* と *endX* を入れ替えることで、検索方向を逆にすることもできます。

/R=[*startP*, *endP*]　検索対象のウェーブの X 座標範囲を指定します。*startX* と *endX* を入れ替えることで、検索方向を逆にすることもできます。

/T=*dx*　より正確な結果が得られる可能性があるため、検索は 2 方向で行われます。*dx* は、2 回目の検索が開始される位置をコントロールします。

詳細

/B=box、/T=dx、/P、/Q フラグの動作は、FindLevel コマンドの場合と同じです。

EdgeStats は、*startX* と *endX* と呼ばれる 2 つの X 座標間の入力ウェーブの領域を対象とします。
startX と *endX* は、/R=(*startX*, *endX*) フラグによって設定されます。
このフラグが指定されていない場合、*startX* と *endX* はデフォルトでウェーブ全体の開始ポイントと終了ポイントになります。
startX は *endX* より大きい値に設定できるため、エッジの検索を「右」から「左」へと進めることができます。

上の図は、ウェーブの「左側」（ポイント番号が小さい方）から「右側」（ポイント番号が大きい方）へ向かう、デフォルトの検索方向を示しています。

startLevel と *endLevel* の値は、エッジのベースレベルを定義します。
これらのレベルは、/L=(*startLevel*, *endLevel*) フラグを使って明示的に設定することも、/A=avgPts フラグを使って *startX* と *endX* 周辺のポイントの平均値を算出し、EdgeStats にベースレベルを自動的に算出させることもできます。

startLevel、*endLevel*、*frac* 値（/F=frac フラグを参照）が指定されると、EdgeStats は上の図に示すように level1、level2、level3 を定義します。

これらのレベルが定義されると、EdgeStats は *startX* から *endX* までウェーブを検索し、level2 を探します。
level2 が見つかったら、次に *level1* と *level3* を検索します。
結果は、以下で説明する変数を通じて返されます。

EdgeStats は以下の変数を設定します。

V_flag　0：　3 か所すべてのレベル交差が確認された
1：　1 か所か 2 か所のレベル交差が見つかった
2：　レベル交差は見つからなかった

V_EdgeLoc1　level1 が発見された X の位置

V_EdgeLoc2　level2 が発見された X の位置

V_EdgeLoc3　level3 が発見された X の位置

V_EdgeLvl0　*startLevel* の値

V_EdgeLvl1　level1 の値

V_EdgeLvl2　level2 の値

V_EdgeLvl3　level3 の値

V_EdgeLvl4　*endLevel* の値

V_EdgeAmp4_0　エッジの振幅（*endLevel* - *startLevel*）

V_EdgeDLoc3_1　エッジの幅（ポイント 1 とポイント 3 の間の X の距離）

V_EdgeSlope3_1　エッジの傾き（ポイント 1 からポイント 3 までの直線の傾き）

これらの X 座標および距離は、/P フラグを使わない限り、指定されたウェーブの X スケーリングに基づいて表されます。
/P フラグを使う場合は、ポイント番号に基づいて表されます。

EdgeStats コマンドは多次元性を認識しません。
詳細は、Multidimensional Waves、特に Analysis on Multidimensional Waves を参照してください。

参照

/B=box、/T=dx、/P、/Q 各フラグ、PulseStats に関する FindLevel。

Edit [/FG=(*gLeft*, *gTop*, *gRight*, *gBottom*) /HIDE=*h* /HOST=*hcSpec* /I /K=*k* /M/N=*name* /W=(*left*, *top*, *right*, *bottom*)] [*columnSpec* [,*columnSpec*]...] [*as titleStr*]

Edit コマンドを実行すると、指定された列を含むテーブルウィンドウまたはサブウィンドウが作成されます。

パラメーター

columnSpec は通常、ウェーブの名前そのものです。
columnSpec が指定されていない場合は、空のテーブルが作成されます。

列の仕様は、ウェーブの名前と、その後に任意で付加される以下の接尾辞のいずれかです。

.i　インデックス値

.l　次元ラベル

.d　データ値

.id　インデックス&データ値

.ld　次元ラベル&データ値

ウェーブが複素数の場合、ウェーブ名の後に「.real」または「.imag」という接尾辞が付くことがあります。
ただし、Igor Pro 3.0 以降では、実数列と虚数列の両方がテーブルにまとめて追加されるため（一方だけを追加することはできません）、これらの接尾辞の使用は推奨されません。

titleStr は、テーブルのタイトルを含む文字列式です。
指定しない場合、テーブルに表示される列を識別できるタイトルを自動的に生成します。

フラグ

/FG=(*gLeft*, *gTop*, *gRight*, *gBottom*)
ホストウィンドウ内で、サブウィンドウの外枠がどのフレームガイドに取り付けられるかを指定します。

フレームガイドの標準名称は、左、右、上、下のフレームガイドそれぞれについて、FL、FR、FT、FB ですが、ホストで定義されたユーザー定義のガイド名を使うこともできます。* を使うと、デフォルトのガイド名を指定できます。

ガイドは、/W で設定された数値による配置を上書きすることができます。

/HIDE=*h*　ウィンドウを非表示（*h*=1）または表示（*h*=0、デフォルト）にします。

/HOST=*hcSpec*　*hcSpec* で指定されたホストウィンドウまたはサブウィンドウに、新しいテーブルを埋め込みます。

hcSpec を使ってサブウィンドウを特定する時は、ウィンドウ階層の構成に関する詳細について、ヘルプ Subwindow Syntax を参照してください。

/I　　/W の座標がインチ単位であることを指定します。

/K=*k*　　ユーザーがウィンドウを閉じようとした時の動作を指定します。

k=0：　ダイアログ付き（通常）（デフォルト）

k=1：　ダイアログ無しで Kill

k=2：　Kill を無効にする

k=3：　ウィンドウを隠す

k=4：　ダイアログ無しで Kill し、エクスペリメントも保存されない

/K=2 または /K=3 を指定した場合でも、KillWindow コマンドを使ってウィンドウを終了させることができます。

/M　　/W の座標がセンチメートル単位であることを指定します。

/N=*name*　　作成されるテーブルに、その名前がまだ使われていない場合は、この名前を付けるよう要求します。すでに使われている場合は、name0、name1 など順に試して、未使用のウィンドウ名が見つかるまで続けます。関数やマクロ内では、S_name に選択されたテーブル名が設定されます。

/W=(*left*, *top*, *right*, *bottom*)

テーブルを画面上の特定の位置とサイズに配置します。/W の座標は、/W の前に /I または /M が指定されていない限り、ポイント単位で指定されます。

/HOST フラグと組み合わせて使う場合、指定された位置座標には、次の 2 つの意味のいずれかが適用されます。

1. すべての値が 1 未満の場合、座標はホストフレームのサイズに対する割合として扱われます。
2. いずれかの値が 1 より大きい場合、座標は、ホストフレームの左上隅を基準として、ポイント単位、またはコントロールパネルホストの場合はコントロールパネル単位で測定された固定位置とみなされます。

詳細

テーブルに表示されている次元インデックスの値を変更することはできません。
Change Wave Scaling ダイアログまたは SetScale コマンドを使ってください。

/N を使わない場合、Edit はテーブルウィンドウに「Table*n*」という形式の名前（*n* は整数）を自動的に割り当てます。
関数やマクロ内では、割り当てられた名前が S_name 文字列に格納されます。
これは、プロシージャからテーブルを参照する時に使用できる名前です。
グラフの名前を変更するには、RenameWindow コマンドを使ってください。

例

次の例では、ウェーブが 1D であることを前提としています。

```
Edit myWave,otherWave       // 2 列：各ウェーブからのデータ値
Edit myWave.id              // 2 列：x 値とデータ値
Edit cmplxWave              // 2 列：データ値の実部と虚部
Edit cmplxWave.i            // 1 列：x 値
```

以下の例は、ウェーブの名前が文字列変数に含まれている場合、プロシージャ内で列名の接尾辞を使う方法を示しています。

```
Macro TestEdit()
    String w = "wave0"
    Edit $w                 // データ値を編集
    Edit $w.i               // インデックス値を表示
    Edit $w.id              // インデックス値とデータ値
End
```

なお、接尾辞がある場合でも、それを文字列に格納してはならないことに注意してください。
ユーザー定義関数では、構文が若干異なります。

```
Function TestEditFunction()
    Wave w = $"wave0"
    Edit w                  // $ はなし。w はウェーブであり、文字列ではないため
    Edit w.i                // インデックス値を表示
    Edit w.id               // インデックス値とデータ値
End
```

参照

Tables コマンド