CONTENTS

ビジュアルヘルプ – 3D グラフィックス（3）

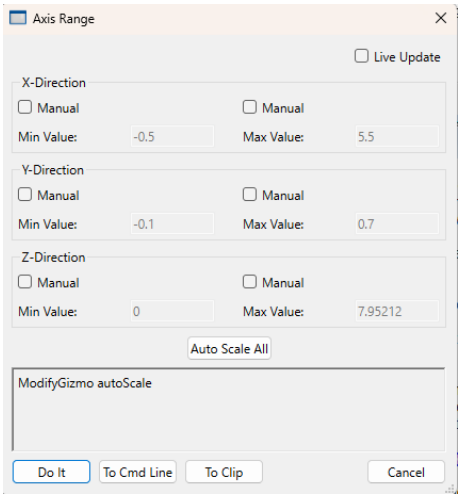
Gizmo データオブジェクト

データオブジェクト（「ウェーブベースのオブジェクト」とも呼ばれます）は、Igor ウェーブのデータを表す表示オブジェクトです。
これには、3D 散布図、パスプロット、サーフェスプロット、リボンプロット、等値面プロット、ボクセルグラムプロット、3D 棒グラフが含まれます。

Gizmo オブジェクトのスケーリング

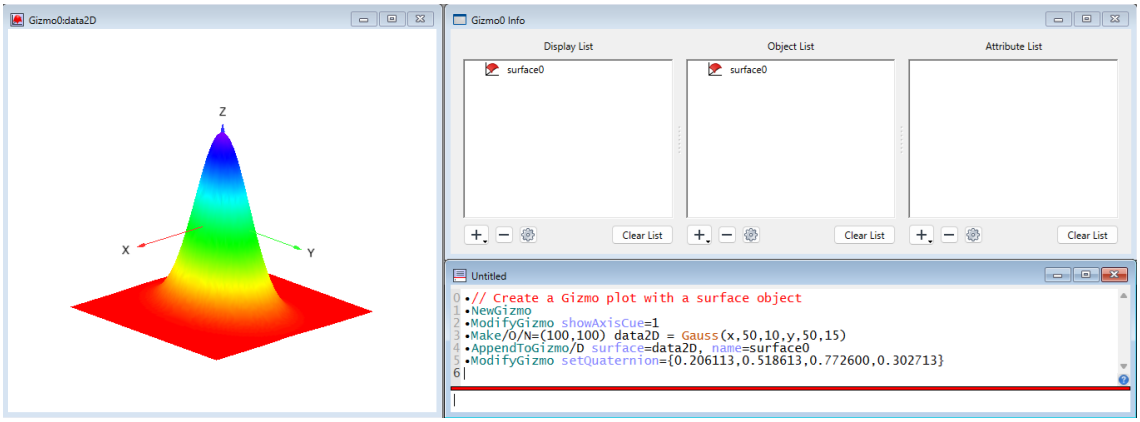
データオブジェクトは、X、Y、Z データ軸のセットに対して表示されます。
これらのデータ軸は、プロットに軸オブジェクトを追加するかどうかにかかわらず存在します。
データ軸は ±1 の表示ボリュームを埋めます。

デフォルトでは、データ軸はプロット内の全データオブジェクトのデータ範囲に合わせて自動スケーリングされます。
Gizmo→Axis Range を使って、データ軸の範囲を変更できます。



次のコマンドはこれらの点を示しています。
新しいエクスペリメントで実行してみてください。

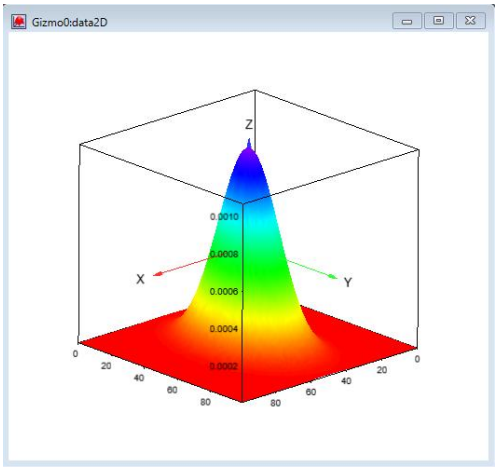
```
// サーフェスオブジェクトで Gizmo プロットを作成
NewGizmo
ModifyGizmo showAxisCue=1
Make/O/N=(100,100) data2D = Gauss(x,50,10,y,50,15)
AppendToGizmo/D surface=data2D, name=surface0
ModifyGizmo setQuaternion={0.206113,0.518613,0.772600,0.302713}
```



ここで、Gizmo→Axis Range を選択すると、現在非表示になっているすべてのデータ軸が自動スケーリングモード（Manual チェックボックスはオフ）であり、X 軸と Y 軸の範囲が 0 から 99 であることがわかります。
これらの値は、data2D データのデフォルトの X 値と Y 値が 0 から 99 の範囲であることに由来します（デフォルトのウェーブスケーリング）。

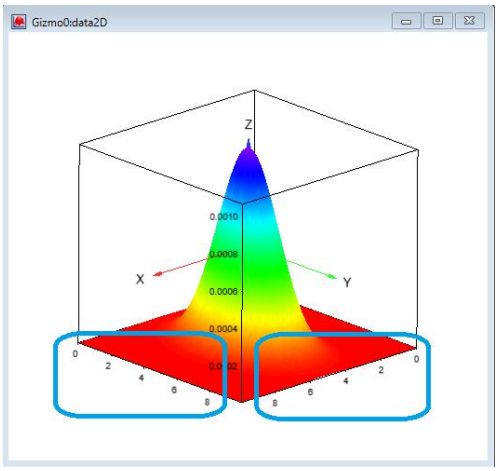
Z 軸の範囲は data2D 内の Z 値の範囲に基づいています。
（Axis Range ダイアログを開いた場合は、Cancel をクリックして閉じます。）

これらのデータ軸を視覚化するために、目盛線と目盛ラベル付きのボックス軸オブジェクトを追加します。




```
// 目盛線と目盛ラベル付きのボックス軸を追加
AppendToGizmo/D Axes=BoxAxes, name=axes0
ModifyGizmo ModifyObject=axes0, objectType=Axes,
        property={4,ticks,3}
ModifyGizmo ModifyObject=axes0, objectType=Axes,
        property={8,ticks,3}
ModifyGizmo ModifyObject=axes0, objectType=Axes,
        property={9,ticks,3}
```

データ内の値を変更した場合、データ軸が自動スケーリングモードであるため、変更後もデータは軸（常に表示領域を埋める）を埋めたままになります。

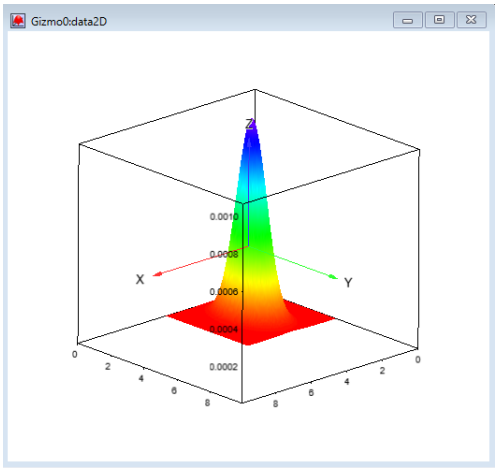



```
// データの X と Y の範囲を変更
SetScale x, 0, 10, "", data2D; SetScale y, 0, 10, "",
        data2D
```

次に、X 軸と Y 軸のデータ軸を手動スケーリングモードに設定しますが、その範囲は変更しません。

```
// X 軸と Y 軸を手動スケーリングモードに設定
ModifyGizmo setOuterBox={0,9.9,0,9.9,1.5286241e-11,
        0.001061033}
ModifyGizmo scalingOption=48
```

データの範囲を変更すると、データ軸は変更されず、データが軸を埋め尽くさなくなります。

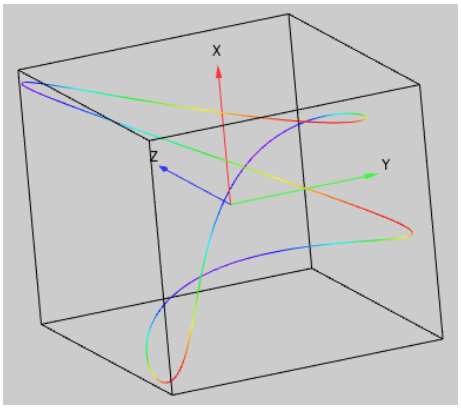



```
// データの X と Y の範囲を変更
SetScale x, 0, 5, "", data2D; SetScale y, 0, 5, "", data2D
```

要約すると、X、Y、Z データ軸は、表示されているかどうかに関わらず常に存在します。
それらは常に ±1 の表示ボリュームを満たします。
データオブジェクトは、デフォルトで自動スケーリングされるデータ軸に対して表示されます。
その結果、デフォルトではデータオブジェクトが表示ボリュームを満たします。

データ軸を手動スケーリングに変更し、その範囲を変更するか、データの範囲を変更した場合、データ軸は依然として ±1 の表示ボリュームを埋めますが、データオブジェクトは正確に収まらなくなります。

パスプロット

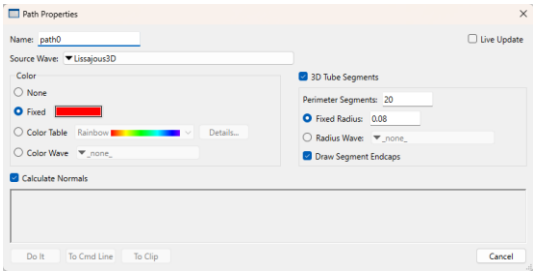


パスプロット内の各データポイントは、直線で隣接する各ポイントと順序通りに接続されます。

個々のポイントにマーカーを描画するには、同じデータウェーブに基づく散布図を追加する必要があります。

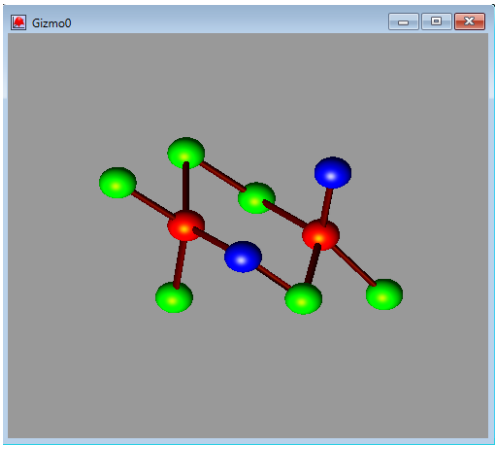
パスプロットには、各頂点の X、Y、Z 座標をそれぞれ表す 3 列の 2 次元ウェーブであるトリプレットが必要です。
パスプロットのカラーウェーブは、各行がデータウェーブの対応する頂点の色を指定する 3 次元ウェーブです。
カラーウェーブは 4 列で構成され、[0,1] の範囲の RGBA エントリを指定します。

Path Properties ダイアログは右のようになります。



パスオブジェクト用の ModifyGizmo で利用可能なオプションの完全なリストはヘルプ ModifyGizmo for Path Objects に記載されています。

Molecule デモエクスペリメント（File→Example Experiments →Visualization）では、化学結合を表示するために使われる 3D チューブセグメントの例を確認できます。



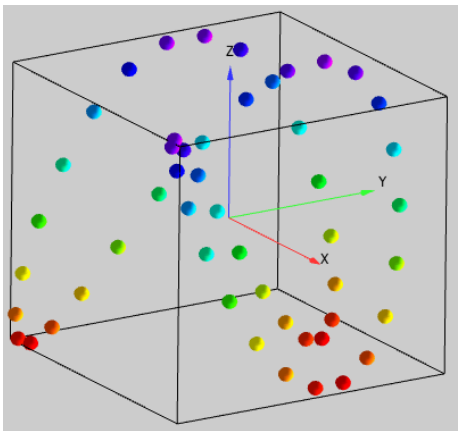
3D 散布図

散布図の各データポイントは 3D マーカーとして表示されます。

マーカーを接続したい場合は、同じトリプレットウェーブを使うパスプロットオブジェクトを追加する必要があります。

散布図にはトリプレットウェーブが必要で、これは各マーカーの X 座標、Y 座標、Z 座標それぞれに対応する 3 列からなる 2 次元ウェーブです。

各マーカーは Gizmo オブジェクトで表現できます。
組み込みの描画オブジェクト（ボックス、球体、円柱、円盤など）から選択することも、Object List 内の任意のオブジェクトを選択することもできます。



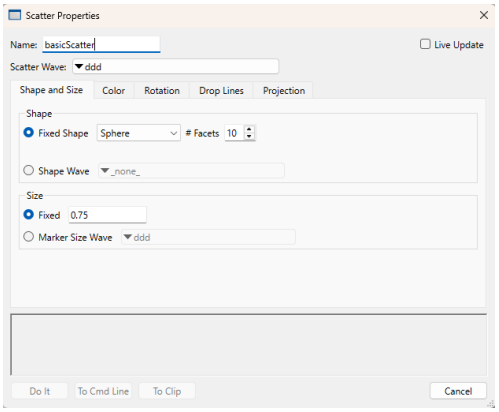
これは、ポイント数が非常に少なく、各マーカーを高解像度で表示したい場合に便利です。
その場合、必要な解像度を持つオブジェクトを Display List に追加し、パスプロットのマーカーとしてこのオブジ

ェクトを選択します。
例えば、Molecule デモエクスペリメントを参照してください。

散布オブジェクトのサイズを定数サイズとして指定するか、各散布ポイントのサイズ指定を含むウェーブを提供できます。
サイズはトリプレットウェーブによって指定され、各行には各マーカーの X、Y、Z 次元のスケーリング係数が含まれます。

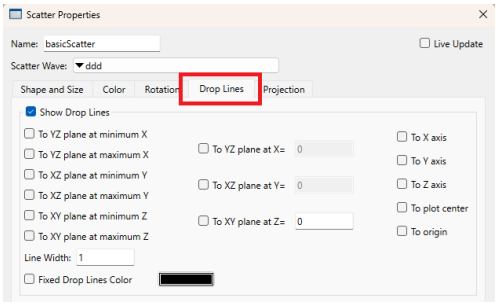
また、各散布ポイントの回転を指定することもできます。
回転は、４つの列を含む 2D ウェーブで指定されます。
最初の列は回転角度（度単位）を指定し、残りの３列は回転軸の正規化ベクトルを指定します。
これは rotate コマンドと同様です。

散布図は、一定の色、組み込みのカラーテーブルから取得した色、またはユーザー指定のカラーウェーブを使って色付けすることができます。
散布図のカラーウェーブは２次元ウェーブであり、各行はデータウェーブ内の対応する要素の色を指定します。
カラーウェーブは４列で構成され、[0,1] の範囲の RGBA エントリを指定します。



Scatter Properties ダイアログでは、オブジェクトのすべてのプロパティを設定できます。
オブジェクトが既に Display List にある状態でダイアログを使う場合、Live Update ボックスをチェックすると、変更を加えた瞬間にすべての変更を確認できます。

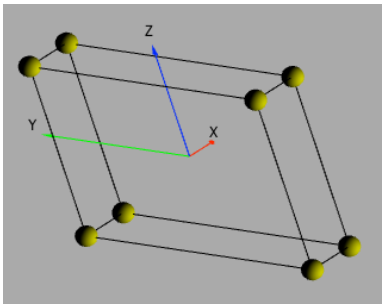
散布図の便利な機能の一つは、「ドロップライン」を描画できることです。
ドロップラインは各マーカーの中心から始まり、14 の可能なポイント、線、または平面のいずれかに伸びます。
Scatter Properties ダイアログの Drop Lines のタブから、ドロップラインの任意の組み合わせを選択できます。



散布図用の ModifyGizmo で利用可能なオプションの完全なリストはヘルプ ModifyGizmo for Scatter Objects に記載されています。

Gizmo では散布図を用いて結晶構造を表示できます。
例を以下に示します。

結晶は、トリプレットで指定された座標を中心に散らばるオブジェクトとして簡単に表示できます。
例えば、各中心に描画されるオブジェクトとして球体を使用できます。
追加のウェーブを指定して、各球の色やサイズをコントロールできます。

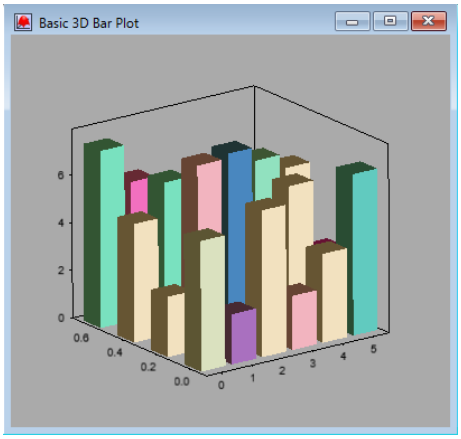


Igor は、トリプレットウェーブの結晶学座標から直交座標への変換、およびその逆変換を行う２つの組み込み変換を提供します。
Crystal Demo デモエクスペリメント（File→Example Experiments→Visualization→Advanced）と WaveTransform コマンドのキーワード crystalToRect と rectToCrystal を参照してください。

3D 棒グラフ

3D 棒グラフは、直線（３軸すべてに平行）の矩形の棒で構成されます。
棒は規則的なグリッド上に配置されることもあれば、ボリューム内に完全に散らばっていることもあります。

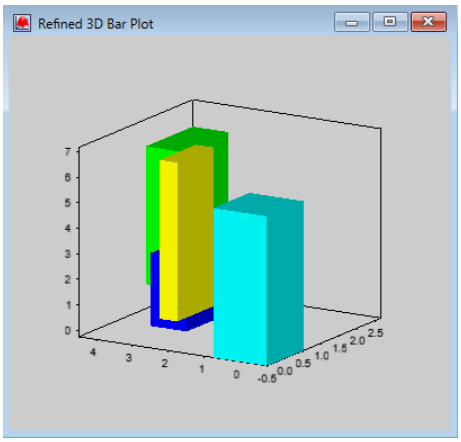
3D 棒グラフには、基本モードと洗練モードの２つのモードがあります。



基本的な 3D 棒グラフでは、すべての棒がゼロから始まり、正の Z 方向に向かって伸びています。

基本的な 3D 棒グラフは、正の値の２次元行列を表します。
各値は、行列内の行/列に応じてゼロベースの棒として表示されます。
すべての棒は同じ幅を持ちます。

洗練された棒グラフでは、棒を任意の位置に配置でき、すべての棒のサイズを自由にコントロールできます。



洗練された 3D 棒グラフの表示には、６列の入力ウェーブが必要です。
各行が１つの棒を表し、各列には以下の情報が含まれます：

列 0： 棒の X 中心
列 1： 棒の Y 中心
列 2： 下の Z 値
列 3： 上の Z 値
列 4： X 方向の棒の幅
列 5： Y 方向の棒の幅

洗練されたモードを使うことで、積み重ねや重なり合った 3D 棒グラフや、サイズが異なる棒グラフを作成することが可能です。

詳細を確認するには、3D Bar Plot Demo デモエクスペリメントを開いてください。

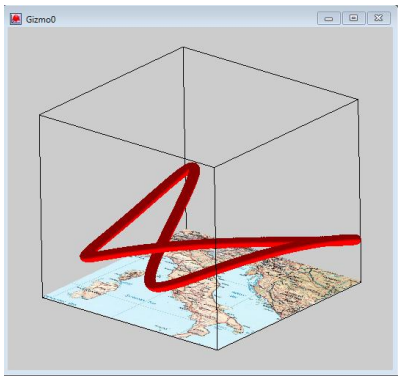
Gizmo 画像プロット

Gizmo 画像は、内部的に画像データによってテクスチャリングされた四角形オブジェクトの一形態です。

画像ソースウェーブは、次の３つの形式のいずれかです。

- Z 値の２次元行列
- ３層構造の符号なしバイト型 3D RGB ウェーブ
- ４層構造の符号なしバイト型の RGBA ウェーブ

後者の２つの形式は ImageLoad コマンドによって生成されます。



デフォルトでは、画像は表示ボリュームの下部に表示されますが、回転、平行移動、拡大縮小オプションを使うと、その内部の任意の位置に画像を配置できます。
データを基準に画像を登録する必要がある場合は、Gizmo 表示の軸範囲を変更するか、画像の拡大縮小を変更できます。
拡大縮小は画像の中心を基準に均一に行われます。

Igor には、密な散布図と Gizmo 画像オブジェクトを組み合わせた飛行経路の例が含まれています。
この場合、画像は地図で構成されています。

詳細を確認するには、Flight Path Demo デモエクスペリメント（File→Example Experoments→Visualization）を開いてください。

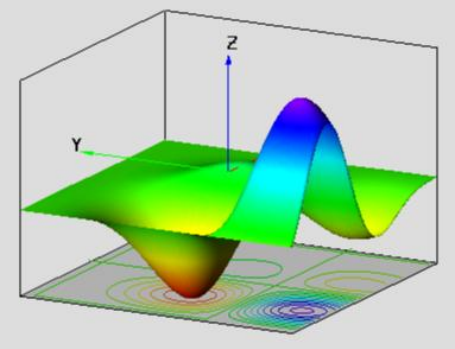
サーフェスプロット

サーフェスプロットは、データ値のグリッドを結ぶシートで構成されます。
サーフェスの色は、組み込みのカラーテーブルまたはカスタムカラーウェーブから指定できます。
詳細は「カラーウェーブ」のセクションを参照してください。
サーフェスに加えて、データ値は点またはグリッド線として表示することもでき、それらは独自の色指定が可能です。

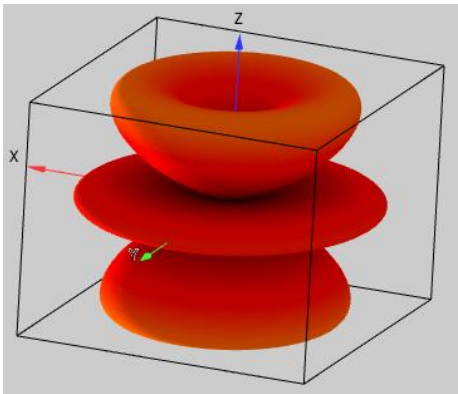
通常、データはサーフェスを構成する Z 値の MxN 2 次元行列で構成されます。
「サーフェスオブジェクトのデータ形式」のセクションを参照してください。
また、パラメトリックサーフェスもサポートされています。
これは 3D ウェーブであり、各レイヤーには X、Y、Z 値が順に含まれます。
詳細は「パラメトリックサーフェスのデータ形式」のセクションを参照してください。

利用可能なオプションの完全なリストは、ヘルプ ModifyGizmo for Surface Objects で説明されています。

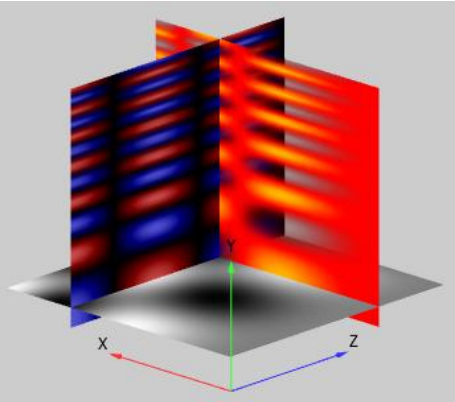
右の例は、下部にコンター図を備えたサーフェスプロットを示しています。



右の例は、球面調和関数のパラメトリックサーフェスプロットを示しています。



右の例はボリュームデータの直交スライスを示しています。



非直交スライスでサンプリングしたデータを表示するには、任意のパラメトリックサーフェスを作成し、その頂点でサンプリングしたデータを用いて着色します。
以下に例を示します。

次のような単純な 3D データセットがあると仮定します。
新しいエクスペリメントを作成したところから始めます。

```
Make/O/N=(100,100,100) ddd=z
```

このデータ範囲で球体を記述するパラメトリックサーフェスを作成します。
以下は GizmoSphere デモエクスペリメントにある MakeSphere 関数です。
（プロシージャウィンドウ内に記述します。）

```
Function MakeSphere(pointsx,pointsy)
        Variable pointsx,pointsy

        Variable i,j,rad
        Make/O/N=(pointsx,pointsy,3) sphereData
        Variable anglePhi,angleTheta
        Variable dPhi,dTheta

        dPhi=2*pi/(pointsx-1)
        dTheta=pi/(pointsy-1)
        Variable xx,yy,zz
        Variable sig

        for(j=0;j<pointsy;j+=1)
                angleTheta=j*dTheta
                zz=sin(angleTheta)
                if(angleTheta>pi/2)
                        sig=-1
                else
                        sig=1
                endif
                for(i=0;i<pointsx;i+=1)
                        anglePhi=i*dPhi
                        xx=zz*cos(anglePhi)
                        yy=zz*sin(anglePhi)
                        sphereData[i][j][0]=xx
                        sphereData[i][j][1]=yy
                        sphereData[i][j][2]=sig*sqrt(1-xx*xx-yy*yy)
                endfor
        endfor
End
```

関数は次のように実行します。

```
MakeSphere(100,100)
```

次に結果をサンプルデータの限界値にシフトします。

```
sphereData*=48          // 境界の少し内側
sphereData+=50
```

次に、パラメトリックサーフェスの頂点にデータのサンプルを含むスケールウェーブを作成します。

```
Make/N=(100,100)/O scaleWave
scaleWave=Interp3D(ddd,sphereData[p][q][0],sphereData[p][q][1],sphereData[p][q][2])
```

カラーウェーブを作成するには、まず Gizmo ウィンドウを開き、次に Gizmo にカラーウェーブを作成させます。

```
NewGizmo
AppendToGizmo surface=root:sphereData,name=surface0
```

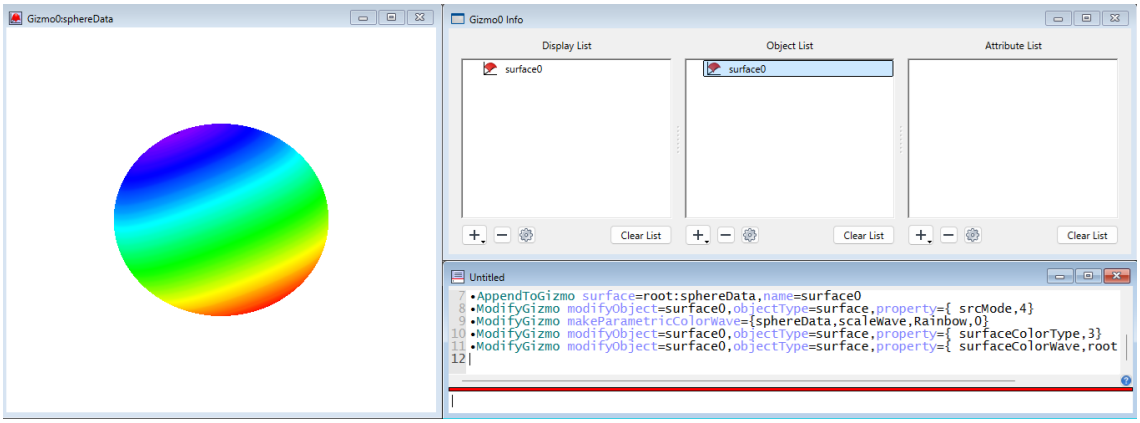
```
ModifyGizmo modifyObject=surface0,objectType=surface,property={ srcMode,4}
ModifyGizmo makeParametricColorWave={sphereData,scaleWave,Rainbow,0}
```

カラーウェーブをサーフェスに適用します。

```
ModifyGizmo modifyObject=surface0,objectType=surface,property={ surfaceColorType,3}
```

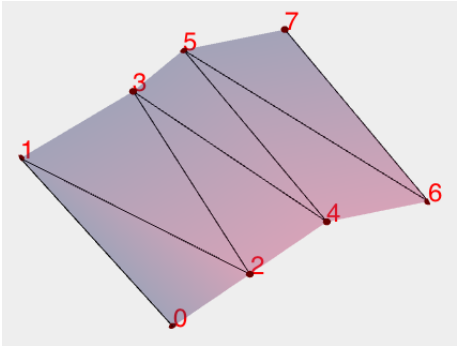
```
ModifyGizmo modifyObject=surface0,objectType=surface,
        property={ surfaceColorWave,root:sphereData_C}
```

最後に Object List から Display List に surface0 を移動すると、次のようになります。



リボンプロット

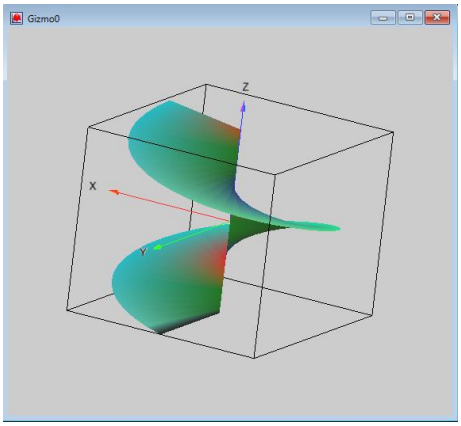
リボンプロットでは、データポイントはリボンオブジェクトを定義するサーフェスによって接続されます。
リボンは、ここに示すように頂点を交互に配置した三角形のリストから構成されます。



個々のポイントや接続を表示するには、散布図またはパスプロットを追加する必要があります。

リボンプロットのデータは、N×3 の値の行列で構成されます。
各行には、リボンの縁の上のポイントの位置座標である X、Y、Z 値が含まれます。
リボンの交互の縁の位置座標は、順番に続きます。
リボンの頂点の順序は、右の図に示されています。

リボンは少なくとも４つの頂点を持たなければならず、頂点の総数は偶数でなければなりません。

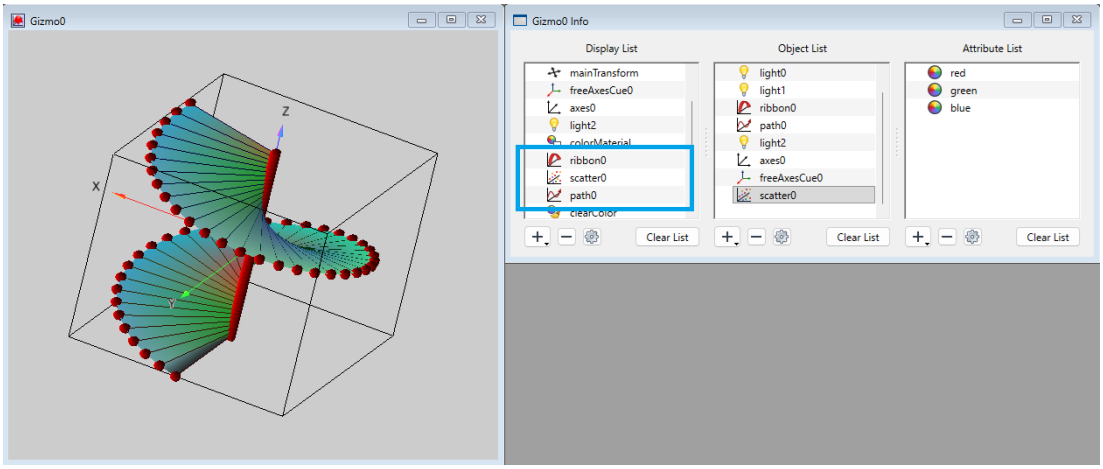


リボンプロットは、定数色、組み込みカラーテーブルから選択した色、またはユーザー指定のカラーウェーブを使って色付けできます。
リボンプロットのカラーウェーブは２次元ウェーブであり、各行はデータウェーブの対応する要素の色を指定します。
カラーウェーブは４列で構成され、[0,1] の範囲の RGBA エントリを指定します。

利用可能なオプションの完全なリストは、ヘルプ ModifyGizmo for Ribbon Objects に記載されています。

右はシンプルなリボンプロットの例です（File→Example Experiments→Visualization→Ribbon）。

次は、リボンの縁に沿ったデータポイントの位置を示すために、同じデータに対するパスプロットおよび散布図を重ねたリボンプロットの例です。



ボクセルグラムプロット

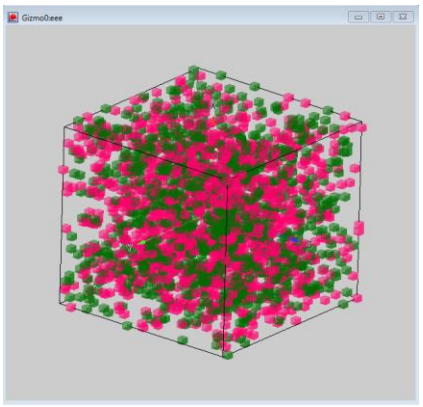
「ボクセル」は「ボリューム要素」の略称です。

ボクセルグラムは、特定の値を含むウェーブの要素を色で示す 3D ウェーブの表現です。
１～５個の値を指定し、各値に関連付けられた RGBA カラーを指定します。
指定された許容誤差内で、特定のウェーブの要素の値が指定された値のいずれかと一致する場合、Gizmo はその要素を関連付けられた RGBA カラーで表示します。
ウェーブの要素が指定されたレベルのいずれとも一致しない場合、その要素はまったく表示されません。

ボクセルは立方体または点で表現できます。

利用可能なオプションの完全なリストは、ヘルプ ModifyGizmo for Voxelgram Objects に記載されています。

このデモエクスペリメントは、２つの指定値を使うボクセルグラムを示しています（File→Example Experiments→Visualization→Ribbon）。

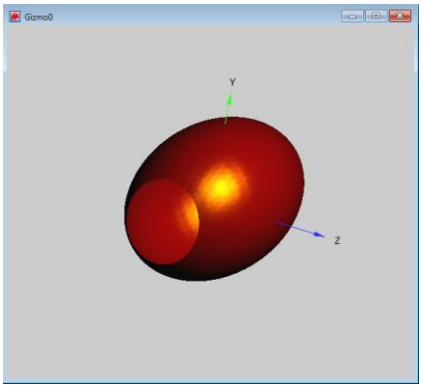


等値面プロット

等値面は、２次元コンター（等高線）の３次元版です。

右は等値面の例です（File→Example Experiments→Visualization→IsoSurfaceDemo）。

等値面の生成は、体積を四面体に分割するため、計算負荷が大くなります。
いずれかの四面体で等値が検出されると、その交点（１つまたは２つの三角形）が計算されます。
等値面自体は、これらの三角形の集合体で構成されます。



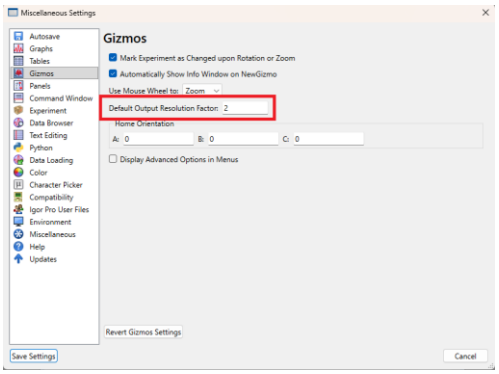
等値面は 3D ボリュームウェーブから構築されます。

利用可能なオプションの完全なリストは、ヘルプ ModifyGizmo for Isosurface Objects に記載されています。

Gizmo ウィンドウの印刷

Gizmo 表示ウィンドウが前面にある状態で、File→Print Gizmo を選択すると、その画像を印刷できます。

ウィンドウは画面解像度の倍数で印刷されます。
この倍率は、Misc メニューからアクセスできる Miscellaneous Settings ダイアログの Gizmo セクションにある Default Output Resolution Factor でコントロールされます。
この倍率の製品出荷時のデフォルト値は 2 です。



デフォルトの印刷解像度を上書きするには、次のコマンドを実行しても行えます。

```
ModifyGizmo outputResFactor = n
```

ここで n は正の整数（通常は 1、2、4、8）です。
これはアクティブなウィンドウにのみ適用されます。
これは後続の印刷に影響し、デフォルトの Default Output Resolution Factor（出力解像度係数）の設定を上書きします。
この設定は、Gizmo ウィンドウの再作成マクロには保存されないため、永続的ではありません。
機能する n の最大値は、グラフィックスハードウェアのビデオメモリ（VRAM）の容量によって異なります。

オブジェクトのアンチエイリアシングによって出力も改善できます。
これを行うには、Display List に GL_SRC_ALPHA と GL_ONE_MINUS_SRC_ALPHA を使ったブレンド関数を追加し、GL_LINE_SMOOTH を使った有効化コマンドを追加します。
また、GL_POINT_SMOOTH を有効にして、散布オブジェクト内のポイントを滑らかにすることもできます。

Gizmo から画像を印刷できない場合、VRAM が不足している可能性があります。
これによりグラフィックスが空白または歪んで表示されることがあります。
試すべき対策は以下の通りです。

- 開いている他の Gizmo ウィンドウをすべて閉じる。
- Gizmo 表示ウィンドウのサイズを小さくする。
- Default Output Resolution Factor で設定された解像度、または ModifyGizmo outputResFactor を使って解像度を下げる。
- 複数のモニターを接続したシステムで作業している場合は、表示ウィンドウを VRAM 容量が最も大きいグラフィックスカードで動作しているモニターに移動する。
- より多くの VRAM を搭載したハードウェアでエクスペリメントを実行する。

Gizmo ウィンドウのエクスポート

Gizmo プロットは、以下のいずれかの方法でエクスポートできます。

- File→Save Graphics を選択し、PNG、JPEG、または TIFF ファイルにエクスポートする。
  これにより SavePICT コマンドが生成される。
- Edit→Export Graphics を選択し、クリップボードに PNG、JPEG、または TIFF 形式でエクスポートする。
- Edit→Copy を選択する。
  これにより、Export Graphics ダイアログで最後に設定された設定を使ってクリップボードにエクスポートされる。

- 右クリックし、ポップアップメニューから　Copy to Clipboard を選択する。
  これは Edit→Copy を選択するのと同じ。

ExportGizmo コマンドは、下位互換性のためだけに利用可能です。
このコマンドは廃止予定であり、代わりに SavePICT を使ってください。

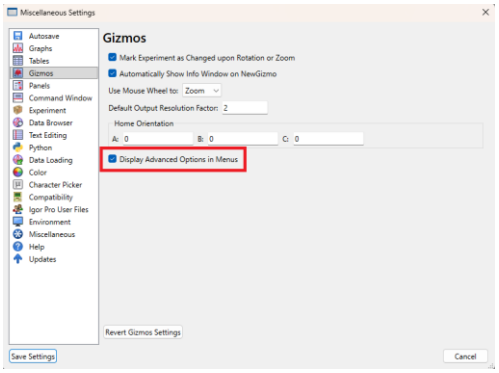
Export Graphics ダイアログと SavePICT コマンドでは、出力解像度を画面解像度の倍数としてコントロールできます。
高解像度でのエクスポートには十分なビデオメモリ（VRAM）が必要です。
ほとんどのハードウェアは２倍（画面解像度の２倍）をサポートしています。
利用可能な VRAM によっては、さらに解像度を上げられる場合があります。

Gizmo から画像をエクスポートできない場合、VRAM が不足している可能性があります。
これによりグラフィックスが空白または歪むことがあります。
試すべき対策は以下の通りです。

- 開いている他の Gizmo ウィンドウをすべて閉じる。
- Gizmo 表示ウィンドウのサイズを小さくする。
- Export Graphics または Save Graphics ダイアログで設定された解像度を下げる。
- 複数のモニターを接続したシステムで作業している場合は、表示ウィンドウを VRAM 容量が最も大きいグラフィックスカードで動作しているモニターに移動する。
- より多くの VRAM を搭載したハードウェアでエクスペリメントを実行する。

高度な Gizmo のテクニック

多くの高度な Gizmo オプションは、デフォルトでは非表示になっています。
これらを表示するには、Misc メニューからアクセスできる Miscellaneous Settings ダイアログの Gizmo セクションにある Display Advanced Options Menus チェックボックスをオンにする必要があります。
ほとんどの Gizmo ユーザーはこれらのオプションを必要としません。



Group（グループ）オブジェクト

Gizmo グループオブジェクトは、既存の Gizmo オブジェクトとコマンドをカプセル化したものであり、最前面の Gizmo 内で１つのオブジェクトとして扱われます。

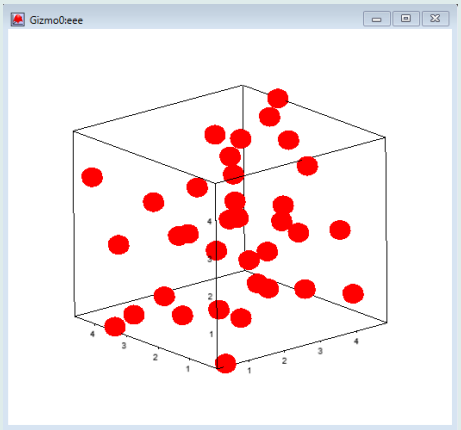
グループオブジェクトを操作するには、高度な Gizmo オプションを有効にする必要があります。
このセクションの例を試す場合は、Miscellaneous Settings ダイアログ（Misc メニュー）の Gizmo セクションで、Display Advanced Options in Menus チェックボックスがオンになっていることを確認してください。

次の例では、グループオブジェクトを使って散布図用のカスタムマーカーを作成する方法を示しています。

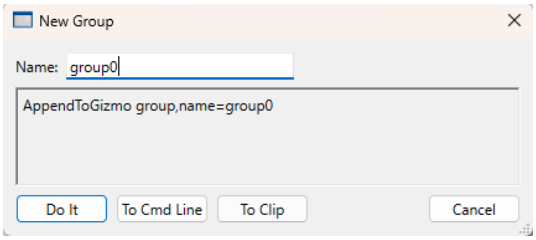
1. 新しいエクスペリメントを作成して、コマンドウィンドウで次を実行します：

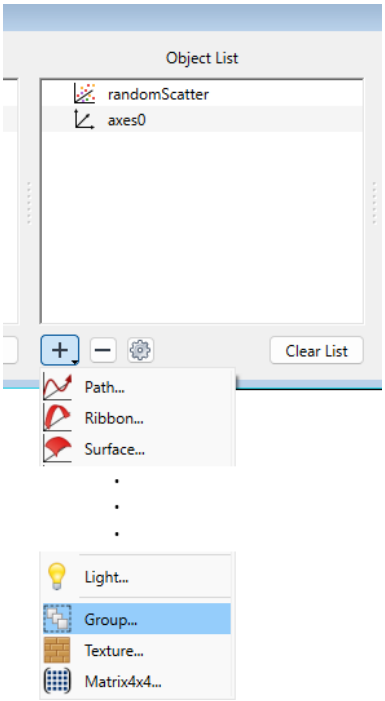
```
NewGizmo/JUNK=3
```

これはデフォルトの赤い球体マーカーを使ったシンプルな散布図を作成します。

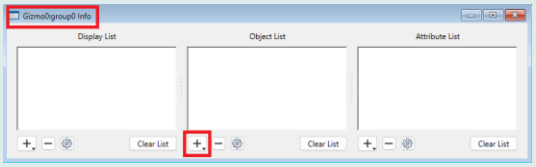


2. 情報ウィンドウの Object List の下にある「+」アイコンを使って Group オブジェクトを作成します。
（名前はデフォルトのままとします）

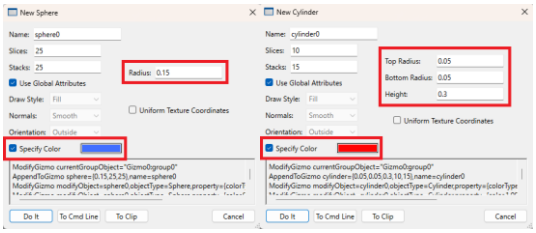




3. Object List 内の group0 オブジェクトをダブルクリックします。
これにより、グループ用の新しい情報ウィンドウが開き、Display List、Object List、Attribute List が表示されます。

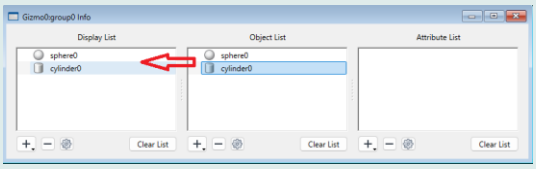


4. 半径 0.15 の青い球体オブジェクトと、上下の半径が 0.05、高さが 0.3 の赤い円柱オブジェクトをグループに追加します。



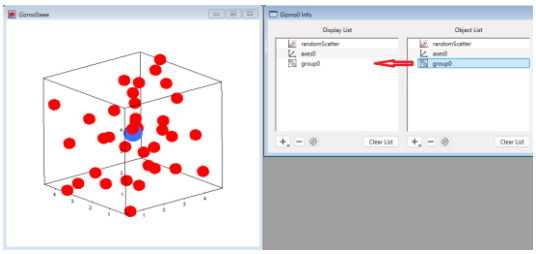
5. 球体オブジェクトと円柱オブジェクトをグループの Display List にドラッグします。

group0 Info ウィンドウを閉じます。



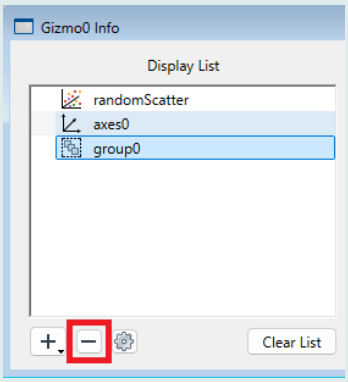
6. Gizmo0 Info ウィンドウで、Object List から group0 オブジェクトを Display List の下部にドラッグします。

球体+円柱グループが Gizmo プロットに表示されます。



7. Gizmo0 Info ウィンドウの Display List で group0 オブジェクトを選択し、「-」アイコンをクリックして削除します。

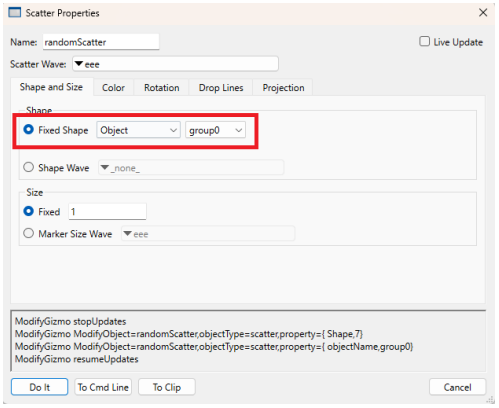
球体+円柱グループが Gizmo プロットから消えます。



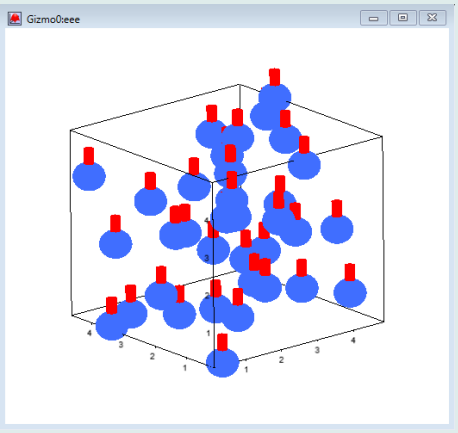
8. Gizmo0 Info ウィンドウで、randomScatter オブジェクトをダブルクリックして Scatter Properties ダイアログを開きます。

ダイアログの Shape and Size タブで、Fixed Shape ポップアップメニューから Object を選択し、そのすぐ右にあるポップアップメニューから group0 を選択します。

Do It をクリックします。



結果として得られるグラフは、おおよそ右のようなものになるはずです。



散布オブジェクトの類似例は、マーカーオブジェクトを円筒形の本体と円錐形の先端を持つ線としたデモエクスペリメント Scatter Arrows（File→Example Experiments→Visualization→Advanced）で見ることができます。

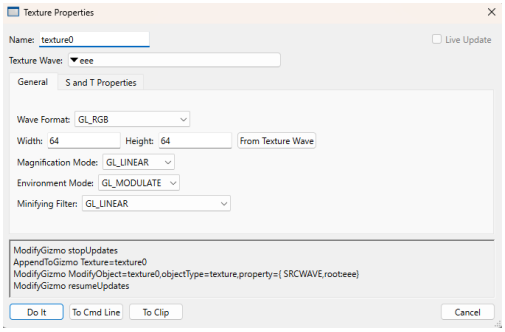
Texture（テクスチャ）オブジェクト

テクスチャは OpenGL で広く使われるため、ほとんどのグラフィックスハードウェアはテクスチャ使用に最適化されています。

テクスチャの詳細な説明は本ドキュメントの範囲外であるため、ここでは Gizmo におけるテクスチャの使用に焦点を当てます。
テクスチャオブジェクトを操作するには、Miscellaneous Settings ダイアログで高度な Gizmo メニューを有効にする必要があります。

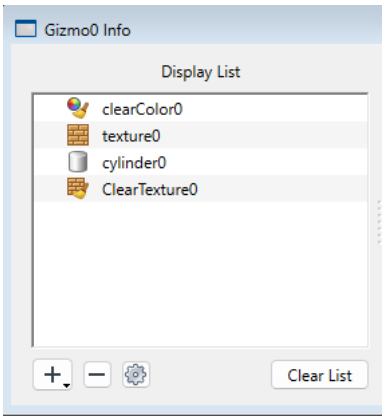
Gizmo のテクスチャオブジェクトは、任意の形状の表面に画像情報を適用できる OpenGL オブジェクトを表します。
Gizmo のテクスチャは 1D または 2D です。
これらは四角形オブジェクト、二次サーフェスオブジェクト（球体、円柱、円盤）、およびパラメトリックサーフェスに適用できます。
単純な四角形にテクスチャを使う場合は、代わりに画像プロットを使うべきです。
詳細は「Gizmo 画像プロット」のセクションを参照してください。

通常、テクスチャは ImageLoad で読み込まれた画像から作成され、各ピクセルの色が順次レイヤーに格納された符号なしバイト 3D RGB ウェーブ形式となります。
Gizmo テクスチャを作成する前に、標準画像形式を 1D ウェーブに変換する必要があります。
この形式では、各ピクセルの色エントリが RGB または RGBA として順次格納されます。
この変換は、キーワード imageToTexture を指定した ImageTransform コマンドで実現できます。



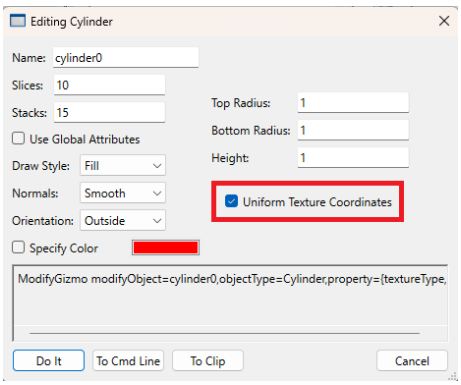
Texture Properties ダイアログは、テクスチャウェーブがテクスチャオブジェクトに変換される方法と、描画におけるテクスチャの適用方法を決定します。

ダイアログを呼び出す前に、テクスチャウェーブの次元、そのパッキング形式、および元の画像サイズを知っておくと便利です。
ImageTransform は、この情報をテクスチャウェーブのウェーブノートに保存します。
ダイアログは、From Texture Wave ボタンをクリックすると、ウェーブノートからテクスチャ情報をロードします。

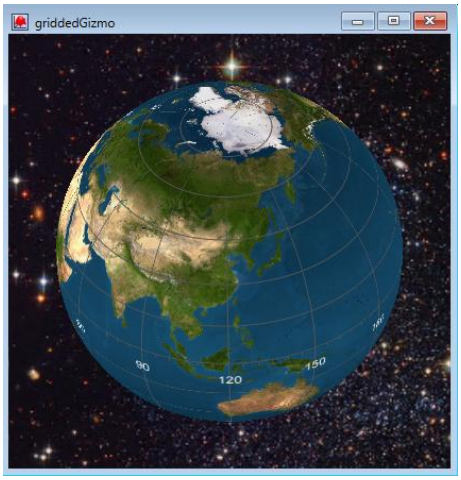


テクスチャオブジェクトを作成したら、他のオブジェクトに適用できます。
例えば、cylinder0 に texture0 を適用するには、Display List に右の並びを含める必要があります。

cylinder0 の前に texture0 が配置され、その後には clearTexture0 が続きます。
Cylinder Properties ダイアログで Uniform Texture Coordinates チェックボックスを必ずオンにしてください。
テクスチャオブジェクトはシリンダーの直前に配置され、クリアテクスチャコマンドが続くため、以降の描画ではテクスチャが適用されません。



GizmoEarth デモエクスペリメント（File→Example Experiments→Visualization→Advanced）はテクスチャの使用例を示しています。
この場合、高解像度の矩形テクスチャが球体を表すパラメトリックサーフェスに追加されます。
このパラメトリックサーフェスは 61×61 の頂点で構成されています。



Matrix4x4 オブジェクト

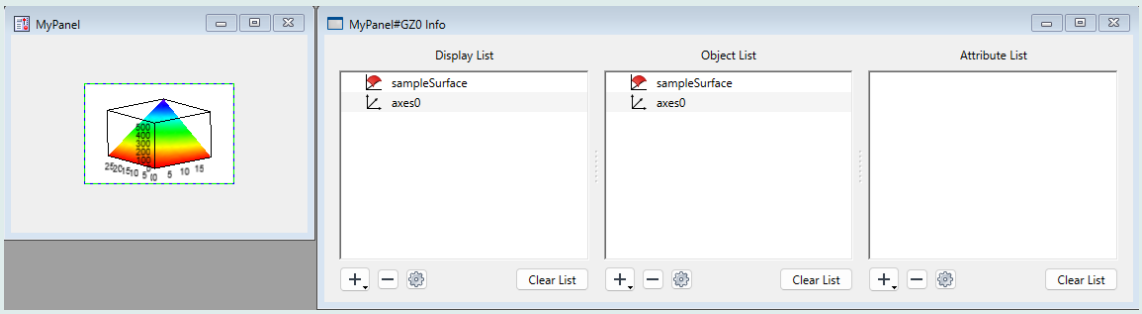
Matrix4x4 オブジェクトは、変換行列に影響を与える Gizmo コマンドで使われます。
これは４行４列の 2D ウェーブをカプセル化します。

Gizmo サブウィンドウ

Gizmo サブウィンドウをグラフ、パネル、またはレイアウトウィンドウに埋め込むことができます。
以下に簡単な例を示します。

1. 新しいエクスペリメントを作成して、コマンドウィンドウで次を実行します：

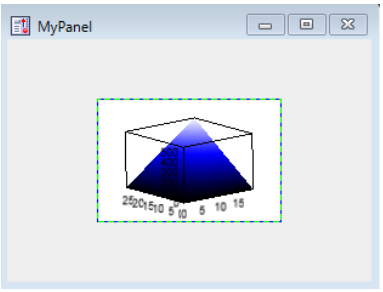
```
NewPanel/N=MyPanel
NewGizmo/HOST=MyPanel/JUNK=2
```



2. ModifyGizmo コマンドは、完全なサブウィンドウ指定を使って実行できます。

例えば、デモサーフェスのカラーマップを変更するには、次を実行します。

```
ModifyGizmo/N=MyPanel#GZ0 modifyObject=sampleSurface, objectType=surface,
        property={surfaceCTab,Blue}
```



詳細についてはヘルプ Subwindow Syntax を参照してください。

Gizmo のトラブルシューティング

このセクションでは、Gizmo の使用中に発生する可能性のある問題の解決方法について説明します。

Gizmo の光源に関する問題のトラブルシューティング

1. 光源オブジェクトが、照らされる予定のオブジェクトよりも、Display List で上に表示されていることを確認してください。
2. 光源の設定で、Ambient、Diffuse、Specular の各コンポーネントが黒以外の色を指定していることを確認してください。
3. 方向性光源から始めます。
   位置指定ライトよりも設定項目が少なく、問題が発生しにくいためです。
   Light Properties ダイアログを開き、Live Update チェックボックスをオンにします。
   方向のコントロールを変更し、シーンの変化を確認してください。
   シーンに変化が見られない場合、光源の方向が問題である可能性は低いでしょう。
4. 照明を適用する予定のオブジェクトごとに、Compute Normals チェックボックスがオンになっていることを確認してください。
5. 通常の法線方向が期待通りであることを確認してください。
   二次サーフェスオブジェクトの場合、法線方向を内側と外側の間で切り替えます。
   その他のオブジェクトでは、前面と背面の定義を考慮してください。
   疑問がある場合は、Display List 内でオブジェクトの直前に前面操作を追加してください。
   この操作により、前面と背面を表す方向を定義できます。
6. 通常のスケールでは問題なく見えるオブジェクトが、スケーリング操作後に描画されると照明効果が不適切に見える場合は、最終サイズでオブジェクトを計算し、スケーリングを避ける必要があります。
   例えば、散布図では、マーカーはデフォルトで等方性スケーリング係数 1（スケーリングなし）で描画されます。
   散布図で使う半径 1 の球体オブジェクトを作成する場合、どこかでサイズを縮小する必要があります。
   これには、Scatter Properties ダイアログでスケールを設定するか、球体の半径を縮小する方法があります。

Gizmo の透明度に関する問題のトラブルシューティング

1. Display List を確認し、透明オブジェクトの上位に GL_BLEND 有効化コマンドが存在すること、およびブレンド関数が存在し正しいパラメーターが設定されていることを確認してください。
   詳細は「透明度と半透明」のセクションを参照してください。
2. オブジェクトの描画順序を確認してください。
   透明オブジェクトは、Display List において不透明オブジェクトの後ろに配置する必要があります。
3. 透明な表面に重なり合う面がある場合、その表面を三角形に分解し、深度順に並べるしか方法はありません。
   この例は DepthSortingDemo（File→Example Experiments→Visualization→Advanced）デモエクスペリメントで見ることができます。

Gizmo の透明度に関する問題のトラブルシューティング

オブジェクトまたはオブジェクトの一部がクリップされたり表示されない場合は、以下のヒントを試してください。

1. メニュー Gizmo→Axis Range を使って、すべての軸の軸範囲を自動に設定します。
2. Display List にクリッピング平面が含まれていないことを確認してください。
3. 対象オブジェクトが、Display List 内で平行移動、回転、または拡大縮小コマンドの後に続いていないことを確認してください。
4. ズームレベルと主な変換を確認してください。
   Display List に変換コマンドがない場合は、Display List の先頭にデフォルトの正射投影コマンド（全軸で ±2）を挿入してください。

Gizmo の互換性

Igor 7 以前の Gizmo は、OpenGL 1.0 仕様に基づいて実装されていました。
OpenGL の変更、特に OpenGL 3.0 以降の変更により、Gizmo の様々な機能を修正する必要が生じました。
古い Igor エクスペリメントにおける Gizmo ウィンドウは、エラーなく開くよう、また可能な限り元の形状に近い状態で表示されるよう努めました。
主な例外として、文字列オブジェクトの向き、軸ラベル、目盛ラベルの表示があります。
詳細は「Gizmo テキストの変更点」のセクションを参照してください。

Gizmo リファレンスドキュメントでは、一部の機能を「Obsolete/廃止」または「Deprecated/非推奨」としてマークしています。
「Obsolete」とは、その機能がサポートされなくなったことを意味します。
「Deprecated」とは、将来の Igor バージョンで削除される可能性があるため、可能な場合は代替機能を使うべきであることを意味します。

ユーザー定義関数内の Gizmo コマンド

Igor 7 以降では、ユーザー定義関数内で Gizmo コマンドを使用できます。
この機能をサポートするため、コマンド構文にいくつかの変更が必要でした。

主な構文変更は、ModifyGizmo への objectType キーワードの追加です。
Igor 7 より前では以下のように書くことができました。

```
ModifyGizmo modifyObject=quad0, property={calcNormals,1}
```

Igor 7 以降の構文では、次のように objectType=<type> キーワードを追加する必要があります:

```
ModifyGizmo modifyObject=quad0, objectType=Quad, property={calcNormals,1}
```

Igor 6 によって作成された Gizmo 再作成マクロを解釈できるようにするため、objectType キーワードはユーザー定義関数内でのみ必要であり、コマンドラインやマクロ内では不要です。

後方互換性のため、Igor Pro 6.3 の Gizmo XOP は objectType キーワードを受け付けます。

Gizmo 再作成マクロの変更

Igor 7 より前のすべての Gizmo 再作成マクロには次の行が含まれていました。

```
ModifyGizmo startRecMacro
```

Igor 7 以降では、構文にバージョン番号が含まれます。

```
ModifyGizmo startRecMacro=700
```

後方互換性のため、Igor Pro 6.3 の Gizmo XOP は新しい startRecMacro 構文を受け入れます。

Igor 7 以降では、Gizmo は反時計回りを前面のデフォルトとして使います（ただし、以前のバージョンの Gizmo 再作成マクロを実行する場合を除く）。
ほとんどのユーザーには影響しないと思われますが、影響がある場合は、Gizmo Display List 内で明示的な frontFace コマンドを指定することでこのデフォルトを上書きできます。

Gizmo コマンド内での GL 定数の使用

以前のバージョンの Gizmo では、コマンド内で OpenGL 定数の使用がサポートされていました。
例えば、次のように記述できました。

```
AppendToGizmo attribute blendFunc={GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA}, name=blend0
```

Igor 7 以降では、GL_SRC_ALPHA などの GL 定数は認識されず、代わりに数値で指定する必要があります。
例えば：

```
AppendToGizmo attribute blendFunc={770,771}, name=blend0
```

GetGizmo を実行することで、特定の定数を表す数値を特定できます。
例えば：

```
GetGizmo constant="GL_SRC_ALPHA"
```

これにより履歴に「770」が出力されます。
出力された数値をコピーして、プロシージャに貼り付けることができます。

Gizmo グラフィックスのエクスポート

Igor 7 以降では、File→Save Graphics を使って Gizmo グラフィックスをエクスポートでき、これにより SavePICT コマンドが生成されます。
ExportGizmo コマンドは、ある程度の後方互換性のために部分的にのみサポートされています。
クリップボードまたは Igor ウェーブへのエクスポート、および印刷は可能ですが、ファイルへのエクスポートはできなくなりました。
代わりに SavePICT を使ってください。

Gizmo テキストの変更

Igor 7 より前の Gizmo では、軸ラベル、目盛ラベル、および文字列オブジェクトを、各文字ごとに 3D 塗りつぶしポリゴンを構成することで表示していました。
このポリゴン表現により、Gizmo は各文字列を真の 3D オブジェクトとして扱い、軸の向きに依存せず表示領域内の任意の位置に描画することが可能でした。
この手法には3つの欠点がありました：

- ラベルは常に判読可能とは限らず、向きによっては完全に消えてしまうこともあった。
- ポリゴンへの変換により、アンチエイリアシングを用いてキャラクターを滑らかにすることが現実的ではなくなった。
- フォントサイズがプラットフォーム間で一貫していなかった。

Igor 7 以降では、Gizmo を 2D の滑らかなテキストを生成する新しいテキストレンダリング技術に移行しました。
この技術は前述の3つの問題に対処しますが、以前のバージョンで生成されたテキストオブジェクトと一致するものは生成しません。
その結果、軸ラベルなどテキストを使う Igor 7 より前のエクスペリメントを読み込む時に、オフセットや回転といったサポートされなくなった要素に差異が生じます。

Igor 7 以降では、書式付きテキストを使って複雑な軸ラベルを作成できます。
Display List または Object List で Axis 項目をダブルクリックし、Axis Properties ダイアログを表示して、Axis Label タブをクリックします。

Igor 7 以降では、Gizmo→Add Annotation を使って、テキストボックスやカラースケールを含む標準的な Igor 注釈を Gizmo ウィンドウに追加できます。
これらの注釈は 3D グラフィックスの前面にオーバーレイ表示され、グラフウィンドウ内の注釈と同様の動作をします。

その他の Gizmo の変更

Igor 7 以降では、shininess 属性の引数が front 値と back 値に変更されました。

Igor 7 以降では、Gizmo オブジェクトはオプションで内部で色属性を持ちます。
オブジェクトを作成する時に、色を指定するか、指定しないかを選択できます。
色を指定すると、Gizmo はそのオブジェクト用のデフォルト色素材を作成します。
デフォルト色素材は、GL_FRONT_AND_BACK と GL_AMBIENT_AND_DIFFUSE 設定を持ちます。
色を指定しない場合、Gizmo はデフォルト色素材を作成せず、ユーザー自身が色素材を作成する必要があります。
この色素材は、デフォルト色素材を持たない Display List 内の後続オブジェクトすべてに影響します。
この変更は、光沢のある表面の作成をサポートするために必要でした。

Gizmo のフック関数

このセクションは上級プログラマーのみを対象としています。

フック関数は、特定のイベントが発生したときに Igor によって呼び出されるユーザー定義関数です。
これにより、プログラマはイベントに対応し、場合によっては Igor の動作を変更することができます。
ウィンドウフック関数は、特定のウィンドウ内のイベントに対して呼び出されるフック関数です。

この機能に対する Igor のサポートはヘルプ Window Hook Functions で説明されており、Igor 7 以降では Gizmo を含む他の種類のウィンドウにも適用されます。

Gizmo は以前 XOP として実装されていたため、Igor のメカニズムとは別の独自のフック関数メカニズムを持っています。
このセクションでは、Gizmo 固有のフック関数サポートについて説明します。

Gizmo ウィンドウには、Igor フック関数、Gizmo フック関数、またはその両方、のどれかを使用できます。
ただし、両方を使うと混乱を招く可能性があります。
特定の Gizmo ウィンドウに Igor フック関数と Gizmo フック関数の両方をインストールした場合、Igor フック関数が最初に呼び出されます。

Igor 本体と同様に、Gizmo も当初は ModifyGizmo の hookFunction キーワードによってインストールされる 1 つのウィンドウフック関数しか持っていませんでした。
その後、ModifyGizmo の namedHook によってインストールされる名前付きフック関数が追加されました。
名前なしフックは廃止されました。名前付きフックの使用を推奨します。

Gizmo 名前付きフック関数

名前付き Gizmo ウィンドウフック関数は、1 つのパラメーター（WMGizmoHookStruct 構造体）を受け取ります。
この組み込み構造体は、様々なウィンドウイベントの状態に関する情報を関数に提供します。

名前付き Gizmo フック関数は、namedHook または namedHookStr キーワードを使って ModifyGizmo でインストールします。
hookEvents キーワードは名前付きフック関数には関係ありません。

名前付きフックを削除するには、空のプロシージャ名を使います。
例えば、次のようにします。

```
ModifyGizmo namedHook={hookName,$""}
```

フック関数は通常、0 を返す必要があります。
マウスホイールのフックイベントの場合、0 以外の値を返すと、ホイール操作に対する Gizmo の回転が妨げられます。

指定されたウィンドウフック関数は以下の形式を持ちます。

```
Function MyGizmoHook(s)
        STRUCT WMGizmoHookStruct &s

        strswitch(s.eventName)
                case "mouseDown":
                        break
                case "mouseMoved":
                        break
                case "rotation":
```

```
                        break
                case "kill":
                        break
                case "scaling":
                        break
        endswitch

        return 0
End
```

構造体の詳細については、ヘルプ WMGizmoHookStruct を参照してください。

現時点で以下のイベント名が定義されています：mouseDown、mouseMoved、rotation、killed、scaling。

例えば名前付き Gizmo ウィンドウフックを使った例については、GizmoWindowHook デモエクスペリメント（File→Example Experiments→Visualization→Advanced）を開き、GizmoRotation.ipf プロシージャファイル内の GizmoRotationNamedHook 関数を確認してください。
これはパックされたプロシージャファイルです。
独立モジュール内に存在するため、表示するには独立モジュールを有効にする必要があります。

Gizmo 名前なしフック関数

名前なしフックは非推奨ですが、下位互換性のため引き続きサポートされています。
名前付きフックを使ってください。
以下のドキュメントは歴史的参考情報としてのみ提供されます。

各 Gizmo ウィンドウには、1 つだけ名前なし Gizmo ウィンドウフック関数を設定できます。
名前なしウィンドウフック関数を指定するには、hookFunction キーワードを付けて ModifyGizmo コマンドを使います。

名前なしフック関数は、様々なウィンドウイベントが発生したときに呼び出されます。
フック関数が呼び出された理由は、フック関数の infoStr パラメーターにイベントコードとして格納されます。
特定のイベントは、hookEvents キーワードを指定した ModifyGizmo コマンドを使って有効にする必要があります。

名前なしフック関数の構文は以下の通りです。

```
Function functionName(infoStr)
        String infoStr

        String event = StringByKey("EVENT",infoStr)
        ...
        return 0        // 返値は無視される
End
```

infoStr は、セミコロンで区切られたキー・値のペアのリストを含む文字列です。

EVENT:eventKey	eventKey はイベントを識別します（下記参照）。	
	mousedown	マウスボタンがクリックされた
	mousemoved	マウスが移動した
	rotation	Gizmo のウィンドウが回転した
	kill	Gizmo のウィンドウが閉じられた
	scaling	Gizmo がウェーブベースのデータオブジェクトのデータ範囲に変化を検知した。

MOUSEX:x	x はマウスカーソルの X 座標（単位：ピクセル）です。
MOUSEY:y	y はマウスカーソルの Y 座標（単位：ピクセル）です。
EULERA:angle	angle は X 軸のオイラー角（単位：度）です。
EULERB:angle	angle は Y 軸のオイラー角（単位：度）です。
EULERC:angle	angle は Z 軸のオイラー角（単位：度）です。
WHEELX:dx	dx はマウスホイールの水平方向の変化量です。
WHEELY:dy	dy はマウスホイールの水平方向の変化量です。
XMIN:val	val はデータオブジェクトの X 値の全範囲における最小値です。
XMAX:val	val はデータオブジェクトの X 値の全範囲における最大値です。
YMIN:val	val はデータオブジェクトの Y 値の全範囲における最小値です。
YMAX:val	val はデータオブジェクトの Y 値の全範囲における最大値です。
ZMIN:val	val はデータオブジェクトの Z 値の全範囲における最小値です。
ZMAX:val	val はデータオブジェクトの Z 値の全範囲における最大値です。
WINDOW:winName	Gizmo ウィンドウの名前です。