CONTENTS

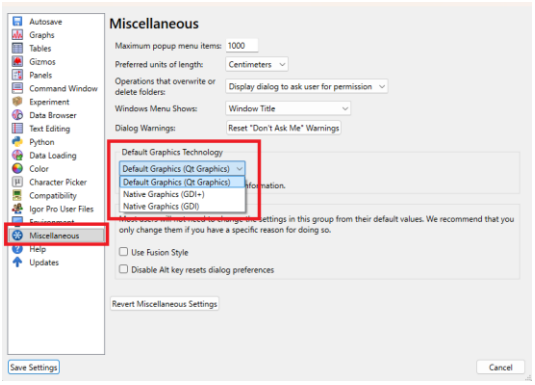
グラフィックスの技術

バージョン 7 以降、Igor Pro はクロスプラットフォームの Qt フレームワークを基盤としており、デフォルトでは Igor はグラフィックスに Qt を使います。
ただし、特別な目的のために、Igor はプラットフォーム固有のグラフィックスへのアクセスを提供します。

可能な限りプラットフォーム固有グラフィックス（ネイティブグラフィックス）は避け、Qt グラフィックスを使うべきです。
これは将来の開発の焦点となるためです。
プラットフォーム固有グラフィックスは主に緊急時の使用を想定して提供されています。

プラットフォーム固有グラフィックスは、グローバルベース（すべてのウィンドウに影響）で以下の 2 つの方法で選択できます：

1. Miscellaneous Settings ダイアログ（Misc→Miscellaneous Settings メニュー）の Miscellaneous カテゴリから選択。



2. 次を実行。

   ```
   SetIgorOption GraphicsTechnology=n
   ```

   ここで、n は次のように解釈されます。

n=0	デフォルト（現在は Qt）
n=1	GDI+
n=2	GDI
n=3	Qt

   他のほとんどの SetIgorOption ケースとは異なり、この変更はディスク上のプレファレンスに保存され、今後の Igor セッションに適用されます。

グローバルなグラフィックス技術の設定の変更に加え、個々のウィンドウを次のように変更できます：

```
SetWindow winName, graphicsTech=n
```

ここで n は上記と同じです。
winName は kwTopWin またはウィンドウの実際の名前になります。
現在、この設定はグラフ専用に保存されますが、今後、変更される可能性があります。

将来的には、Qt グラフィックスが他の技術を置き換えられることも意図しており、SetIgorOption GraphicsTechnology オプションはサポートされなくなる可能性があります。

Windows におけるグラフィックス技術

一般的に、デフォルトのグラフィックス技術（GraphicsTechnology=0）を使うべきです。
これは現在 Qt グラフィックスです。
グラフィックスに問題が発生した場合は、GraphicsTechnology=1（GDI+）または 2（GDI）を試してみてください。
高解像度ディスプレイは Qt グラフィックスでのみサポートされています。

GraphicsTechnology=1 は GDI+ インターフェイスを利用します。
これにより、透明やその他の高度な機能がサポートされます。
残念ながら、GDI+ は、特にテキストのレンダリングにおいて、非常に低速です。

速度が問題となる場合、古い GDI インターフェイスを使うために GraphicsTechnology=2 を試すことができます。
ただし、透明はサポートされません。
アルファ値にかかわらず、色は不透明になります（アルファ値に関する説明は「カラーブレンディング」のセクションを参照）。

GDI+ モードでは、EMF エクスポートは「デュアル EMF」と呼ばれるハイブリッド形式を使います。
この形式には、互換性のために古い GDI と GDI+ の両方が含まれます。
GraphicsTechnology=2 モードでは、古い GDI のみのスタイルのみがエクスポートされます。
Qt グラフィックスモードでは、エクスポート用の EMF のレンダリングに GDI+ が使われ、デュアル EMF 形式が生成されます。
一部のアプリケーションは、EMF+ またはデュアル EMF では正常に動作しません。
その場合は、GraphicsTechnology=2 (旧 GDI モード) を設定して、EMF+ コンポーネントを含まない EMF をエクスポートしてみてください。

当社の経験では、単純な EMF は GDI モードでより良好に描画されます。
Qt グラフィックスでは、貼り付ける画像の内容に基づいて最適なレンダリング技術が選択されます。

選択した技術に関わらず、EMF としてエクスポートするにはネイティブインターフェイスを使います。

SVG グラフィックス

Qt モード (GraphicsTechnology=3) では、SVG (Scalable Vector Graphics) が代替のオブジェクトベースの画像フォーマットとして利用できます。
これは、宛先のプログラムがサポートしている場合は、EMF および PDF を置き換えることができます。
Qt モードでは、その他のインポートされたオブジェクトベースの画像 (EMF) は、高解像度のビットマップ画像としてレンダリングされます。

ネイティブグラフィックスモードでは、インポートされた SVG 画像は Qt グラフィックスを使って高解像度のビットマップ画像としてレンダリングされます。

選択した技術に関わらず、SVG 形式でのエクスポートは Qt を使ってレンダリングされます。

高解像度ディスプレイ

高解像度画面グラフィックスは、Qt グラフィックス技術モードでのみサポートされます。

4K、5K、Ultra HD、Quad HD とも呼ばれる高解像度ディスプレイは、より快適な視覚体験をもたらしますが、パフォーマンス上の課題も生じます。
オペレーティングシステム、グラフィック技術、および実際のデータによっては、グラフの更新が数千倍遅くなる場合があります。
オペレーティングシステムが高解像度グラフィックスを最適化するため、この状況は一時的なものと予想されます。
最新情報については、「グラフと高解像度ディスプレイ」のセクションを参照してください。

Igor Pro 7 以前、Windows ではコントロールパネルは座標にピクセルを使って描画されていました。
現在、解像度が 96 DPI を超える場合、それらの座標はポイントとして扱われます。
これにより、高解像度ディスプレイ上でパネルが小さく表示されるのを防ぎます。
詳細はヘルプ Control Panel Resolution on Windows を参照してください。

グラフと高解像度ディスプレイ

高解像度ディスプレイはより快適な視覚体験をもたらす一方で、パフォーマンス上の課題も生じます。
従来、1 ポイントの線の幅は 1 ピクセルでした。
しかし高解像度ディスプレイでは、このような線の幅は 2 ピクセルとなり、オペレーティングシステム、グラフィック技術、実際のデータによっては、描画速度が数千倍遅くなる場合があります。

Igor Pro 8 以降、2 ピクセル以上の幅を持つ線を描画する時には高速なカスタムコードを使っています。
ただし、カスタムコードによる線はシステムコードほど見栄えが良くないため、システムコードの処理が遅い場合、特に非常に長いトレースを使う場合にのみ適用されます。

最速の更新速度が必要な場合、ModifyGraph ライブキーワードに値 2 を指定することで、トレースに高速線描画コードを使わせることができます。
これにより、1 ピクセル幅の線でも新しいコードで描画されます。
速度向上は 2 倍程度となります。

ライブモードのデモ（Live Mode demo）エクスペリメントを使うと、異なる設定がグラフの更新速度にどのように影響するかを確認できます。

Windows 高解像度の推奨事項

高解像度 DPI（高 DPI／ドット毎インチ）ディスプレイは、Windows 標準の 96 DPI よりも大幅に高い解像度を有します。
これらのディスプレイは、Retina、4K、5K、Ultra HD、Quad HD などの名称で呼ばれることがあります。

高解像度モニターのピクセルは通常、標準モニターのピクセルの約半分であるため、高解像度モニターで動作するソフトウェアは調整を行う必要があります。
これらの調整は、オペレーティングシステムとプログラム自体の組み合わせによって行われます。

オペレーティングシステムと Igor が使う Qt フレームワークの両方が複雑さを生み出し、時に完全とは言えない動作を引き起こすことがあります。
このセクションでは、Windows ユーザーがこれらの問題を最小限に抑えるためのガイダンスを提供します。

高解像度ディスプレイを最適に利用するには、Windows 10 以降の使用をお勧めします。
旧バージョンの Windows における高 DPI 機能のテストや明示的なサポートは行っておりません。

ノートパソコンの高解像度ディスプレイのように 1 つのディスプレイのみを使用し、外部ディスプレイが接続されていない場合、正しい動作を実現するために Igor のデフォルト設定を変更する必要はありません。

複数のディスプレイを異なる解像度で使っている場合（例：高解像度のノートパソコン画面と標準解像度の外部モニター）、テキストやメニュー、ウィンドウ、アイコンのサイズが不適切に大きくなったり小さくなったりする問題が発生することがあります。
Windows のディスプレイ設定で高解像度ディスプレイをメインディスプレイに設定すれば、こうした問題の大半または全てが解決されます。

例として、一般的に組み合わせる解像度システム（内蔵高解像度ディスプレイと外部標準解像度モニターを備えたノートパソコン）の設定手順を以下に示します。

1. 外部標準解像度モニターを接続します。

2. ディスプレイ設定ページを開きます。
   これを行うには、スタートメニューを右クリックし、[設定] を選択し、[システム] をクリックし、左側のペインで [ディスプレイ] を選択します。

3. 複数ディスプレイ設定で、パネルの下部付近にある「これらのディスプレイを拡張する」を選択します。
4. 画面上部のディスプレイ図の下にある「識別」リンクをクリックし、各ディスプレイの識別を確認してください。
   本手順では、ディスプレイ#1 が高解像度の内蔵ディスプレイ、ディスプレイ#2 が標準解像度の外部モニターであると仮定します。
5. 内蔵の高解像度ディスプレイを表すボックス（この例ではディスプレイ#1）をクリックします。
6. 解像度を推奨値（例：3840 x 2160）に設定します。
7. スケールとレイアウトを推奨値に設定します。
   推奨値は、お使いのハードウェアに応じて 200%、225%、または 250%となる場合があります。
8. ペインの下部付近にある「これをメインディスプレイにする」チェックボックスを選択してください。
9. 設定コントロールパネルを閉じます。
10. マシンを 2 回再起動します。
    ディスプレイ設定を変更した後は、最良の結果を得るために、マシンを 2 回再起動する必要があります。
11. マシンを 2 回再起動した後、Igor を起動します。
    時折発生する軽微な不具合を除けば、正常に動作するはずです。

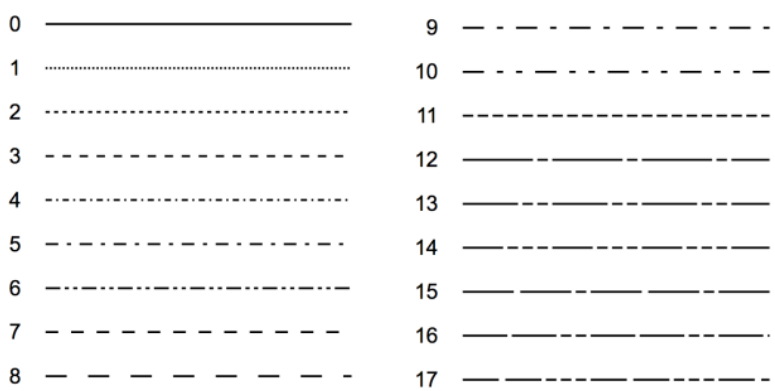
前述の手順で有用な結果が得られない場合は、高解像度ディスプレイを標準解像度に戻す必要があるかもしれません。
その場合は、解像度を標準解像度に設定し、スケーリングとレイアウトを 100% に設定してください。
標準解像度は通常、推奨解像度の半分です。
例えば、3140 x 2160 ではなく 1920 x 1080 です。
この設定を行うことで、どちらのディスプレイもメインディスプレイとして設定できます。

グラフィックス機能

このセクションでは、グラフ、ページレイアウトウィンドウ、および描画レイヤーで使われる一般的なグラフィックス機能について説明します。

破線

様々な線種でトレースをグラフ上に表示できるほか、描画された線、長方形、楕円、ポリゴンを表示できます。
下の表は、ColorsMarkersLinesPatterns.pxp のサンプルエクスペリメントから生成されたもので、Igor のデフォルトの線種を示しています。



通常は必要ありませんが、Misc→Dashed Lines を選択して Dashed Lines ダイアログを使えば、組み込みの破線スタイルを編集できます。

破線 0（実線）は編集できません。
カスタムの破線パターンを作成する必要がある場合は、番号の大きい破線を変更し、番号の小さい破線はデフォルトの状態のままにしておくことをお勧めします。
これにより、番号の小さいパターンはすべてのユーザーで同じになります。

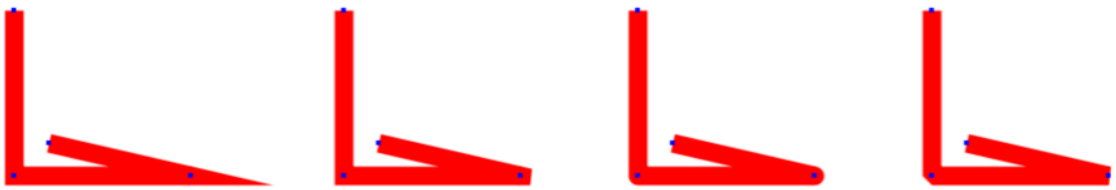
破線のパターンは SetDashPattern コマンドで変更することもできます。

破線はエクスペリメントごとに保存されるため、各エクスペリメントで異なる破線を使用できます。
現在の破線を新規エクスペリメントのプレファレンスとして保存できます。

ライン接合部とエンドキャップのスタイル

グラフトレースや描画された線が非常に太い場合、線分間の接合部や線端の外観をコントロールしたい場合があります。
ModifyGraph の lineJoin キーワードは、点間の線モードにおけるグラフトレースの線の接合をコントロールします。
SetDrawEnv の lineJoin キーワードは、描画された線のコントロールを提供します。

線の接合部には、マイター（Miter）、丸め、またはベベル（面取り）のスタイルがあります。
マイター接合の場合、非常に鋭角な角度でのマイターの長さをコントロールするために、マイター制限を指定できます。
これらの図では、青い点が線の端点と接合点の位置を示しています：



lineJoin={0, 10}
miter, w/ miter limit=10

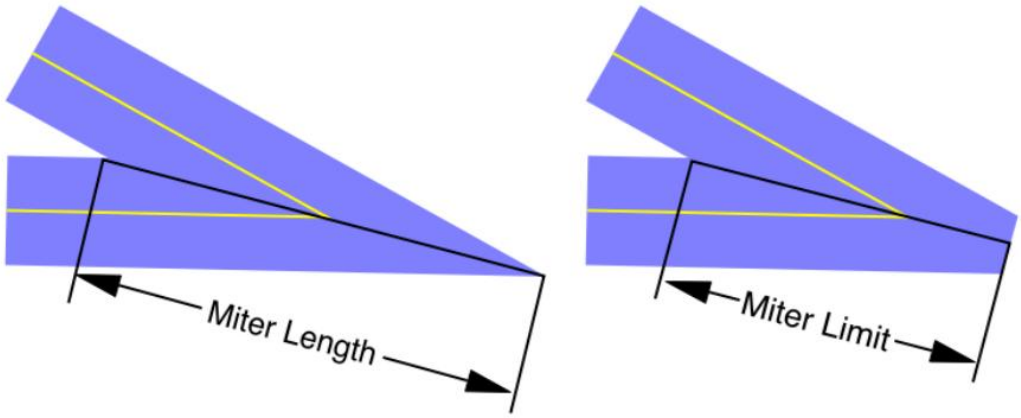
lineJoin={0, sqrt(2)}
miter, limit=sqrt(2)

lineJoin={1, 0}
round

lineJoin={2, 0}
bevel

マイター接合は線分外縁を延長し、交差するまで伸ばします。
鋭角では非常に長いマイターが生じる場合があります。
丸め接合は接合点を中心に円弧を描き、ベベル接合は交点で面取りを施して交差部分を切り詰めます。

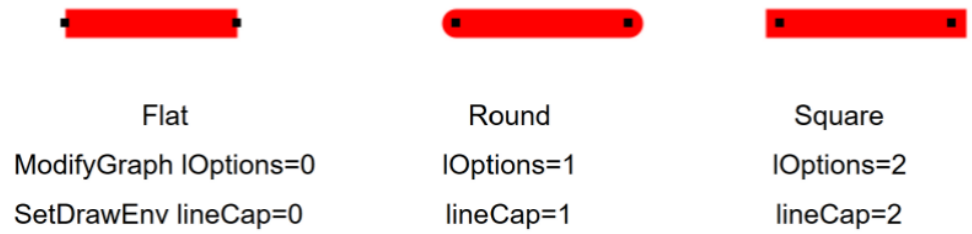
非常に長いマイター延長を避けるために、マイター制限を設定できます：



マイター制限が sqrt(2) の場合、90 度より鋭角な線分接合は切り捨てられます。
残念ながら、切り捨ての性質はグラフィックス技術によって異なります。
GDI、PDF、および Postscript は、ベベル接合に戻すことでマイターを切り捨てます。
Qt グラフィックスおよび GDI+ は、マイター制限でベベル処理することでマイターを切り捨てます。
PNG や JPG などのビットマップ形式でグラフィックをエクスポートする場合、マイター制限は Miscellaneous Settings ダイアログで選択したグラフィック技術（通常は Qt グラフィックス）によってコントロールされます。
上記の 2 枚目の図は Qt グラフィックスで PNG ビットマップとして描画されたものです。
非常に鋭角な交点がマイター制限で切り詰められています。

線の端部の描画方法をコントロールするには、ModifyGraph lOptions キーワードを使うか、描画済み線に対しては SetDrawEnv lineCap を使います。

線の端点は平ら、丸、または四角にできます。
次の図は各オプションの外観を示しています。
ドットは各線の幾何学的端点が位置する箇所を示しています：



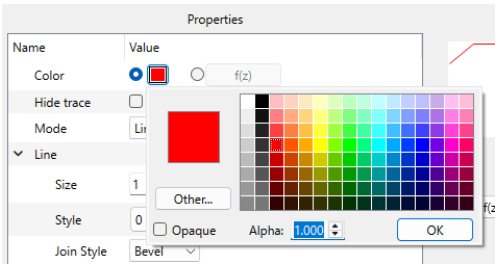
破線の各区間の末端には、エンドキャップスタイルが適用されます：

スクエアエンドキャップは、ラインのスクエアエンドを幾何学的エンドを超えて延長するものであり、Igor ではおそらく有用ではありません。

色環境

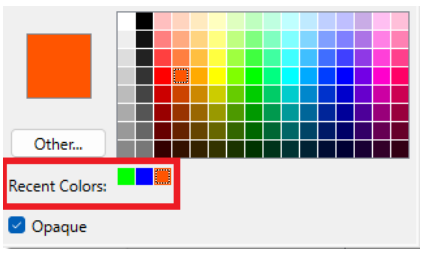
Igor には、グラフのトレース、テキスト、軸、背景などに使用できる色を含むメインカラーパレットがあります。
メインカラーパレットは、Modify Trace Appearance ダイアログなど、いくつかのダイアログでポップアップメニューとして表示されます。
このセクションでは、このパレットについて説明します。

Igor には、コンタープロットや画像プロットを表示する時に選択可能なカラーテーブルとカラーインデックスウェーブも備わっています。
これらはヘルプ Contour Plots と Image Plots で説明しています。



カラーパレットに表示されている色から選択できます。

Other ボタンを使うと、パレットにない色を選択できます。
Igor を使うにつれて、Recent Colores エリアに色が追加されます。



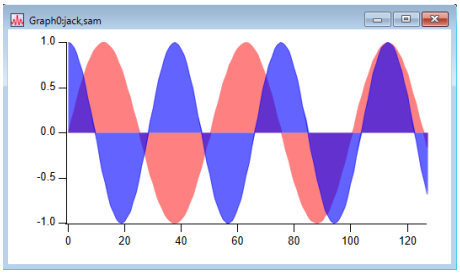
Miscellaneous Settings ダイアログの Color Settings カテゴリで Save and restore color palette's recent colors チェックボックスを選択している場合、終了時に最近のカラーが記憶され、再起動時に復元されます。

色のブレンド

RGB 形式で色を指定できる場所では、オプションで「アルファ」と呼ばれる 4 番目のパラメーターを指定できます。
アルファ値は、R、G、B 値と同様に、65535（100% 不透明）から 0（100% 透明）までの範囲を取ります。
中間値は半透明効果を提供します。

例えば：

```
Make jack=sin(x/8),sam=cos(x/6); Display jack, sam
ModifyGraph mode=7,hbFill=2
ModifyGraph rgb(jack)=(65535,32768,32768)
ModifyGraph rgb(sam)=(0,0,65535,40000)  // Translucent
```



カラーブレンド機能は Igor Pro 7 で追追加されました。

「Windows のグラフィックス技術」のセクションで説明している古い GDI グラフィックスモードでは、カラーブレンドは機能しません。

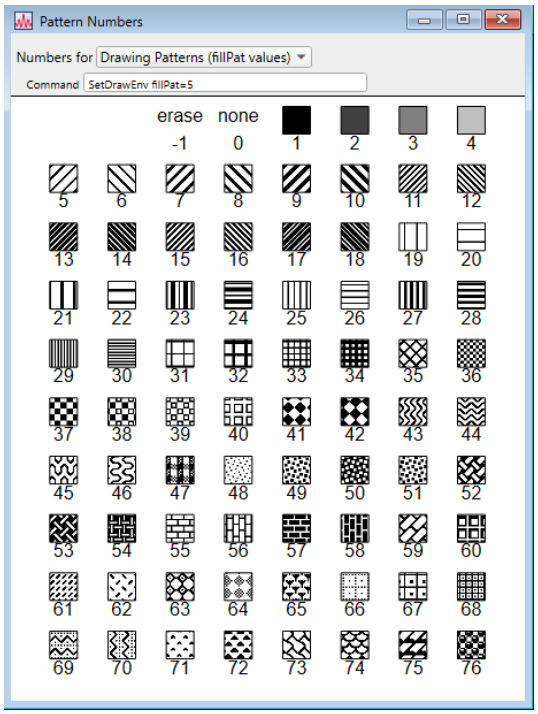
EPS 形式でエクスポートされたグラフィックスではカラーブレンドが機能しません。
代わりに PDF を使ってください。

塗りつぶしパターン

塗りつぶしパターンはグラフや描画レイヤーで使用できます。
右のテーブルは、ColorsMarkersLinesPatterns.pxp のサンプルエクスペリメントから生成されたもので、利用可能な塗りつぶしパターンを示しています。

塗りつぶしパターンコードは描画コマンドに適しています。
ModifyGraph hbFill などのグラフ関連コマンドについては、以下の調整を行ってください。

- Erase は -1 ではなく 1
- 0 より大きい全てのパターンコードに 1 を加える



グラデーション塗りつぶし

描画要素、グラフトレースの塗りつぶし、グラフウィンドウおよびプロット領域の背景塗りつぶしにグラデーション塗りつぶしを指定できます。
グラデーション塗りつぶしとは、色が徐々に変化する塗りつぶしです。

プログラム上では、ModifyGraph、ModifyLayout、または SetDrawEnv コマンドを呼び出す時に、gradient および gradientExtra という 2 つのキーワードを使ってグラデーションを指定します。

gradient キーワードの構文には 2 つの形式があります。
最初の形式はグラデーション全体のコントロールを提供し、 2 番目の形式は色の詳細をコントロールします。

gradient キーワードの最初の形式は次のとおりです：

```
gradient = 0, 1, 2, or 3
```

0：グラデーションを完全に削除します。
1：既存のグラデーションを有効にするか、デフォルト値で新しいグラデーションを作成します。
2：グラデーションをオフにしますが、設定は保持します。
3：ModifyLayout コマンドと組み合わせて使われ、レイアウト全体のグラデーションが設定されている場合でも、特定のページにグラデーションを表示しないことを示します。

gradient キーワードの第二の形式は次のとおりです：

```
gradient = {type , x0 , y0 , x1 , y1  [,color0  [, color1 ] ]}
```

type はビットフィールドです。

ビット 0 から 3 はグラデーションモードを指定します。
0 は線形、1 は放射状、その他の値は予約済みです。

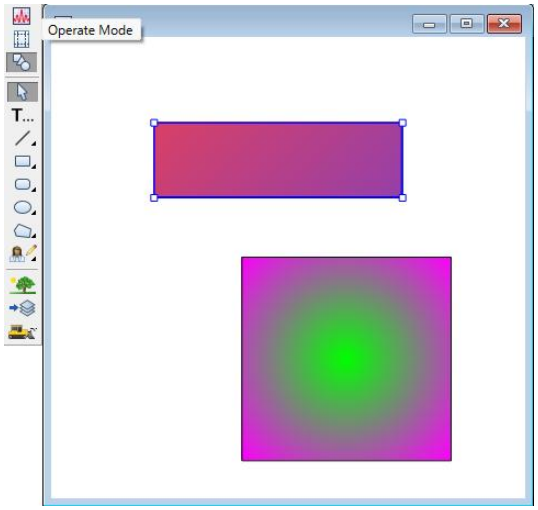
ビット 4 から 7 は座標モードを指定します。
0 はオブジェクト矩形、1 はウィンドウ矩形です。

ウィンドウ矩形モードにおける放射状グラデーションを表す type パラメーターを構築するには、次のように記述できます：

```
Variable gradientType = 1 | (1*16)
```

以下のコマンドは、オブジェクト矩形モードとウィンドウ矩形モードの違いを示します。
これらのコマンドを実行した後、矩形をドラッグして移動して確認してください：

```
Display /W=(150,50,474,365)
SetDrawLayer UserFront
SetDrawEnv xcoord=abs,ycoord=abs
SetDrawEnv save
SetDrawEnv
fillfgc=(16385,16388,65535),fillbgc=(65535,16385,16385)
// Window rect mode
SetDrawEnv gradient={16, 0, 0, 1, 1}
DrawRect 25,38,95,138
// Object rect mode
SetDrawEnv gradient={1, 0.5, 0.5, 0, 1, (0,65535,0),
    (65535,0,65535)}
DrawRect 130,150,273,289
ShowTools/A
```



x0、y0、x1、y1 パラメーターは、グラデーションを定義する正規化された位置を指定します。
(0,0) は境界矩形の左上隅、(1,1) は右下隅です。
線形グラデーションの場合、(x0, y0) は開始点を、(x1, y1) は終了点を定義します。
実際の範囲ではなく、直線の傾きのみが使われます。
放射状グラデーションの場合、(x0, y0) は境界円の中心を定義し、(x1, y1) は現時点では使われません。

オプションの color0 および color1 キーワードは (r,g,b) または (r,g,b,a) として指定されますが、アルファの使用はすべてのプラットフォームで完全にはサポートされていません。
省略された場合、または (0,0,0) の値が指定された場合、実際に使われる色は、その状況におけるデフォルト値となります。
デフォルト値は、個々のコマンドのドキュメントで指定されています。

gradientExtra キーワードは追加の色変化点を追加します。
必要な数の変化点を追加できます。
gradientExtra キーワードの構文は次のとおりです：

```
gradientExtra = {loc , color }
```

loc は 0.0 から 1.0 の範囲です。
正確に 0 または 1 の値は、gradient キーワードで指定された元の color0 または color1 の値を置き換えます。

color は (r,g,b) または (r,g,b,a) で指定されますが、アルファチャンネルの使用はすべてのプラットフォームで完全にはサポートされていません。

以下のコマンドは、gradient および gradientExtra キーワードをサポートしています。

SetDrawEnv
グラデーションを指定すると、描画要素の塗りつぶしが上書きされます。
color0 パラメーターを省略するか、(0,0,0,0) を指定した場合、現在のパターン背景色が使われます。
color1 パラメーターを省略するか、(0,0,0,0) を指定した場合、現在の前景色が使われます。
すべての gradientExtra キーワードは、gradient キーワード自体と同じ行に記載する必要があります。

ModifyLayout
個々のページに対してグラデーションを指定するには、/PAGE=pageNum フラグと gradient および gradientExtra キーワードを組み合わせて使います。
ページ番号は 1 から始まります。
現在アクティブなページを使うには /PAGE=0 を使ってください。

/PAGE=-1 は、明示的なグラデーションが設定されていない対象ページレイアウト内のすべてのページに適用されるレイアウトのグローバルなグラデーションを変更する操作を引き起こします。

トレースに対する **ModifyGraph**
gradient を指定すると、トレースの塗りつぶしが上書きされます。
gradient および gradientExtra キーワードは、次のようにトレース名を指定しない限り、すべてのトレースに適用されます：

```
gradient(<trace name>) = {...}
gradientExtra(<trace name>) = {...}
```

color0 パラメーターを省略するか (0,0,0,0) を指定した場合、現在のパターン背景色が使われます。
color1 パラメーターを省略するか (0,0,0,0) を指定した場合、plusRGB カラーが使われます。
type パラメーターがウィンドウ矩形を指定する場合、プラス領域とマイナス領域が自動的に使われ、color1 はゼロレベルから離れます。

カラーに対する **ModifyGraph**
wbGradient および wbGradientExtra キーワードは、ウィンドウの背景グラデーション（存在する場合）をコントロールします。
gbGradient および gbGradientExtra キーワードは、グラフプロット領域の背景グラデーション（存在する場合）をコントロールします。
メニュー File→Example Experiments→Sample Graphs の「Demo Experiment #1」と「Demo Experiment #2」のサンプルエクスペリメントでは、これらのグラデーションが示されています。
Macros メニューを使って、これらの機能をオン/オフできます。

グラフィックスのエクスポート（Windows）

このセクションでは、Windows 上で Igor のグラフ、ページレイアウト、テーブル、および Gizmo プロットから別のプログラムへグラフィックスをエクスポートする方法について説明します。
グラフィックスは、Edit→Export Graphics を選択してクリップボード経由で、または File→Save Graphics を選択してファイル経由でエクスポートできます。

Igor Pro は様々なグラフィックス出力形式をサポートしています。
グラフィックスの性質、プリンター、および出力先のプログラムの特性に応じて適切な形式を選択することで、通常は非常に良好な結果を得られます。

残念ながら、特定の状況に最適なエクスポート形式を見つけるには、試行錯誤が必要な場合があります。
このセクションでは、適切な選択を行うために必要な情報を提供します。

この表は、Windows で利用可能なグラフィックス出力形式を示しています：

エクスポートフォーマット	**エクスポート方法**	**メモ**
EMF (Enhanced Metafile)	クリップボード、ファイル	Windows 固有のベクトルフォーマット。
BMP (Bitmap)	クリップボード、ファイル	Windows 固有のビットマップフォーマット。 圧縮を使用しません。
PDF	クリップボード、ファイル	プラットフォーム非依存で高品質。 CMYK カラーを使った Igor PDF は透明をサポートしていません。
EPS (Encapsulated Postscript)	ファイルのみ	画面プレビューを除き、プラットフォーム非依存。 高解像度をサポートします。 EPS は透明をサポートしていません。 PostScript プリンターでの印刷、PDF ファイルの作成、または PostScript 対応プログラム（例：Adobe Illustrator、Tex）へのエクスポート時にのみ有効です。
PNG (Portable Network Graphics)	クリップボード、ファイル	プラットフォーム非依存のビットマップフォーマット。 非可逆圧縮を使用します。 高解像度をサポートします。 PNG は科学的なグラフィックに適した選択肢です。
TIIF	クリップボード、ファイル	プラットフォーム非依存のビットマップフォーマット。 Igor の実装は現在、圧縮を使っていません。 高解像度をサポートしています。
SVG	クリップボード、ファイル	プラットフォームに依存しない 2D ベクトルおよびラスターグラフィックス形式。 出力先プログラムが SVG をサポートしている場合に適した選択肢です。

執筆時点では、Windows プログラムで SVG をサポートしているものはあまりありません。

メタファイルフォーマット

メタファイルフォーマットは、画像を構成する線、矩形、テキストなどの個々のオブジェクトに対する描画コマンドをサポートする Windows ベクトルグラフィックス形式です。
描画プログラムは、メタファイルを構成要素に分解して、個々のオブジェクトを編集できるようにします。
ほとんどのワードプロセッサは、メタファイルをブラックボックスとして扱い、それを表示または印刷するためにオペレーティングシステムを呼び出します。

拡張メタファイル（EMF）は、Windows ネイティブの主要なグラフィック形式です。
主に 2 種類存在します：従来のプレーン EMF と、新しい EMF+ です。
Igor はデフォルトで「デュアル EMF」を出力します。
デュアル EMF にはプレーン EMF と EMF+ の両方が含まれます。
EMF+ をサポートしないアプリケーションはプレーン EMF コンポーネントを使います。
透明（アルファチャンネル付きカラー）を使う場合は EMF+ が必要です。
出力先のプログラムが EMF+ に対応していない場合、従来の EMF 形式でエクスポートできます（詳細は「グラフィックス技術」のセクションを参照）。

EMF は、ほぼすべての Windows プログラムがインポートでき、クリップボードへのコピーやファイルへの書き込みが可能なため、使いやすいです。
一部のプログラム（特に Microsoft Office の古いバージョンなど）では、クリップボードから EMF を貼り付ける時に「貼り付け」ではなく「形式を選択して貼り付け」を選択する必要があります。

描画プログラムは、EMF を構成要素に分解して個々のオブジェクトを編集できるようにしますが、メタファイル形式の複雑さのために、しばしば誤った分解を行います。
EMF を正しく分解するアプリケーションでも、デュアル EMF には混乱します。
そのようなプログラムでは、グラフィック技術を GDI に設定して、プレーンな EMF をエクスポートする必要があります。

BMP フォーマット

BMP は Windows ビットマップ形式です。
多くのプログラムで受け入れられますが、圧縮されていないため、大量のメモリとディスク容量を必要とします。
BMP は DIB（デバイス独立ビットマップ）としても知られています。

エクスポート先のプログラムが PNG をサポートしている場合、PNG の方が適しています。

PDF フォーマット（Windows）

PDF（Portable Document Format）は、Adobe のプラットフォーム非依存のベクトルグラフィックス形式です。
これは、目的のプログラムがこれをサポートしている限り、最適な形式です。

Igor の PDF 形式は OS ではなく Igor 独自のコードによって生成されます。
Igor Pro 9 以降、Igor PDF は透明度をサポートし、フォント埋め込みの精度が向上しています。
ただし、CMYK カラーを指定した場合、旧式のコードが使われます。

Windows 上で Igor がグラフィックスを PDF としてエクスポートする場合、Igor にインポートされた画像（PDF 内の画像を含む）はすべてビットマップ画像としてエクスポートされます。

PDF 内のぼやけた画像（Windows）

Igor が画像プロットをエクスポートする場合、可能な限り 1 つの画像オブジェクトとしてエクスポートします。
しかし、一部の PDF ビューア（特に Apple のもの）は、独自の判断でピクセルをぼかしてしまいます。
これを回避するには、ModifyImage の補間キーワードに値 -1 を指定して、画像ピクセルを個別の矩形として描画するよう Igor に指示します。
結果として生成される PDF のサイズが大幅に大きくなるため、必要な場合にのみこの操作を行ってください。

EPS（Encapsulated PostScript）フォーマット（Windows）

EPS は、プレーンテキスト形式の PostScript コマンドで構成される、プラットフォームに依存しない広く使われたベクトルグラフィックス形式でした。
EPS はほとんど使われなくなりましたが、まだ使われています。
通常は高品質ですが、PostScript プリンターで印刷する場合、または Adobe Illustrator などの PostScript 対応プログラムにエクスポートする場合にのみ機能します。
Helvetica などの PostScript フォントのみを使ってください。
EPS は透明をサポートしていません。

Igor Pro 7 以前のバージョンでは、Igor は EPS ファイルに画面プレビューを埋め込んでいました。
しかし、このプレビューはクロスプラットフォームに対応しておらず、多くのプログラムで問題を引き起こしたため、現在は行われていません。

EPS ファイルは通常、色を表現するために RGB エンコーディングを使いますが、CMYK も使用できます。
詳細は「色のエクスポート（Windows）」のセクションを参照してください。

Igor Pro は PostScript Level 2 を使って EPS ファイルをエクスポートします。
これにより、印刷時の塗りつぶしパターンが大幅に改善され、Adobe Illustrator が Igor の塗りつぶしパターンを正しくインポートできるようになります。
古いプリンターとの下位互換性を確保するため、SavePICT コマンド時に /PLL=1 を指定することで、Igor に Level 1 の使用を強制できます。

EPS としてエクスポートするグラフやページレイアウトに、他のプログラムからインポートされた非 EPS 画像が含まれている場合、Igor はその画像を EPS 出力ファイルに組み込まれた画像としてエクスポートします。

Igor は TrueType フォントをアウトラインとして埋め込むことができます。
詳細は「フォントの埋め込み（Windows）」のセクションおよびヘルプ Symbols with EPS and Igor PDF を参照してください。

SVG フォーマット

SVG（Scalable Vector Graphics）は、World Wide Web Consortium によって開発された XML ベースのプラットフォーム非依存の 2D ベクトルおよびラスターグラフィックス形式です。
ウェブページでのグラフィックス表示によく使われ、目的のプログラムがサポートしている場合、他の用途にも適しています。
執筆時点では、Microsoft Office は SVG をサポートしていますが、その他の Windows プログラムでサポートしているものはあまりありません。

プラットフォーム非依存ビットマップ形式（Windows）

PNG (Portable Network Graphics) は、プラットフォームに依存しないビットマップ形式です。
ロスレス圧縮を使い、高解像度をサポートしています。
JPEG や GIF に代わる優れた選択肢です。
Igor はファイルやクリップボードを介して PNG 画像をエクスポートおよびインポートできますが、PNG ファイルの挿入が可能なプログラムの中には、クリップボードから PNG 画像を貼り付けることができないものもあります。

JPEG は非可逆圧縮形式であり、その主な利点はすべての Web ブラウザーで受け入れられることです。
しかし、現代の Web ブラウザーはすべて PNG をサポートしているため、科学的なグラフィックに JPEG を使う理由はほとんどありません。
Igor はクリップボード経由で JPEG のエクスポートとインポートが可能ですが、ほとんどの Windows プログラムは JPEG を貼り付けることができません。
ただし、Microsoft Office は可能です。

TIFF は、デジタル写真によく使われる Adobe のフォーマットです。
Igor の TIFF エクスポート実装は圧縮を使いません。
TIFF ファイルは通常、色を指定するために RGB 方式を使用しますが、CMYK も使用できます。
詳細は「色のエクスポート（Windows）」のセクションを参照してください。
PNG をサポートしないプログラムにエクスポートする場合を除き、PNG ではなく TIFF を使う特別な理由はありません。
Igor はファイル経由およびクリップボード経由で TIFF のエクスポートとインポートが可能であり、ほとんどのグラフィックスプログラムは TIFF をインポートできます。

グラフィックスフォーマットの選択（Windows）

グラフィックスの種類、出力先プログラム、プリンタの機能、オペレーティングシステムの動作、ユーザーの優先順位など多岐にわたるため、エクスポート形式の選択について決定的な指針を示すことはできません。
ただし、ほとんどの状況で有効なアプローチを以下に示します。

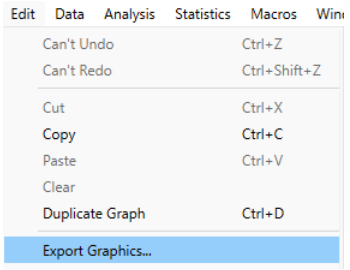
PNG は、ビットマップである画像プロットや Gizmo プロットをエクスポートする時の推奨形式です。

目的のプログラムが PDF または SVG を受け入れる場合、高品質なベクターグラフィックスとプラットフォーム非依存性のため、おそらくこれらが最適な選択肢となるでしょう。

EPS も非常に高品質な形式であり、出力先プログラムが対応していれば良好に動作しますが、透明をサポートしていません。

PDF、SVG、EPS が適切でない場合、ベクトルグラフィックスの次善の選択肢は高解像度のビットマップとなります。
PNG 形式が推奨されるのは、プラットフォームに依存せず、ロスレス圧縮をサポートしているためです。

クリップボード経由でのグラフィックスのエクスポート（Windows）



アクティブなウィンドウからグラフィックスをクリップボード経由でエクスポートするには、Edit→Export Graphics を選択します。
これにより、Export Graphics ダイアログが表示されます。

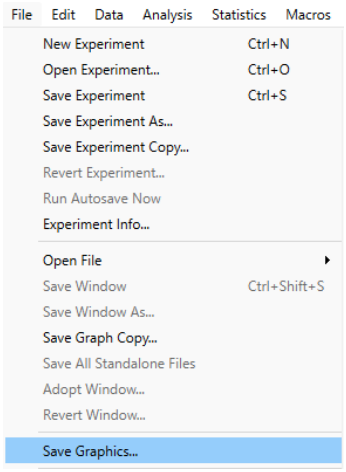
OK ボタンをクリックすると、Igor はアクティブウィンドウのグラフィックスをクリップボードにコピーします。
その後、別のプログラムに切り替えて貼り付けを行うことができます。

グラフ、ページレイアウト、または Gizmo プロットがアクティブで操作モードの場合、Edit→Copy を選択すると、Export Graphics ダイアログで最後に使われた形式がクリップボードにコピーされます。
テーブルの場合、Edit→Copy を選択すると、選択された数値がクリップボードにコピーされ、グラフィックスはコピーされません。

ページレイアウトでオブジェクトが選択されている場合、またはマーキーがアクティブな場合、Edit→Copy を選択すると、Igor オブジェクトが Igor 内部で使われる形式と拡張メタファイル形式でコピーされ、Export Graphics ダイアログの形式は使われません。

Igor は複数の異なる形式でエクスポートできますが、すべてのプログラムがクリップボード上のそれらを認識できるわけではありません。
ファイル経由でエクスポートする必要がある場合があります。

ファイル経由でのグラフィックスのエクスポート（Windows）



アクティブなウィンドウからグラフィックスをファイル経由でエクスポートするには、File→Save Graphics を選択します。
これにより Save Graphics ダイアログが表示されます。

ダイアログの Format 領域のコントロールは、各エクスポート形式に適したオプションを反映するように変更されます。

Do It ボタンをクリックすると、Igor がグラフィックスファイルを作成します。
その後、別のプログラムに切り替えてそのファイルをインポートできます。

Path ポップアップメニューから _Use Dialog_ を選択すると、Igor は Save File ダイアログを表示し、保存ファイルの場所と名前を指定できます。

レイアウトの一部をエクスポート（Windows）

ページレイアウトの一部をエクスポートするには、マーキーツールで該当部分を選択し、Edit→Export Graphics または File→Save Graphics を選択します。

マーキーを使用せず、Crop to Page Contents チェックボックスがオンの場合、Igor は使用中のレイアウト領域と小さな余白をエクスポートします。
オフの場合、Igor はページ全体をエクスポートします。

カラーのエクスポート（Windows）

EPS および TIFF グラフィックス形式は、通常、RGB スキームを使って色を指定します。
一部の出版物では、RGB ではなく CMYK の使用が要求されますが、出版社が出力デバイスの実際の特性を使って RGB から CMYK への変換を行う場合に最良の結果が得られます。
CMYK を要求する出版物については、SavePICT /C=2 フラグを使用できます。

フォントの埋め込み（Windows）

このセクションの内容はほぼ使われていません。
CMYK カラー用に生成された EPS ファイルおよび PDF ファイルにのみ適用されます。
それでも、この情報は特殊な状況では必要となります。

TrueType フォントは EPS ファイルおよび PDF ファイルに埋め込むことができます。
これにより、対応する PostScript フォントがインストールされていないシステムでも EPS や PDF ファイルを印刷できます。
また、フォントの埋め込みが必要な出版物にも役立ちます。

フォントの埋め込みは常に有効ですが、標準フォントを埋め込まないことを勧めます。
ほとんどの用途では、非標準フォントのみを埋め込むことが最良の選択です。

Igor は TrueType フォントを、そのアウトラインから生成された合成 PostScript Type 3 フォントとして埋め込みます。
フォントには実際に使われた文字のみが含まれます。

システム上のすべてのフォントおよびフォントスタイルを埋め込むことができるわけではありません。
一部のフォントは埋め込みを許可していない場合や、TrueType ではない場合、エラーが発生する場合があります。
EPS ファイルを出版社に送る前に、必ずローカルプリンタでテストするか、Adobe Illustrator にインポートしてテストしてください。
また、「TrueType Outlines.pxp」のサンプルエクスペリメントを使って、埋め込み用のフォントを検証することもできます（Example Experiments→Feature Demos 2→TrueType Outlines）。

EPS の場合、Igor はフォント置換を実行した後、「PostScript フォント名（Windows）」のセクションで定義されたテーブルにフォント名を照会することで、非標準フォントかどうかを判定します。
さらに、非プレーンフォントスタイル名がプレーンフォント名と同じ場合、そのフォントは埋め込まれます。
これは、イタリック体バージョンが存在しない標準 PostScript フォント（Symbol など）の場合、イタリック体では埋め込みが行われるが、プレーン体では行われないことを意味します。

PDF の場合、非標準フォントとは、PDF 仕様によって、あらゆる PDF リーダーに組み込まれることが保証されている基本フォント以外のフォントを指します。
これらのフォントは、Helvetica および Times のプレーン、ボールド、イタリック、ボールドイタリックのスタイル、ならびに Symbol および Zapf Dingbats のプレーンバージョンです。
埋め込みが行われない場合、またはフォントを埋め込めない場合、上記以外のフォントは Helvetica としてレンダリングされ、意図した結果が得られません。

PostScript フォント名（Windows）

PostScript を生成する時、Igor は適切な PostScript フォント名を生成する必要があります。
これは Windows 環境下では問題を引き起こすことがあります。
Igor はまた、非 PostScript フォントを PostScript フォントに置換できる必要があります。
以下は標準規格に変換されるフォント名のリストです：

TrueType フォント名	PostScript フォント名
Arial	Helvetica
Arial Narrow	Helvetica-Narrow
Book Antiqua	Palatino
Bookman Old Style	Bookman
Century Gothic	AvantGarde
Century Schoolbook	NewCenturySchlbk
Courier New	Courier
Monotype Corsiva	ZapfChancery
Monotype Sorts	ZapfDingbats
Symbol	Symbol
Times New Roman	Times

画像

Igor は、他のプログラムから画像をインポートし、グラフ、ページレイアウト、ノートブックに表示できます。
また、グラフ、ページレイアウト、テーブルから画像をエクスポートし、他のプログラムで使うこともできます。
エクスポートについては「グラフィックスのエクスポート（Windows）」で説明しています。
このセクションでは、Igor への画像のインポート方法、画像で実行可能な操作、および Igor による画像の保存方法について説明します。

画像をグラフィックスではなくデータとしてインポートする方法については、ヘルプ Loading Image Files を参照してください。

画像のインポート

画像をインポートするには３つの方法があります。

1. クリップボードからグラフ、レイアウト、またはノートブックに貼り付けます。
2. Pictures ダイアログ（Misc メニュー）を使って、ファイルまたはクリップボードから画像をインポートします。
3. LoadPICT コマンドを使って、ファイルまたはクリップボードから画像をインポートします。

これらの方法（ノートブックへの貼り付けを除く）はそれぞれ、１つ以上のグラフやレイアウトで使用できる名前付きグローバル画像オブジェクトを作成します。
ノートブックへの貼り付けは、そのノートブックにローカルな画像を作成します。

この表は、Igor が画像をインポートできるグラフィック形式の種類を示しています：

フォーマット	メモ
PDF (Portable Document Format)	Igor Pro 9.0 以降でサポート。 「PDF 画像のインポート」のセクションを参照。

EMF (Enhanced Metafile)	Windows ネイティブグラフィックスでのみサポート。 さまざまな種類の EMF 画像に関する情報は、「Windows 上のグラフィックス技術」のセクションを参照。
BMP (Windows bitmap)	BMP は DIB（デバイス独立ビットマップ）とも呼ばれる。
PNG (Portable Network Graphics)	ロスなしのクロスプラットフォームビットマップ形式。
JPEG	ロスありのクロスプラットフォームビットマップ形式。
TIFF (Tagged Image File Format)	ロスなしのクロスプラットフォームビットマップ形式。
EPS (Encapsulated PostScript)	高解像度ベクトル形式。 PostScript プリンターが必要。 Qt グラフィックスモードでは、画面上にプレビューが表示。
SVG (Scalable Vector Graphics)	クロスプラットフォーム対応のベクトルおよびビットマップ形式。 常に Qt グラフィックスを使って描画される。 他のグラフィックス技術モードでは、高解像度のビットマップに描画された後、画面に表示、エクスポート、または印刷される。

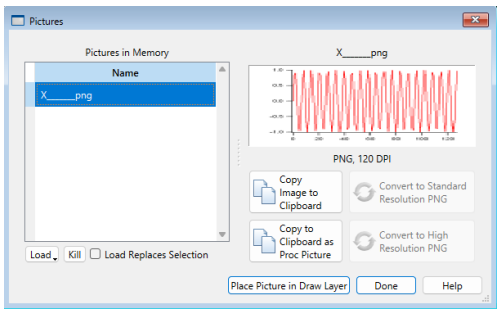
現在のプラットフォームでサポートされていないフォーマットは、灰色のボックスとして描画されます。

PDF 画像のインポート

PDF（Portable Document Format）は、Adobe のプラットフォームに依存しないベクターグラフィックス形式です。
ただし、すべてのプログラムが PDF をインポートできるわけではありません。

Igor Pro 9 以降では、PDF グラフィックスのインポートをサポートしています。
PDF 画像はビットマップを使って描画されます。

画像ギャラリー



上記のいずれかの方法で名前付き画像を作成すると、Igor はそれを現在のエクスペリメントの画像ギャラリーに保存します。
エクスペリメントを保存すると、画像ギャラリーはエクスペリメントファイルに保存されます。
Pictures ダイアログ（Misc メニュー）を使ってコレクションを確認できます。

Igor は画像に名前を付け、Igor プロシージャから参照できるようにします。
例えば、レイアウトに画像を貼り付けると、Igor は PICT_0 という形式の名前を割り当て、画像ギャラリーに保存します。
その後レイアウトを閉じ、Igor に再作成マクロの作成を指示すると、マクロはその名前で画像を参照します。

名前付き画像の名前変更は、Misc メニューの Pictures ダイアログ、Misc メニューの Rename Objects ダイアログ、Data メニューの Rename ダイアログ、または RenamePICT コマンドで実行できます。
名前付き画像の削除は、Pictures ダイアログまたは KillPICTs コマンドで実行できます。

ノートブックの画像

フォーマットされたノートブックに画像を貼り付けると、そのノートブック内でのみ利用可能なノートブック画像が作成されます。
これらは、ワードプロセッサ文書内の画像とまったく同じように機能します。
コピーや貼り付けが可能です。
これらの画像は、Pictures ダイアログボックスや Rename Objects ダイアログボックスには表示されません。