CONTENTS

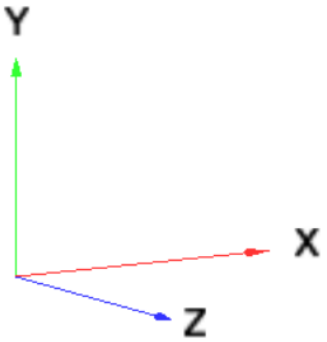
ビジュアルヘルプ – 3D グラフィックス（2）

Gizmo 表示環境

Gizmo は 3D グラフィックスを表示するための環境を構成します。
このセクションでは、その環境の主な特性について説明します。

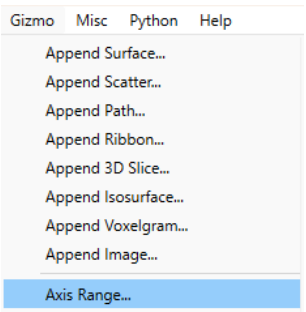
Gizmo 座標系

Gizmo は右手の法則に基づく 3 次元座標系を使用します。
X 軸の正方向が右を指し、Y 軸の正方向が上を指す場合、右手の法則により Z 軸の正方向は画面から外側、つまりあなたの方へ向かいます。



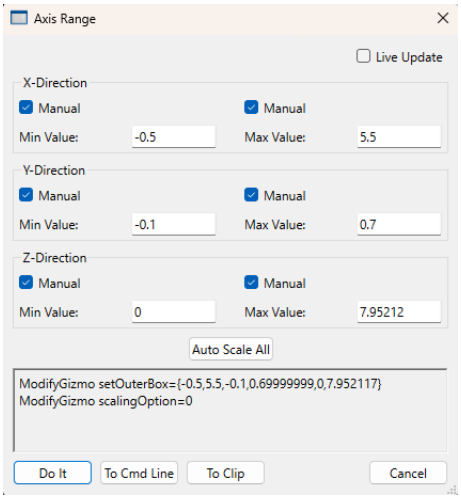
Gizmo の次元

デフォルトの Gizmo 表示ボリュームは、3 次元すべてで 4 単位幅の空間です。
実際の表示ボリュームは、表示ボリュームの中央を中心として、各次元で 2 単位です。
表示ボリュームの各次元は、原点を中心に -1 から +1 まで広がっています。
プロットが回転したときに角でクリッピングが発生しないように、表示ボリュームは画面のボリュームよりも小さくなっています。



球体や円柱などの描画オブジェクトはすべて、表示ボリューム ±1 単位でサイズが設定されます。
例えば、一辺が 2 単位のボックスを作成すると、表示ボリューム全体が完全に埋まります。
高さ 3 単位の円柱を作成すると、円柱の上部が表示ボリュームの境界外に伸びるため、クリッピングされます。

表示ボリュームに重ねて、それを正確に埋めるように、散布図やサーフェスプロットなどのウェーブベースのデータオブジェクトがプロットされる軸座標系があります。
Gizmo→Axis Range を選択することで、各次元の軸座標範囲を設定できます。
デフォルトでは軸は表示されていませんが、軸座標系は存在しています。



軸座標系はデフォルトで自動スケーリングされます。
その結果、ウェーブベースのオブジェクトを最初に表示すると、各軸の範囲全体に表示されます。
軸座標系が表示ボリューム全体を占めるため、表示されるウェーブベースのオブジェクトも表示ボリューム全体を占めます。

軸が自動スケーリングに設定されている状態で、２つ以上のウェーブベースのオブジェクトを同時に表示する場合、Gizmo は全データオブジェクトの組み合わせにおける各次元の最小値と最大値に基づいて、各軸の範囲を設定します。

自動スケーリングをオフにすると、設定した軸範囲によって、ウェーブベースのオブジェクトが表示ボリュームを埋める範囲が決まります。

描画オブジェクトとウェーブベースのオブジェクトを組み合わせた場合、描画オブジェクトの大きさと位置は表示ボリューム単位で ±1 の範囲内に留まる一方、ウェーブベースのオブジェクトは軸座標系に対して表示されます。

Gizmo のクリッピング

軸の範囲を設定する場合、データの全範囲を含まない値を使うことができます。
この場合、結果を正しく表示するために、Gizmo は表示ボリュームの関連する側面にクリッピング平面を作成します。
クリッピング平面は、作成されると、ウェーブベースのデータオブジェクトと描画オブジェクトの両方に影響を与えます。
データオブジェクトが軸の範囲を超えていない限り、クリッピング平面は作成されません。

独自のクリッピングを行いたい場合、この自動の Gizmo クリッピングが干渉する可能性があります。
自動クリッピングを無効にするには、例えば surface という名前のサーフェスオブジェクトに対して、以下を実行します：

```
ModifyGizmo modifyObject=surface0, objectType=surface, property={Clipped,0}
```

高度なモード（「高度な Gizmo のテクニック」のセクションを参照）で作業している場合は、カスタムのクリッピング平面を作成して、サーフェスプロットに隙間などの特殊効果を作成することができます。
クリッピング平面を使うには、どの次元でもデータの範囲よりも小さい軸範囲を使っていないことを確認してください。
現在のグラフィックスハードウェアは6から8個のクリッピング平面をサポートしており、軸範囲クリッピング平面は優先度があります。
デモエクスペリメントの Clipping Demo を参照してください。

Gizmo の投影

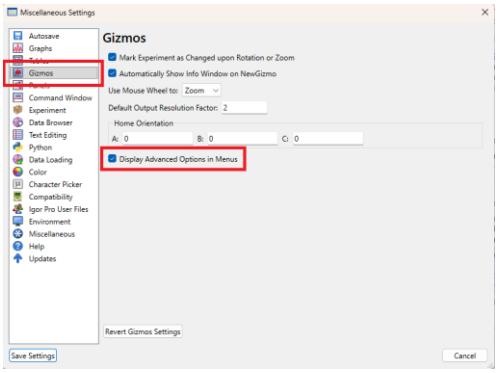
2D Gizmo ウィンドウにおける 3D オブジェクトの表示のコントロールには、複数の異なる投影法が利用可能です。

デフォルトでは、Gizmo は組み込みのデフォルト正投影を使います。
これは Display List には表示されません。
デフォルトの正射投影は、ほとんどのアプリケーションに最適です。

Gizmo は、Display List 内の全オブジェクトをスキャンし、それらの最大範囲を計算することで、デフォルトの正射投影で使うパラメーターを算出します。
デフォルトの正射投影は全方向に２単位となり、ウェーブベースのオブジェクトをクリッピングなしで完全に回転させることが可能になります。

Display List 上のオブジェクトの自動スキャンでは、オプションの平行移動、回転、またはスケーリングコマンドは考慮されません。
Display List でこれらのいずれかを使う場合は、投影空間の適切な定義を持つ別の正射投影コマンドも提供する必要があります。

投影コマンドを追加するには、Display List の下にある「+」アイコンをクリックします。
デフォルトでは直交投影のみが提供されており、これはほぼ全ての用途で十分です。
別の投影タイプを選択するには、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。
このダイアログは Misc メニューからアクセスできます。



Display List に対して任意の数の投影コマンドを実行できます。
複数の投影がある場合、Gizmo は最後のもののみを実行します。

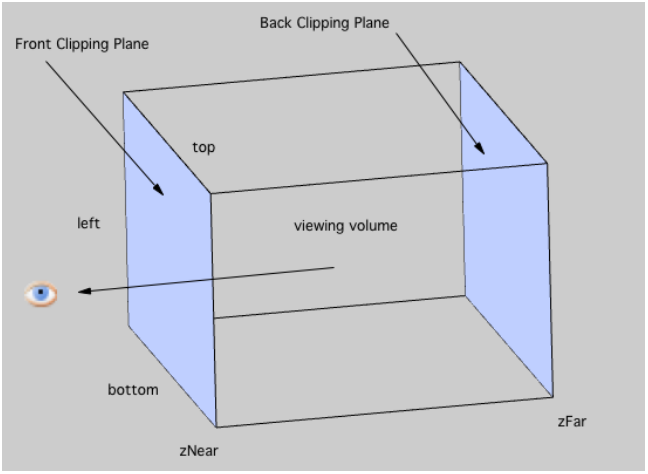
Ortho 以外の投影法を使う場合は、視角も指定する必要があります。
その結果、標準のマウス回転およびマウスホイールによるズームは適用されません。

正射投影法

正射投影は、3D オブジェクトの幾何学的向きと縮尺を維持します。
これは Gizmo 表示ウィンドウのデフォルトの投影法であり、ほとんどの用途で推奨されます。

次のセクションで説明する透視投影とは異なり、正射投影では物体の平行性が保たれ、物体の短縮は発生しません。
正射投影は、表示されるシーン内の物体の大きさが、視距離に比べて小さい配置に似ています。

別の考え方としては、像面が座標軸の１つに対して垂直であるということです。
視体積内に含まれる物体のみが表示されます。
この投影は、透視投影よりも高速です。



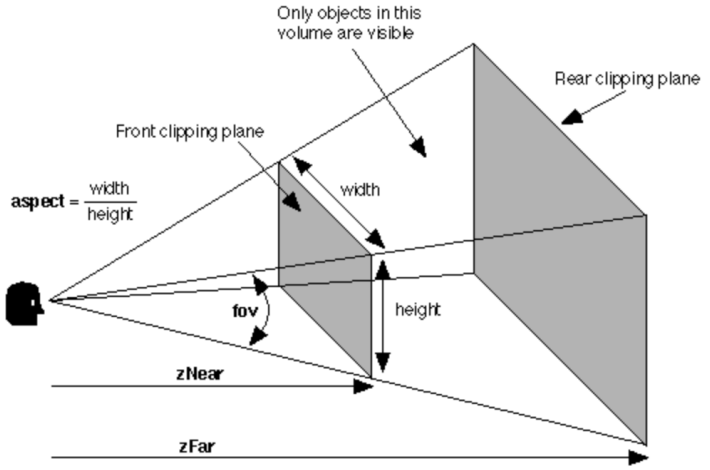
図に示すように、正射投影は６つのパラメーターに依存します：left、right、bottom、top、zNear、zFar。
これらのパラメーターは表示ボリュームの中心から測定され、±1 表示ボリューム単位で表されます。

ズーム（マウスホイール使用）およびパンツールは、デフォルトの正射投影を修正することで実装されています。
Display List に独自の投影を追加すると、ズームツールは無効になります。

透視投影

透視投影は、人間の目が 3D 物体を見る方法を模倣します。
これは現実的な投影法ですが、物体の正確な向きや形状を保持しません。
例えば平行線は分岐または収束し、物体の短縮（遠近法による短縮）が生じます。
表示領域は、底面と平行に頂部が切断された切頂ピラミッドの形状をしています。
表示ボリューム内のオブジェクトのみが表示されます。
これは表示ボリュームの対称的な透視図です。
非対称ボリュームは、次節で説明する台形（フラストラム／角錐台）によって支えられています。

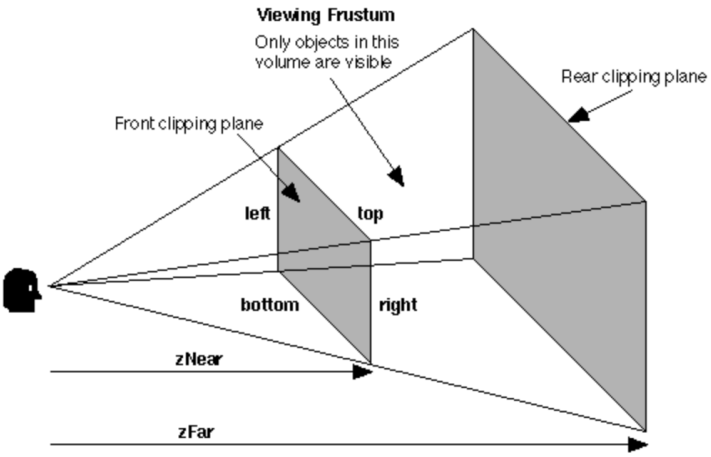
図に示すように、透視投影は 4 つのパラメーターに依存します：視野角（fov）、アスペクト比（aspect）、近点距離（zNear）、遠点距離（zFar）。



フラストラム（角錐台）投影

フラストラム投影は本質的に透視投影と同じですが、この場合、より柔軟な設定が可能です。
つまり、対称である必要もなければ、Z 軸に沿う必要もありません。

図に示すように、フラストラム投影は 6 つのパラメーターに依存します：left、right、bottom、top、zNear、zFar です。



ビューポート（表示領域）投影

ビューポートとは、投影されたシーンが描画されるウィンドウ内の長方形の 2D 領域です。
これを使って、Gizmo 表示ウィンドウ内のシーンを拡大縮小したり歪ませたりできます。

ビューポート投影は、4 つのパラメーターに依存します：left、bottom、width、height です。

LookAt 投影

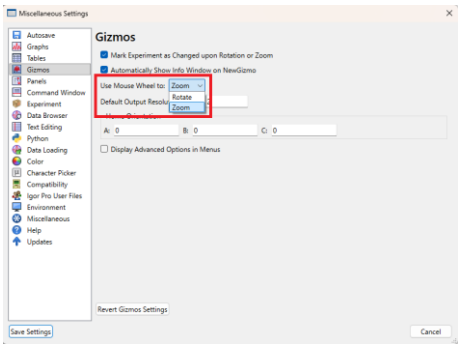
LookAt 投影は、中心点を負の Z 軸に、視点を中心に、上方向ベクトルを Y 軸にマッピングします。
座標系の原点と基本方向を変更したい場合に役立ちます。

Gizmo オブジェクトの回転

Gizmo 表示内でクリックし、マウスをドラッグするとプロットを回転できます。
デフォルトの回転は、Gizmo ウィンドウ中央に仮想トラックボールがあるかのように実装されています。
原点からの線に沿って見た場合、正の角度での回転は反時計回り方向となります。

回転を試してみる時は、X、Y、Z 方向を示す軸のガイドを表示すると便利です。
右クリックし、Show Axis Cue を選択します。
軸のガイドは、メインの Gizmo 軸の方向を示します。

Gizmo 表示ウィンドウがアクティブな場合、キーボードでプロットを回転できます。
x、y、z キーを押すと、対応する軸を中心に反時計回りに 1 度刻みで表示を回転します。
Shift キーを押しながら、x、y、z キーを押すと、時計回りに回転します。



上矢印キーと下矢印キーは、表示領域の中央を通る水平線を中心にプロットを回転させます。
左矢印キーと右矢印キーは、表示領域の中央を通る垂直線を中心にプロットを回転させます。

デフォルトでは、マウスホイールでプロットをズームします。
その他の設定でプロットの回転に変更できます。
Misc→Miscellaneous Settings を選択し、左側のリストで Gizmo をクリックすると、Gizmo のその他の設定が表示されます。
マウスホイールによるスクロールは、矢印キーを使った場合と同じ動作をします。

マウスをクリックして軽くフリックすると、3D シーンを連続回転させることができます。
この操作では、マウスを止める前にマウスボタンを離す必要があります。
連続回転を停止するには、プロットを一度クリックしてください。
表示が回転ではなくパンになっている場合、Gizmo ツールパレットの矢印ツールをクリックして回転を有効にしてください。

ツールパレットから連続回転を開始することもできます。
Gizmo→Show Tools を選択し、3 つの回転アイコンのいずれかをクリックすると、X 軸、Y 軸、Z 軸を中心とした回転が開始されます。
プロットを 1 回クリックするか、ツールパレットの停止アイコンをクリックすると回転が停止します。

ツールパレットのホームアイコンは、シーンをホーム方向へ回転させます。
デフォルトのホーム方向は X=0、Y=0、Z=0 です。
この場合、正の X 軸は右方向、Y 軸は上方向、Z 軸はウィンドウの平面から自分に向かって外側を指します。
ホーム方向は、右クリックして Set Home Orientation を選択することで設定できます。

連続回転中は、x、y、z キーを使って向きを微調整できます。

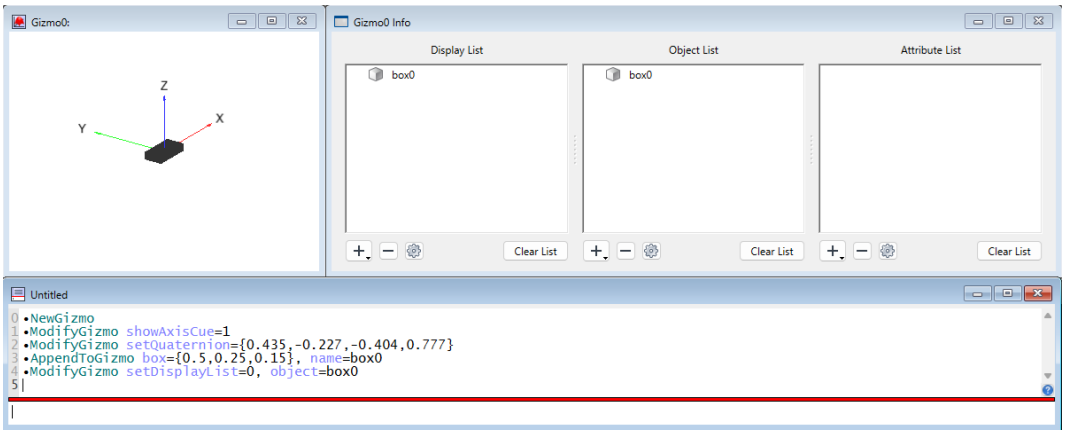
Gizmo オブジェクトの回転と移動

前のセクションでは、メインの Gizmo 軸を中心とした Gizmo プロット全体の回転について説明しました。
各 Gizmo オブジェクトは独自の軸セットを持ち、デフォルトではメイン軸に対応しています。
あまり頻繁には必要とされませんが、Gizmo オブジェクトの軸をメインの Gizmo 軸とは独立して回転させること

も可能です。
これを行うには、Display List に回転コマンドを追加します。

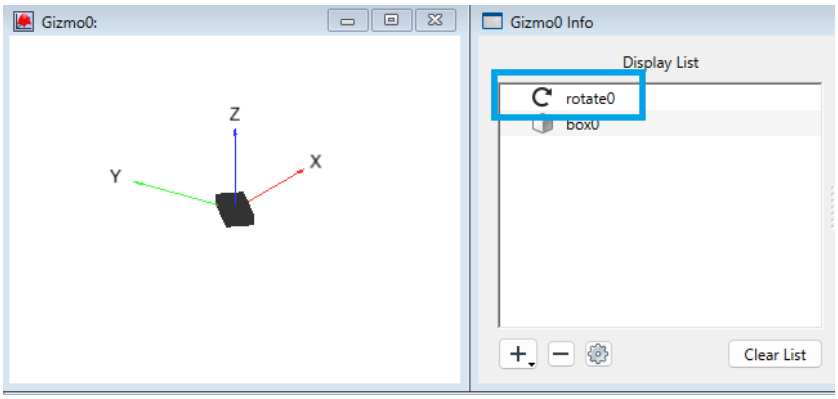
これを確認するため、軸キューと回転していないボックスを持つ Gizmo を作成します：

```
NewGizmo
ModifyGizmo showAxisCue=1
ModifyGizmo setQuaternion={0.435,-0.227,-0.404,0.777}
AppendToGizmo box={0.5,0.25,0.15}, name=box0
ModifyGizmo setDisplayList=0, object=box0
```



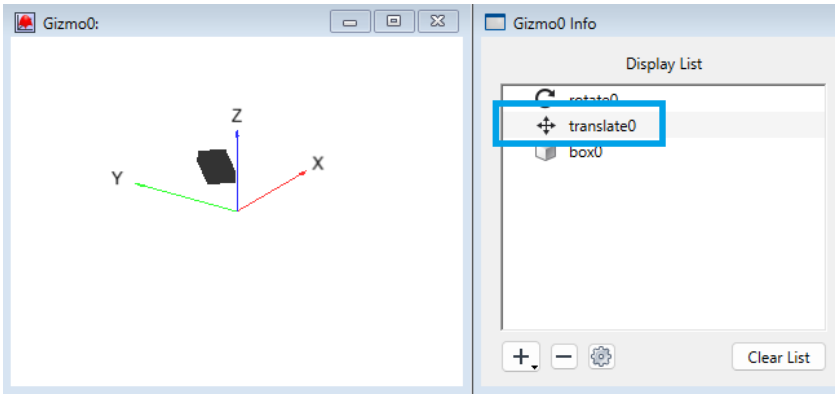
次に、ボックスオブジェクトの前に回転コマンドを Display List に追加します。
これによりボックスオブジェクトの軸が 45 度回転します：

```
ModifyGizmo insertDisplayList=0, opName=rotate0, operation=rotate, data={45,0,0,1}
```



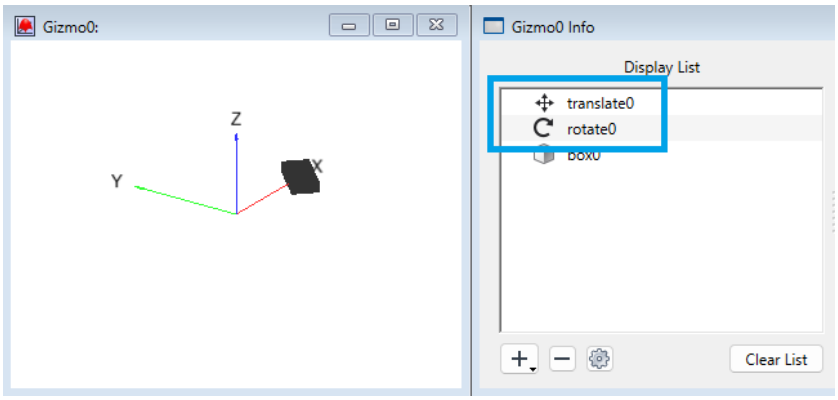
次に X 軸に沿って移動を挿入します：

```
ModifyGizmo insertDisplayList=1, opName=translate0, operation=translate, data={1,0,0}
```



オブジェクトの X 軸に沿った平行移動であり、主 X 軸に沿ったものではありません。
このため、回転後に平行移動を行う場合と、平行移動後に回転を行う場合では結果が異なります。
これを説明するため、コマンドの順序を入れ替えます：

```
ModifyGizmo setDisplayList=0,opName=translate0, operation=translate, data={1,0,0}
ModifyGizmo setDisplayList=1, opName=rotate0, operation=rotate, data={45,0,0,1}
```



この最後のケースでは、移動が行われた時点で、オブジェクトの軸は主軸と一致していました。
移動はオブジェクトの X 軸に沿って行われ、その X 軸は主 X 軸と同じ方向を指します。

平行移動と回転のコマンドは、Display List でその下にあるすべてのオブジェクトに適用されます。
これらは累積的です。
つまり、特定のオブジェクトに対する平行移動または回転コマンドは、そのコマンドが適用されたときのオブジェクトの現在の位置または回転から開始されます。

Gizmo 内の回転のプログラミング

Gizmo プロットの向きは内部で四元数として保存されます。
四元数は複素数に類似していますが、３次元へ拡張されたものです。

プロットを手動で回転させると、内部四元数が変更されます。
GetGizmo curQuaternion を使ってクエリできます。

方向をプログラムで設定する ModifyGizmo キーワードがいくつかあります。

setQuaternion、setRotationMatrix、euler キーワード
: それぞれ絶対的な方向を設定し、四元数パラメーター、変換行列、またはオイラー角のセットを受け取ります。

appendRotation キーワード
: 四元数で指定された回転を現在の方向に適用します。

goHome キーワード
: ホーム位置に移動します。

idleEventQuaternion、idleEventRotation キーワード
: 定期的に向きを変更します。

matchRotation キーワード
: 別の Gizmo ウィンドウの向きに合わせるように設定します。

syncRotation キーワード
: １つの Gizmo ウィンドウの回転を別のウィンドウの回転に同期させます。

stopRotation キーワード
回転を停止します。

向きをどのように設定しても、内部的には四元数として保存されます。

Gizmo の表示を回転させて、X 軸が右を指し、Y 軸が自分から離れ、Z 軸が上を指すようにしたい場合、X 軸周りの 90 度回転を行うための四元数が必要です。
これは次のコマンドで実現できます。

```
ModifyGizmo setQuaternion={sin(pi/4),0,0,cos(pi/4)}
```

プロットを回転軸と角度で指定された向きに回転させたい場合、まずそれらの入力を四元数に変換する必要があります。
回転軸が Ax, Ay, Az で与えられ、角度 θ がラジアン単位の場合、回転四元数は次の４つの要素で構成されます。

```
Qx = Ax*sin(theta/2)/N
Qy = Ay*sin(theta/2)/N
Qz = Az*sin(theta/2)/N
Qw = cos(theta/2)
```

ここでは、回転ベクトルを N=sqrt(Ax^2+Ay^2+Az^2) を使って正規化しました。

連続する２つの回転、すなわち四元数 q1 で指定される回転に続いて四元数 q2 で指定される回転を表す回転四元数を計算するには、四元数乗法を用いて積四元数 qr=q2*q1 を計算する必要があります。
この乗法は可換ではありません。
これは以下の関数を用いて計算できます。

```
// q1 と q2 は、{x, y, z, w} 四元数に対応する４要素のウェーブです。
// この関数は、四元数積 q2*q1 を表す新しい四元数を qr に計算します。

Function MultiplyQuaternions(q2,q1,qr)
	Wave q2,q1,qr

	Variable w1=q1[3]
	Variable w2=q2[3]
	qr[3]=w1*w2-(q1[0]*q2[0]+q1[1]*q2[1]+q1[2]*q2[2])
	Make/N=4/FREE vcross=0
	vcross[0]=(q2[1]*q1[2])-(q2[2]*q1[1])
	vcross[1]=(q2[2]*q1[0])-(q2[0]*q1[2])
	vcross[2]=(q2[0]*q1[1])-(q2[1]*q1[0])
	MatrixOP/FREE aa=w1*q2+w2*q1+vcross
	qr[0]=aa[0]
	qr[1]=aa[1]
	qr[2]=aa[2]
	Variable NN=norm(qr)
	qr/=NN
End
```

前の例で生成された回転（X 軸が右方向、Y 軸が自分から離れる方向、Z 軸が上方向）を Gizmo の向きに設定し、その後 Z 軸を中心に 90 度回転させ、X 軸が自分の方向、Y 軸が右方向、Z 軸が上方向を指す向きを生成したいとします。
これを行うには次を実行します。

```
Make/O/N=4 q1={sin(pi/4),0,0,cos(pi/4)}    // Z up, X right, Y away
Make/O/N=4 q2={0,0,sin(pi/4),cos(pi/4)}    // 90 degree rotation about Z
Make/O/N=4 qr                              // Resultant orientation
```

```
MultiplyQuaternions(q2,q1,qr)              // Compute resultant orientation
Print qr
  qr[0]= {0.5,0.5,0.5,0.5}
ModifyGizmo setQuaternion={0.5,0.5,0.5,0.5}
```

これを行う別の方法は、ModifyGizmo の appendRotation コマンドを使うことです。
このコマンドは四元数の乗算を自動的に行ってくれます。

```
ModifyGizmo setQuaternion={sin(pi/4),0,0,cos(pi/4)}      // Z up, X right, Y away
ModifyGizmo appendRotation={0,0,sin(pi/4),cos(pi/4)}
```

最後のコマンドは、現在の向きから Z 軸を中心に 90 度回転します。

回転のロック

状況によっては、Gizmo プロットの回転を禁止したい場合があります。
これを行うには、次のコマンドを実行します。

```
ModifyGizmo lockMouseRotation = 1
```

これにより回転がロックされますが、ロックされていることを示す目に見える手がかりが提供されないことに注意してください。

より複雑な手法としては、Display List に固定の表示変換を設定し、描画対象のオブジェクトをすべて追加した後、MainTransformation コマンドで終了する方法があります。
メインの変換より上位にある Display List 内の全オブジェクトは、固定の回転とスケーリングで描画されます。

Gizmo の色、素材、照明

Gizmo オブジェクトのレンダリング色は、その内部色、色属性、カラー素材コマンドで指定された素材、および照明によって決まります。
以下のセクションではこれらのトピックについて説明します。

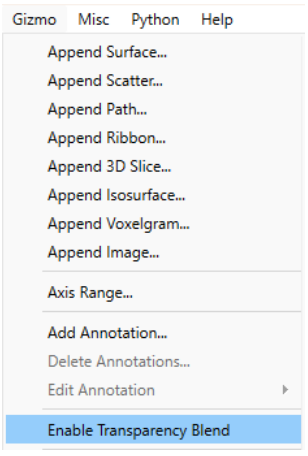
Gizmo の色指定

物体や光の色彩は、４つの浮動小数点数（RGBA）を用いて指定されます。

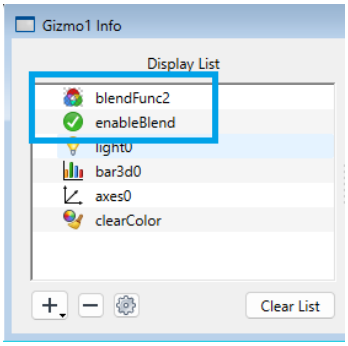
最初の３つは、最終的な色を足し算的に生成する三原色の赤、緑、青です。
各成分の強度値は [0.0～1.0] の範囲の数値です。

４つ目の要素はアルファで、オブジェクトの不透明度を決定します。
値は、完全に不透明な色である 1.0 から、完全に透明または無色のオブジェクトである 0.0 までの範囲です。

グラフィックスリソースを節約するため、Gizmo ウィンドウのアルファブレンディングはデフォルトで無効になっています。
透明度を持つオブジェクトを作成するには、色のアルファ成分を設定するだけでなく、メニュー Gizmo→Enable Transparency Blend を選択して透明度ブレンディングを有効にする必要があります。

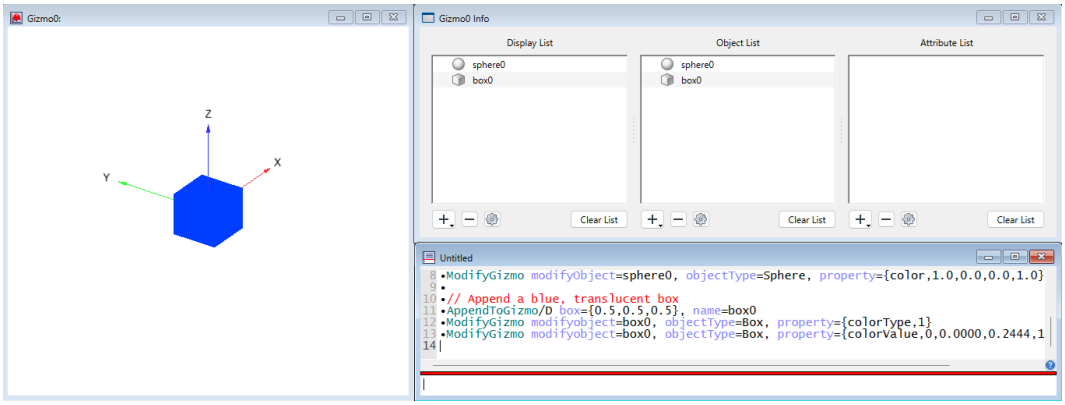


デフォルトでは、blendFunc0 と enableBlend 項目はディスプレイリストの最上部に挿入されるため、これらに続く全ての項目の描画に影響を与えます。
この２つの項目を、透明度ブレンドを必要とする項目の直上に移動することで、描画速度を向上できる可能性があります。
その場合、透明度を使う最後の項目の直後に、ブレンドを無効化する disable コマンドを追加する必要があります。
詳細は「透明度と半透明」のセクションを参照してください。



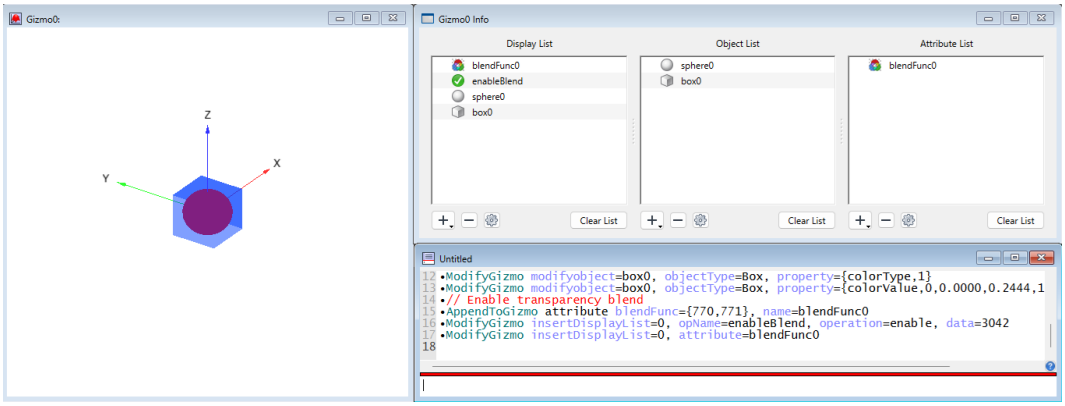
以下は透明度を伴う色の指定例です。

```
// 軸キューを持つ新規 Gizmo を作成し、回転を設定する
NewGizmo
ModifyGizmo showAxisCue=1
ModifyGizmo setQuaternion={0.435,-0.227,-0.404,0.777}

// 赤い不透明な球体を追加する
AppendToGizmo/D sphere={0.25,25,25}, name=sphere0
ModifyGizmo modifyObject=sphere0, objectType=Sphere, property={colorType,1}
ModifyGizmo modifyObject=sphere0, objectType=Sphere, property={color,1.0,0.0,0.0,1.0}

// 青色の半透明のボックスを追加する
AppendToGizmo/D box={0.5,0.5,0.5}, name=box0
ModifyGizmo modifyobject=box0, objectType=Box, property={colorType,1}
ModifyGizmo modifyobject=box0, objectType=Box,
        property={colorValue,0,0.0000,0.2444,1.0000,0.5000}
```



半透明の青いボックスが赤い球体を囲んでいますが、アルファ合成がデフォルトで無効化されているため赤い球体は見えません。
ここで有効にします。

```
// Enable transparency blend
AppendToGizmo attribute blendFunc={770,771}, name=blendFunc0
ModifyGizmo insertDisplayList=0, opName=enableBlend, operation=enable, data=3042
ModifyGizmo insertDisplayList=0, attribute=blendFunc0
```



最後のコマンド群は、メニュー Gizmo→Enable Transparency Blend を選択した時に実行されるものです。

Igor の色を Gizmo の色に変換

歴史的な理由により、Igor は色成分を 0 から 65535 までの整数値で表現します。
OpenGL、そして結果の Gizmo は、カラーコンポーネントを 0.0 から 1.0 までの浮動小数点値で表現します。
ColorTab2Wave コマンドから受け取るような Igor カラーコンポーネント値を、Gizmo コマンドで使うための Gizmo カラーコンポーネント値に変換するには、Igor カラーコンポーネント値を 65535 で割る必要があります。

以下に、そのような変換の一例を示します。

```
ColorTab2Wave Rainbow                              // M_colors を作成
MatrixOP/O gizmoRainbowColors = M_colors/65535 // SP に変換しスケーリング
Redimension/N=(-1,4) gizmoRainbowColors          // アルファのための列を追加
gizmoRainbowColors[][3] = 1                        // アルファを 1（不透明）に設定
```

Gizmo の色、素材、照明

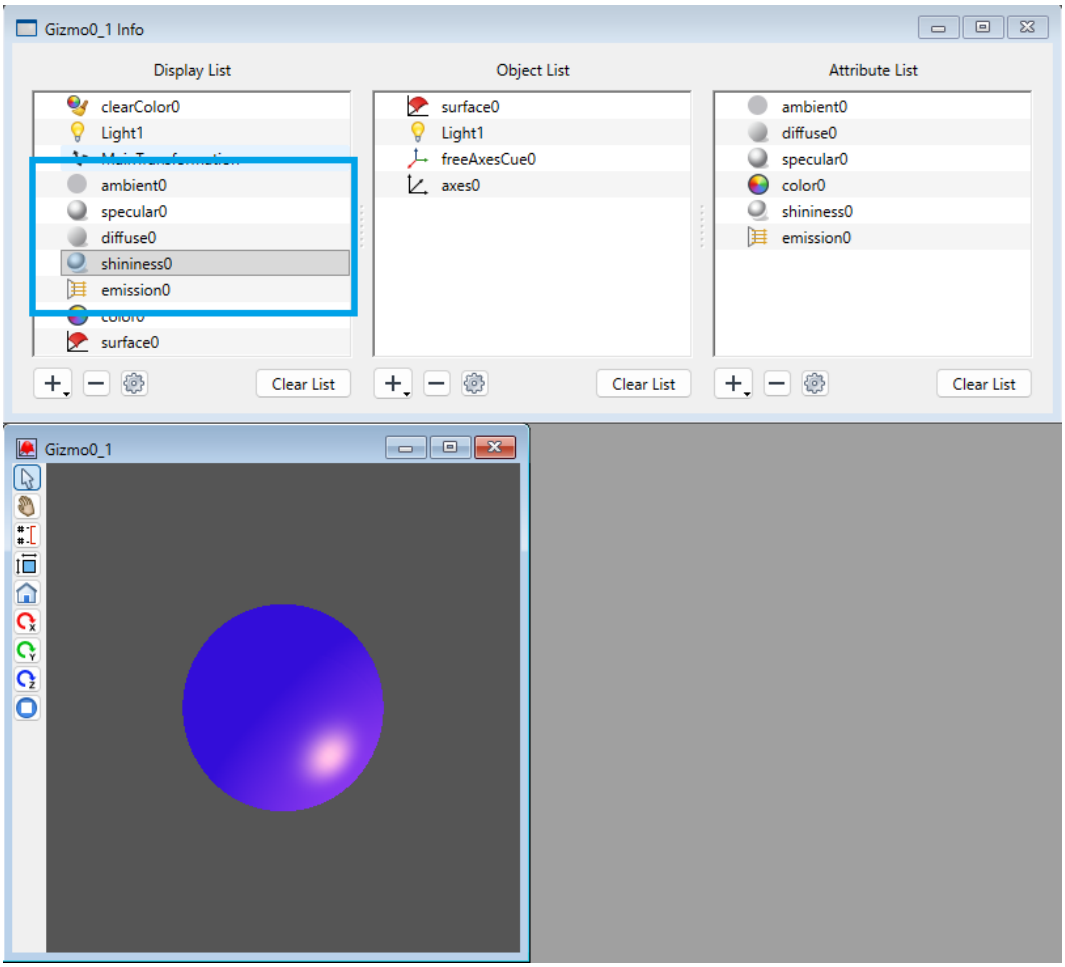
物体の知覚される色は、色、素材、照明の組み合わせによって決まります。

色素材は、オブジェクトのどの面が OpenGL によって着色されるか、およびオブジェクトが光を放射または反応する方法を指定します。
「face」プロパティは、GL_FRONT、GL_BACK、または GL_FRONT_AND_BACK に設定できます。
「mode」プロパティは、GL_EMISSION、GL_AMBIENT、GL_DIFFUSE、GL_SPECULAR、または GL_AMBIENT_AND_DIFFUSE に設定できます。

Gizmo オブジェクトを作成する時、色を指定するか、未指定のままにするかを選択できます。
色を指定すると、Gizmo はオブジェクト用のデフォルト色素材を作成します。
デフォルト色素材には、GL_FRONT_AND_BACK、GL_AMBIENT_AND_DIFFUSE の設定が適用されます。
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。

特定のオブジェクトに適用されている素材の色に関わらず、Display List 内でその素材の属性の上に ambient（環境光）、diffuse（拡散光）、specular（鏡面反射）、shininess（光沢度）または emission（発光）属性を追加することで変更できます。

光沢のあるオブジェクト（例：球体）を作成するには、球体オブジェクトから始め、光沢度と鏡面反射属性を追加し、Display List で拡散成分と鏡面反射成分が一致するライトオブジェクトの後に球体を追加します。
この場合、情報ウィンドウと表示ウィンドウは次のようになります。



例として、デモエクスペリメント Material Attributes（File→Example Experiments→Visualization）を参照してください。

カラーウェーブ

ウェーブベースのオブジェクトは、固定色または組み込みの Igor カラーテーブルを使って描画できます。
また、独自のカラーウェーブを使って、Gizmo 表示ウィンドウ内のオブジェクトの色を指定することもできます。
カラーウェーブのフォーマットは、対応するデータウェーブのフォーマットと似ていますが、各データノード（頂点）には、赤、緑、青、アルファのカラーコンポーネントが関連付けられている点が異なります。

カラーウェーブが有用な状況の１つは、3D サーフェス上のポイントに対応するスカラー値（例：温度測定値）のセットを表示したい場合です。
この場合、サーフェスの形状を表すウェーブと、スカラー測定値を含む別のウェーブが１つずつ存在します。
カラーウェーブを適用することで、表面上に分布するスカラーデータを完全に自由に表現できます。
ほとんどの場合、ModifyGizmo の makeSurfaceColorWave と makeTripletColorWave キーワードを使って、組み込みテーブルのいずれかに基づいてデータ用のカラーウェーブを作成し、その後カラーウェーブ内で適切なアルファ値を指定することで、適切なカラーウェーブを作成できます。

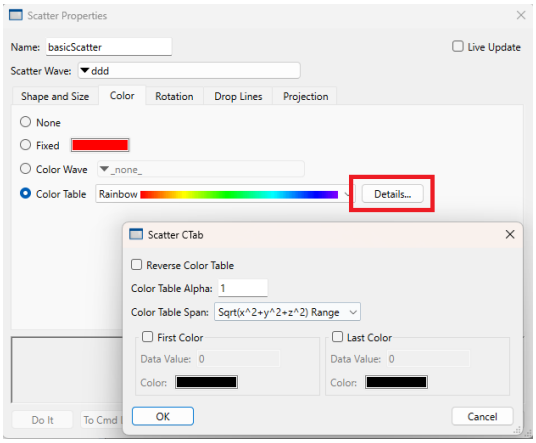
必要なカラーウェーブのフォーマットは、色を適用する対象を記述するデータウェーブのフォーマットによって異なります。
次の表は、様々なデータウェーブフォーマットと対応するカラーウェーブフォーマットの一部を示しています。
次元は括弧内に示しています。

データウェーブ	カラーウェーブ	用途
トリプレット（Mx3）	（Mx4）連続する列で RGBA	パス、リボン、散布図
Z 値の行列（MxN）	（MxNx4）連続するレイヤーで RGBA	サーフェス、3D 棒グラフ
連続した 4 辺形（Mx4x3）	（Mx4x4）連続するレイヤーで RGBA	パラメトリックサーフェス
分離された 4 辺形（Mx12）	（Mx4x4）連続するレイヤーで RGBA	パラメトリックサーフェス
三角形（Mx3）	（Mx4）連続する列で RGBA	パラメトリックサーフェス
パラメトリック（MxNx3）	（MxNx4）連続するレイヤーで RGBA	パラメトリックサーフェス

Gizmo のカラーテーブル

Igor の組み込みカラーテーブルを使って、サーフェス、散布、パス、リボンオブジェクトの色を指定できます。
等値面、ボクセルグラム、3D 棒グラフ、および画像オブジェクトはカラーテーブルの使用をサポートしていません。

サーフェス、散布、パス、リボンオブジェクトについては、各オブジェクトタイプのプロパティダイアログでカラーテーブルを選択できます。
利用可能なカラーテーブルの一覧についてはヘルプ CTabList を参照し、詳細についてはヘルプ Color Table Details を参照してください。



カラーテーブルを選択すると、関連するオプションも設定できます。
プロパティダイアログでは、Details ボタンを使ってこれらのオプションを設定します。
このボタンをクリックすると、右のようなサブダイアログが表示されます。

Color Table Alpha 設定は、カラーテーブル内のすべての色に適用されます。
可変アルファを適用したい場合は、各データポイントのアルファ値を指定するカラーウェーブを使ってオブジェクトの色を設定してください。
カラーウェーブは等値面オブジェクトではサポートされていません。
アルファ効果には、ブレンドが有効化されていることと、Display List にブレンド関数が存在することが必要です。
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。

Color Table Span 設定は、カラーテーブルから色を選択するために使われる数値量を決定します。
散布図の場合、最も一般的な選択はグローバル Z 範囲です。
デフォルトでは、これはプロットに表示される最低の Z 値が最初の色に、最高の Z 値が最後の色にマッピングされることを意味します。
特定の散布要素の色は、その要素の Z 値に基づいて選択されます。

このマッピングは、First Color と Last Color 設定を使って調整できます。
First Color を有効にして対応するデータ値を入力すると、そのデータ値はカラーテーブルの最初の色にマッピングされ、入力した値よりも小さいデータ値を持つ散布要素は、下のカラーポップアップメニューから選択した色で表示されます。
Last Color を有効化し対応するデータ値を入力すると、そのデータ値はカラーテーブルの最終色にマッピングされ、入力値より大きいデータ値を持つ散布要素は、下のカラーポップアップメニューから選択された色で表示されます。

反射

Gizmo は、光との表面オブジェクト相互作用において、ambient（環境光）、diffuse（拡散光）、specular（鏡面反射）、shininess（光沢度）の４種類をサポートしています。

光には環境光、拡散光、鏡面反射の成分があります。

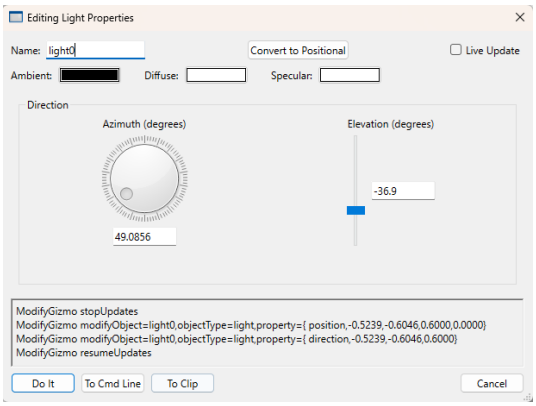
素材には、環境光、拡散光、鏡面反射、光沢度の属性があります。

環境光

環境反射は物体の全体的な色を決定します。
環境反射は物体の影で最も顕著に現れます。
環境反射率の総量は、グローバル環境光と個々の光源からの環境光によって決定されます。

拡散光

拡散表面反射は、光を全方向に均等に散乱させます。
これは物体の色を決定する最も重要な要素です。
入射する拡散光の色と、法線方向に対する入射光の角度の影響を受けます。
入射光が表面に垂直に当たる場所で最も強くなります。
視点位置の影響をは受けません。



鏡面反射

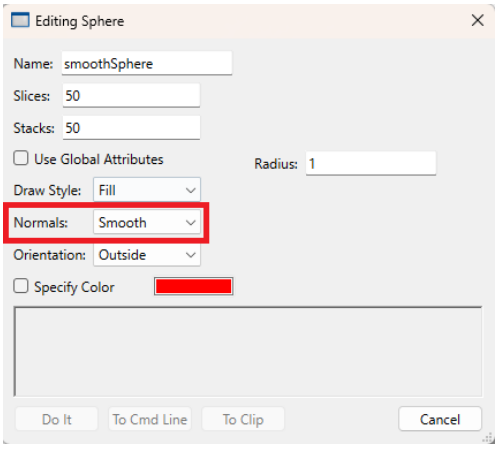
鏡面反射は物体上のハイライトの外観を決定します。
鏡面反射の量は視点の位置に依存し、直接反射角に沿って最も明るくなります。

光沢度

光沢度は鏡面反射ハイライトのサイズと明るさをコントロールします。
物体が光沢度が高いほど、ハイライトは小さく明るくなります（より焦点が絞られます）。

法線、照明、シェーディング

照明効果のあるシーンを表示する時は、Display List 内の全オブジェクトで法線計算を有効にしてください。
各オブジェクトのプロパティダイアログで設定可能です。
オブジェクトの種類に応じて、Calculate Normals チェックボックスをオンにするか、Normals ポップアップメニューから選択してください。
これらの設定は、グラフィックス処理リソースを節約するために、デフォルトではオフになっています。

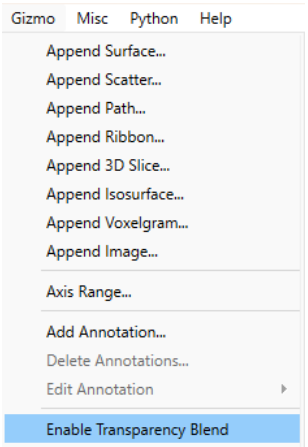


法線が必要な理由は、各ピクセルのシェーディングが、表面の法線と光源の方向の間の角度に依存するためです。
二次曲面オブジェクト（球体、円柱、円盤）の特殊なケースでは、フラット、スムース、法線なし、から選択できる内部設定があります。
法線を計算する場合、すべてのオブジェクトの描画速度は大幅に低下します。

透明度と半透明

OpenGL で透明度を適切に実装するには、オブジェクトをシーンの奥から手前に向かって描画する必要があります。
つまり、視点から最も遠いオブジェクトから描き始め、最も近いオブジェクトで終わらせます。
固定された視覚変換に対しては、表示されるオブジェクトが原始的で交差しない要素で構成されている限り、それらをソートすることが可能です。
二次曲面などの複合オブジェクトを描画する場合、構成セグメントの順序をコントロールすることはできません。
ほとんどのウェーブベースのオブジェクトは三角形配列に変換され、三角形が交差しない限り距離ソートが可能です。

距離ソートは計算コストが高いため、ほとんどのアプリケーションでは様々な手法を用いてこれを回避しています。
「貧者の解決策」はアルファブレンディングを使うことです。
この種の半透明効果は、限定された視野角の範囲で望ましい効果を提供でき、
Display List 上のオブジェクトの順序変更が必要になる場合があります。



アルファブレンディングを使うには、Display List 上の 2 つの異なるオブジェクトに色を割り当てます。
半透明オブジェクトは、その不透明度に対応するアルファ値を持つ必要があります。
不透明オブジェクトはアルファ=1、透明オブジェクトはアルファ=0 です。
半透明化に必要な最終手順は、ブレンド関数属性の追加と有効化コマンドです。
Gizmo→Enable Transparency Blend メニューを選択し、ブレンド関数と有効化コマンドを作成して Display List に追加します。

等値面は、その構成上、交わらない三角形で構成されるため、特殊なケースです。
ほとんどのアプリケーションでは、三角形を、その三角形の重心からの視点までの距離の順にソートすれば十分です。
saveToWave キーワードと ModifyGizmo を使って、等値面オブジェクトに対応する三角形を取得し、サンプルの視点を設定することができます。
標準的な正射投影法では、視点を無限遠とみなします。
この種のソート手法の例は、デモエクスペリメント Depth Sorting で見ることができます。

Gizmo の光源

Gizmo は指向性光源と位置光源の両方をサポートしています。
表示に追加する光の種類と色は、表示ウィンドウ内のオブジェクトの外観に影響を与えます。

Object List のポップアップメニューから「Light」を選択することで、他のオブジェクトと同様に光源を作成します。
以降で説明する Light Properties ダイアログを使うと、光源の種類や各種光源パラメーターを指定できます。
光源を効果的に機能させるには、DisplayList 内の照らしたいオブジェクトの上に追加する必要があります。

照明効果は、その Ambient（環境光）、Diffuse（拡散光）、Specular（鏡面反射光）の各要素によって定義されます。
光強度の分布は、光源の位置、方向、円錐角、および減衰によって記述されます。
物体の最終的な外観は、光の特性と物体素材の特性の組み合わせによって決まります。

照明効果はピクセル単位で計算されます。
したがって、滑らかなシェーディングを実現するには、十分な数の頂点を使ってオブジェクトを記述する必要があります。
１つの四角形（4 頂点）などの単純なオブジェクトの場合、四角形全体で照明の変化はほとんど見られないでしょ

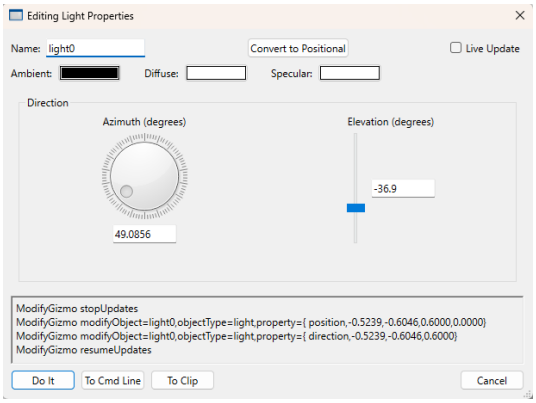
う。
シェーディングは、各頂点における表面法線と光源の方向との内積を使って計算されます。
光源からの光を遮る物体や光の多重反射は考慮されていません。

Display List に光源を追加しない場合、Gizmo はデフォルトの、色調に影響を与えない環境光を使います。
Display List に光源を追加すると、Gizmo はデフォルトの照明を削除します。

Gizmo の指向性光源

指向性光源は、シーンから非常に遠くに位置する光源と考えることができます。
これにより、その光線は表示領域内で実質的に平行になります。
太陽は指向性光源の良い例です。
新規に作成する光源オブジェクトはデフォルトで指向性となります。

Light Properties ダイアログには、光源の位置、色プロパティ、および分布を指定するために必要なコントロールが含まれています。
Display List に既に存在する光源オブジェクトを編集する時は、Live Update チェックボックスをクリックすると、変更が Gizmo 表示にどのように影響するかを確認できます。



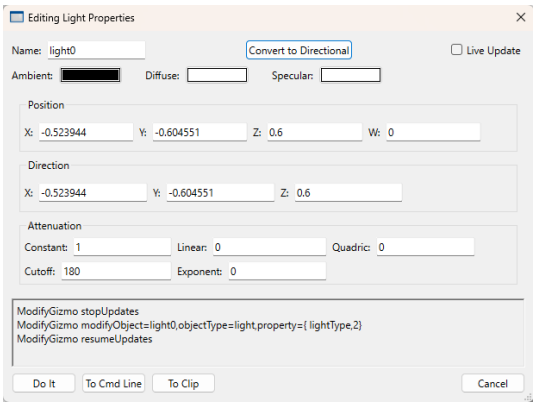
光源の位置は、方位角と仰角という 2 つの角度で指定されます。
仰角は、特に天文学では「高度」とも呼ばれます。
これらの角度の意味については、
http://en.wikipedia.org/wiki/Azimuth で説明されています。
仰角が +90 度または -90 度の場合、方位角は未定義となります。

環境光、拡散光、鏡面反射光については http://en.wikipedia.org/wiki/Phong_reflection_model で説明されています。
環境光は、物体の向きや光源の位置に関係なく、全ての物体の全ての部分を均等に照らします。
拡散光は、光が粗い表面に当たった場合のように、表面から全方向に反射されます。
鏡面反射光は、光が光沢のある表面に当たった場合のように、特定の方向に反射されます。
物体上の特定のポイントにおける拡散光と鏡面反射光による照明は、そのポイントの表面法線に対する光線の角度によって決まります。

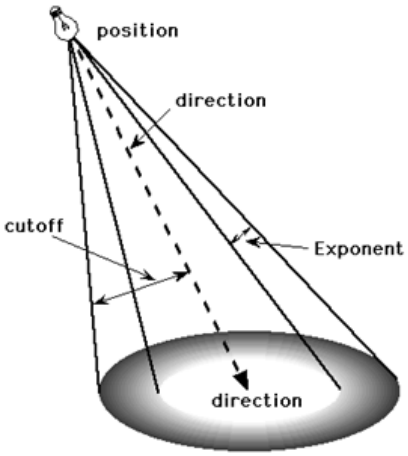
Gizmo の位置光源

位置光源とは、照明対象から有限の距離にある光源を指します。
デスクランプは位置光源の典型例です。
デスク上の任意の位置に設置可能で、照明スポットの中心からの距離に応じて光量が減衰するため、不均一な照明を生成します。

光源オブジェクトを作成すると、指向性光源として初期化されます。
ダイアログ内上部の Convert to Positional ボタンをクリックすると、対応する位置光源の設定が表示されます。
その後、必要に応じて調整できます。



位置光源のパラメーターを図に示します。



Positional Light Geometry

指向性光源ダイアログの上位 3 つのコントロールは、環境光、拡散光、鏡面反射光の各コンポーネントの RGBA 値を指定します。
Display List 内の光源のうち、少なくとも 1 つには環境光コンポーネントを設定する必要があります。
この要件は、主に拡散光または鏡面反射光の効果を作成しようとしている場合でも適用されます。

Gizmo オブジェクトを作成する時、色を指定するか、未指定のままにするかを選択できます。
色を指定すると、Gizmo はそのオブジェクト用にデフォルトの色素材を作成します。
デフォルト色素材には、GL_FRONT_AND_BACK と GL_AMBIENT_AND_DIFFUSE の設定が適用されます。
鏡面反射効果を表現したい場合は、鏡面反射属性と光沢度属性を追加する必要があります（詳細は「色、素材、照明」のセクションを参照）。

ダイアログの Position と Direction コントロールは、光源の位置と方向を表します。
位置は、最後の要素（w）が X、Y、Z 成分を正規化するために使われる、同次座標で表現されます。
したがって、w=1 を設定すると、X、Y、Z 成分は空間における光の絶対的な位置を指定します。
w=0 に設定すると、光源は無限遠に位置し、実質的に指向性光源を作成したことになります。

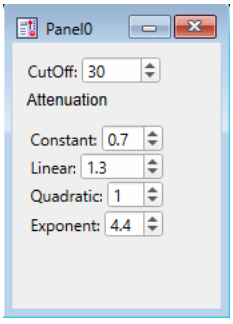
w=1 に設定し、ダイアログの次のグループにあるコントロールを使って方向を設定することを推奨します。
この方向は、光源の位置から光点の中心を照らしたい点までを指すベクトルの 3 成分として記述されます。
ここでよくある誤りは、方向ベクトルではなく光源の位置を入力してしまうことです。

光源の指定位置は、Display List 内の他のオブジェクトと同様の変換が適用されます。
特に、表示オブジェクトを回転させると、ライトも同様に回転します。
回転する表示の異なる部分を照らすために光源を固定したい場合は、最後の光源オブジェクトの直後、かつすべての回転オブジェクトの前に、メインの変換コマンドを Display List に追加する必要があります。
これにより、メイン変換より上にリストされているすべてのオブジェクトは固定され、回転はその後続の表示項目にのみ適用されます。

メイン変換コマンドを使うと、固定する要素と回転可能な要素を区別できます。
メイン変換コマンドは、Display List 内で座標変換が適用される起点となる点を設定します。
座標変換は回転に加え、平行移動と拡大縮小にも影響します。
Display List 内でメイン変換より上位の要素には変換が適用されないため、明示的な平行移動・回転・拡大縮小コマンドを挿入しない限り、それらの要素はデフォルトの状態で描画されます。

デフォルトでは、スポットのカットオフ角（光錐の半角）は 180 度であり、これは光が全方向に均一な照明を提供する意味です。
位置と方向を指定する手間をかけるなら、カットオフ角をより現実的な値（90 度未満）に減らすことが有用です。

定数、線形、および二次減衰が組み合わさって、位置光源（つまり、w が 0 以外の場合）を減衰させます。
0 から 128 の範囲の指数値は、スポットの中心からの強度減衰を決定します。
これは、中心強度を、光の方向と当該頂点との間の角度のコサインに指数値を乗じた値で乗算することで行われます。
これにより、光を非常に鏡面反射的にすることができます。



デモエクスペリメント Positional Light Demo（Example Experiments→Visualization→Advanced）には、各種位置光パラメーターの相互作用と、それらがオブジェクトの照明に与える影響を探索できるコントロールパネルが含まれています。

Gizmo の描画オブジェクト

Gizmo は、いくつかの描画プリミティブへのアクセスを提供します。
これらには、Line（線）オブジェクト、Triangle（三角形）オブジェクト、Quad（四角形）オブジェクト、Box（ボックス）オブジェクト、Sphere（球体）オブジェクト、Cylinder（円柱）オブジェクト、Disk（円盤）オブジェクト、Tetrahedron（四面体）オブジェクト、Pie Wedge（円弧）オブジェクトが含まれます。
さまざまなオブジェクトに関連するダイアログを使って、オブジェクトの様々なプロパティを設定できます。

「Gizmo の次元」のセクションで説明したように、描画オブジェクトのサイズは表示ボリューム単位で表現されます。
表示ボリュームは原点から各次元に ±1 まで広がります。
原点を中心とするサイズ 1 のボックスは、各次元にわたり原点から表示ボリュームの端まで半分広がります。

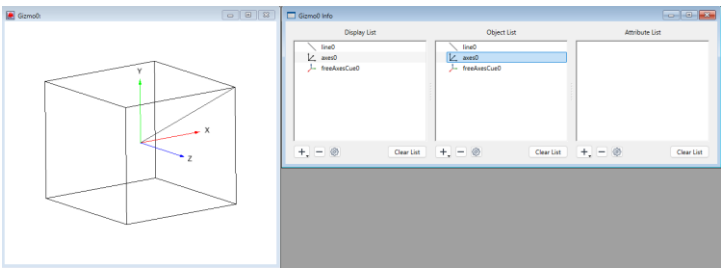
描画オブジェクトとは異なり、散布図や表面プロットなどのウェーブベースのオブジェクトは、別の座標系に対して表示されます。
描画オブジェクトとウェーブベースのオブジェクトを組み合わせる場合、描画オブジェクトの位置とサイズは常に +/-1 表示ボリュームで表現されることに留意してください。
2D ウェーブをサーフェスとして描画し、その周囲にボックスを描画したい場合は、長さ、幅、高さが 2 単位のボックスを追加するだけです。

Line オブジェクト

Line（線）オブジェクトは、 2 つの端点の座標を結ぶ直線です。
線の始点、終点、または中点に矢印を追加できます。コマンドによって作成された（1,1,1）から（0,0,0）への線を示します。

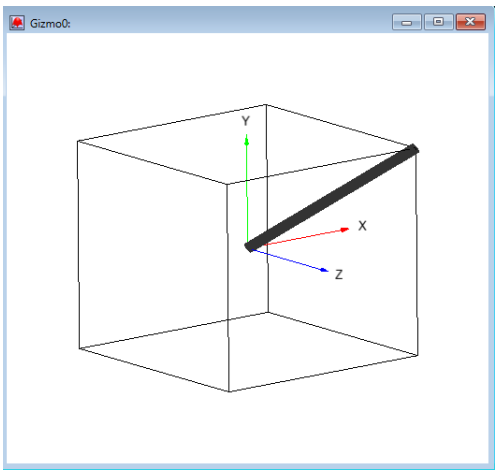
```
NewGizmo
AppendToGizmo/D line={1,1,1,0,0,0}
```

Object List で Axes と Free Axes Cue を追加して、Display List に追加すると図のようになります。



次の方法で線を 3D 円柱に変換できます。

```
ModifyGizmo modifyObject=line0,objectType=line,
        property={arrowMode,16}
ModifyGizmo modifyObject=line0,objectType=line,
        property={cylinderStartRadius,0.05}
ModifyGizmo modifyObject=line0,objectType=line,
        property={cylinderEndRadius,0.05}
```



Triangle オブジェクト

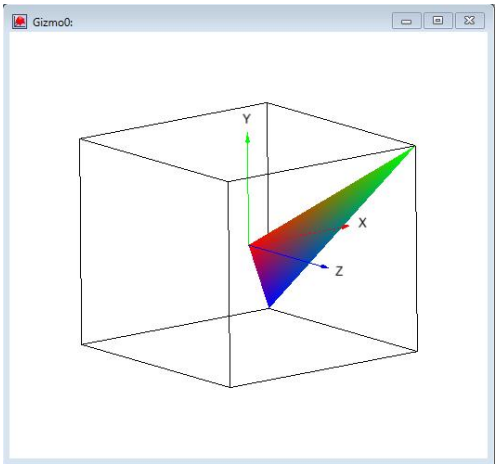
Triangle（三角形）は、 3 つの頂点を結ぶ単純なポリゴンで囲まれた平面オブジェクトです。
三角形オブジェクトは常に塗りつぶされて描画されます。

塗りつぶし色は、内部色属性、埋め込み色属性、グローバル色属性の順で優先度が高く決定されます。
すべての照明属性も適用されます。

状況によっては、三角形を単純な向きで作成し、その後移動と回転のコマンドを使って三角形を最終的な向きに配置する方がより簡潔な場合があります。

頂点ごとに三角形の色を設定でき、OpenGL が頂点間で色を補間します。

```
NewGizmo
AppendToGizmo/D triangle={0,0,0,1,1,1,1,-1,-1}
ModifyGizmo modifyobject=triangle0, objectType=triangle,
        property={colorType,2}
ModifyGizmo modifyobject=triangle0, objectType=triangle,
        property={colorValue,0,1,0,0,1}
ModifyGizmo modifyobject=triangle0, objectType=triangle,
        property={colorValue,1,0,1,0,1}
ModifyGizmo modifyobject=triangle0, objectType=triangle,
        property={colorValue,2,0,0,1,1}
```

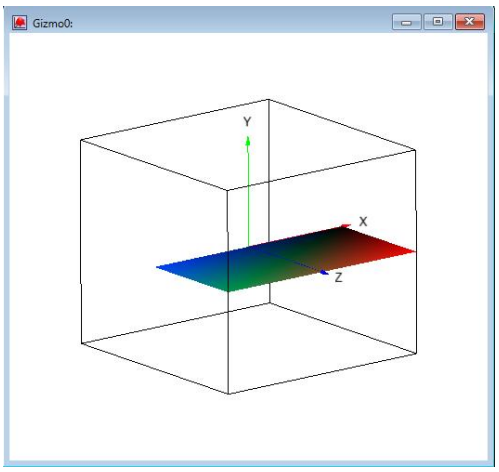


この例では、Object List で Axes と Free Axes Cue を追加して、Display List に追加しています。

Quad オブジェクト

Quad（四角形）オブジェクトは、最初の頂点から任意の方向に順次配置された４つの頂点を結ぶシートを塗りつぶします。
四角形オブジェクトは常に塗りつぶされて描画されます。
塗りつぶし色は、内部色属性、埋め込み色属性、グローバル色属性の順に優先度が高く決定されます。
四角形の法線は頂点ごとに計算されます。
これにより、光源計算が有効な場合にグラデーションシェーディングが生じます。
例えば、四角形を作成するコマンドは以下の通りです。

```
NewGizmo
AppendToGizmo/D quad={1,0,1,-1,0,1,-1,0,0,1,0,0}
ModifyGizmo modifyObject=quad0, objectType=quad,
        property={colorType,2}
ModifyGizmo modifyObject=quad0, objectType=quad,
        property={colorValue,0,1,0,0,1}
ModifyGizmo modifyObject=quad0, objectType=quad,
        property={colorValue,1,1.5259e-05,0.6,0.30425,1}
ModifyGizmo modifyObject=quad0, objectType=quad,
        property={colorValue,2,1.5259e-05,0.244434,1,1}
ModifyGizmo modifyObject=quad0, objectType=quad,
        property={colorValue,3,0,0,0,1}
```



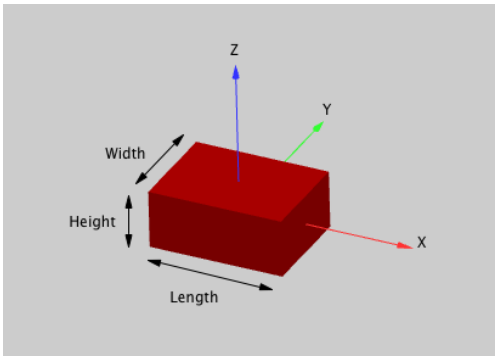
四角形を作成する時には、X、Y、または Z 平面のいずれかで単純な座標を使って四角形を作成し、その後、平行移動や回転コマンドを使って四角形を希望の最終的な向きに配置する方が容易である場合があります。

この表は、単位次元を使った基本的な四角形の例をいくつか示しています。

四角形の向き	**コマンド**
XZ 平面	quad={1,0,1,-1,0,1,-1,0,-1,1,0,-1}

XY 平面	quad={1,1,0,-1,1,0,-1,-1,0,1,-1,0}
YZ 平面	quad={0,1,1,0,1,-1,0,-1,-1,0,-1,1}
斜め Z 面、+X、+Y	quad={1,0,1,1,0,-1,0,1,-1,0,1,1}
斜め Z 面、-X、+Y	quad={-1,0,1,-1,0,-1,0,1,-1,0,1,1}
斜め Z 面、-X、-Y	quad={-1,0,1,-1,0,-1,0,-1,-1,0,-1,1}
斜め Z 面、+X、-Y	quad={1,0,1,1,0,-1,0,-1,-1,0,-1,1}

Box オブジェクト



Box（ボックス）オブジェクトは、X 方向の長さ、Y 方向の幅、Z 方向の高さで定義されます。
ボックスは原点を中心とし、常に塗りつぶされて描画されます。
塗りつぶし色は、内部色属性、埋め込み色属性、グローバル色属性の順に優先度が高く決定されます。

図は、長さ=1、幅=0.75、高さ=0.5 のボックスを示しています。

ボックスを、辺が軸と平行でない向きや原点を中心としない向きに配置するには、ボックスアイテムの前に回転および／または平行移動コマンドを表示リストに追加します。

Sphere オブジェクト

Sphere（球体）オブジェクトは二次曲面オブジェクトであり、二次式によって生成されることを意味します。
内部的には、二次曲面オブジェクトは曲面を近似する三角形と四角形のリストで構成されます。

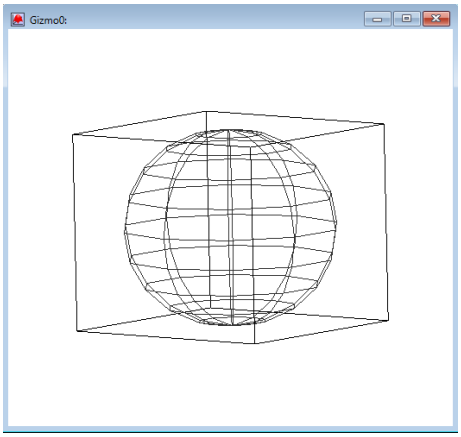
スライス数（Z 軸周りの細分化／経度の分割）とスタック数（Z 軸に沿った細分化／緯度の分割）によって、球体の滑らかさが決まります。
細分化の数が多いほど、球体の外観は滑らかになります。
値が小さいと他の幾何学的オブジェクトが生成されます（下図参照）。
分割数を増やすほど球体の描画に時間がかかりますが、散布図のマーカーとして使う場合など、球体を多数複製しない限り、ほとんどのアプリケーションでは問題になりません。

新しく作成された球体は原点を中心とし、両極は Z 軸上にあります。
デフォルトでは、球体は初期状態で塗りつぶされて描画されます。
塗りつぶし色は、内部色属性、埋め込み色属性、グローバル色属性の順に優先度が高く決定されます。

球体を他の位置や向きに配置するには、移動と回転コマンドを使います。

次のコマンドは、半径=1、スタック数=10、スライス数=10 の球体を生成し、その描画スタイルを線に設定します。

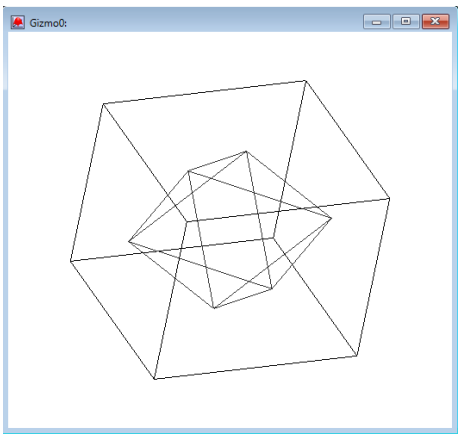
```
NewGizmo
AppendToGizmo/D sphere={1,10,10}
ModifyGizmo modifyObject=sphere0, objectType=Sphere,
        property={useGlobalAttributes,0}
ModifyGizmo modifyObject=sphere0, objectType=Sphere,
        property={drawStyle,100011}
```



（図には軸を追加しています）

スタック数を小さく設定し、スライス数を 2 に保つことで他の幾何学的形状を作成します。
次のコマンドは、半径=1、スタック数=4、スライス数=2 の八面体を生成し、描画スタイルを線に設定します。

```
NewGizmo
AppendToGizmo/D sphere={1,4,2}
ModifyGizmo modifyObject=sphere0, objectType=Sphere,
        property={useGlobalAttributes,0}
ModifyGizmo modifyObject=sphere0, objectType=Sphere,
        property={drawStyle,100011}
```



（図には軸を追加しています）

スタックの数を増やし、スライス数を 2 に保つことで、他の幾何学的形状を作成できます。

Cylinder オブジェクト

Cylinder（円柱）オブジェクトは二次曲面体であり、二次関数によって生成されます。
円柱はスライスとリングから構成されます。
スライス数を増やすほど滑らかな円柱が作成されます。
初期状態では、円柱軸は Z 軸を中心に配置され、高さは正の Z 方向を向きます。
円柱の底面は XY 平面上にあります。

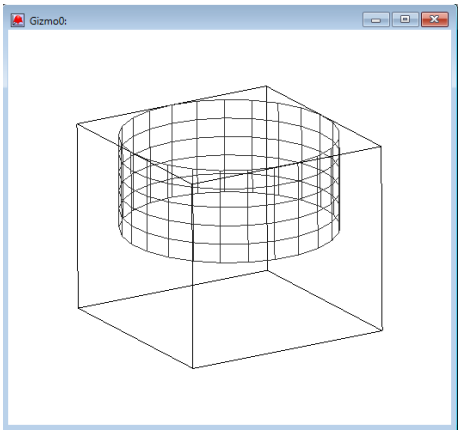
円錐形のオブジェクトを作成するには（下図参照）、底面半径と頂点半径のパラメーターに異なる値を指定します。

デフォルトでは、円柱は初期状態で塗りつぶされて描画されます。
塗りつぶし色は、内部色属性、埋め込み色属性、グローバル色属性の順に優先度が高く決定されます。

円柱を他の位置や向きに配置するには、移動と回転コマンドを使います。

次のコマンドは、baseRadius=1、topRadius=1、height=1、slices=25、rings=5 の円柱を生成し、その描画スタイルを線に設定します。

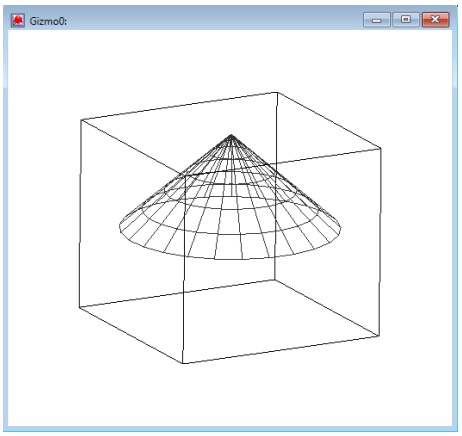
```
NewGizmo
AppendToGizmo/D cylinder = {1,1,1,25,5}
ModifyGizmo modifyObject=cylinder0, objectType=Cylinder,
        property={useGlobalAttributes,0}
ModifyGizmo modifyObject=cylinder0, objectType=Cylinder,
        property={drawStyle,100011}
```



（図には軸を追加しています）

同じコマンドを使いますが topRadius=0 を指定すると円錐が作成されます。

```
NewGizmo
AppendToGizmo/D cylinder = {1,0,1,25,5}, name=cone0

ModifyGizmo modifyObject=cone0, objectType=Cylinder,
property={useGlobalAttributes,0}

ModifyGizmo modifyObject=cone0, objectType=Cylinder,
property={drawStyle,100011}
```



スライス数を小さく指定することで、他の種類の円筒形状を作成できます。
スライス数=3 で三角柱を作成、スライス数=4 で四角柱または開放端のボックスを作成します。
底面半径または上面半径に異なる値を設定することで、ピラミッド形状を作成できます。

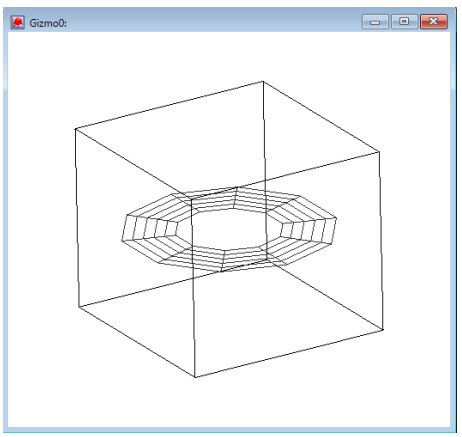
Disk オブジェクト

Disk（円盤）オブジェクトは二次曲面であり、二次関数によって生成されます。
円盤は初期状態では原点を中心とする XY 平面上に位置します。
スライス数を多く指定することで、より滑らかな円盤を作成できます。

デフォルトでは、ディスクは初期状態では塗りつぶされて描画されます。
塗りつぶし色は、内部色属性、埋め込み色属性、グローバル色属性の順に優先度が高く決定されます。

移動と回転コマンドを使って、ディスクを他の位置や向きに配置します。

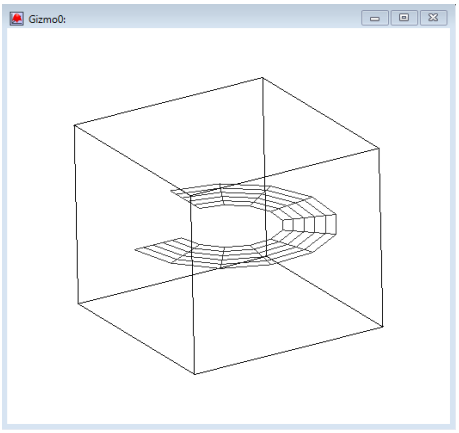
次のコマンドは、内半径=0.5、外半径=1、スライス数=10、スタック数=5、開始角度=0、掃引角度（SweepAngle）=360 のディスクを生成し、その描画スタイルを線に設定します。




```
NewGizmo
AppendToGizmo/D disk={0.5,1,10,5,0,360}
ModifyGizmo modifyObject=disk0, objectType=Disk,
        property={useGlobalAttributes,0}
ModifyGizmo modifyObject=disk0, objectType=Disk,
        property={drawStyle,100011}
```

次のコマンドはディスクを部分ディスクに変更し、sweepAngle を 360 ではなく 270 に設定します。
sweepAngle は startAngle から時計回りに度数で指定されます。

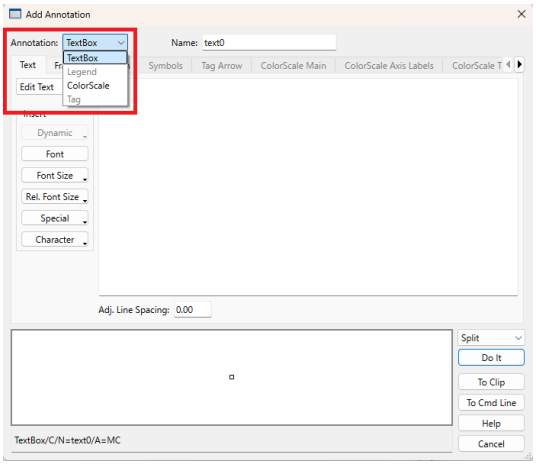
```
ModifyGizmo modifyObject=disk0, objectType=Disk,
        property={sweepAngle,270}
```



String オブジェクト

Gizmo では、グラフと同様に標準の Igor 注釈を使用できます。

注釈は、すべての 3D グラフィックスの前に平面上に平らに配置される 2D テキストグラフィックスです。
一般的なラベル付け目的に適しています。
注釈を作成するには、Gizmo→Add Annotation を選択し、Annotation ポップアップメニューから TextBox を選択します。
下位互換性のため、Gizmo は文字列オブジェクトを引き続きサポートしており、3D グラフィックスが必要な場合に便利です。
一般的なラベル付けには注釈が適しています。

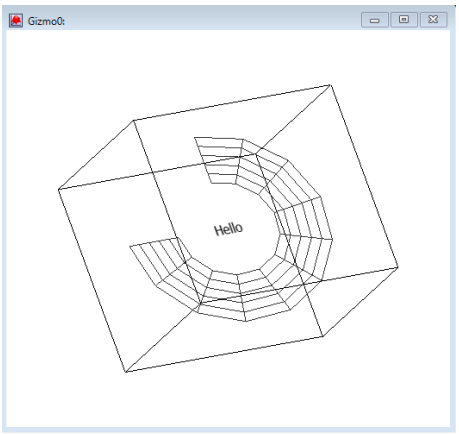


注釈と文字列オブジェクトの比較に関する詳細は、「Gizmo テキストの変更」のセクションを参照してください。
このセクションの残りの部分では、旧バージョンで使われていた Gizmo 文字列オブジェクトについて説明します。

文字列オブジェクトは、短いテキストラベルや注釈に使用できます。
次のコマンドで文字列オブジェクトを作成できます。

```
AppendToGizmo string="Hello", strFont="Geneva",
        name=string0
```

前述のプリミティブオブジェクトと同様に、文字列オブジェクトは ±1 表示ボリューム座標で描画される 3D オブジェクトです。
文字列をグラフの適切な位置に配置し、希望の向きに設定するには、移動と回転コマンドを使用できます。
文字列を正しく配置する鍵は、文字列が原点から発生し、文字の進みが X 方向、文字の高さが Y 方向であることを理解することです。

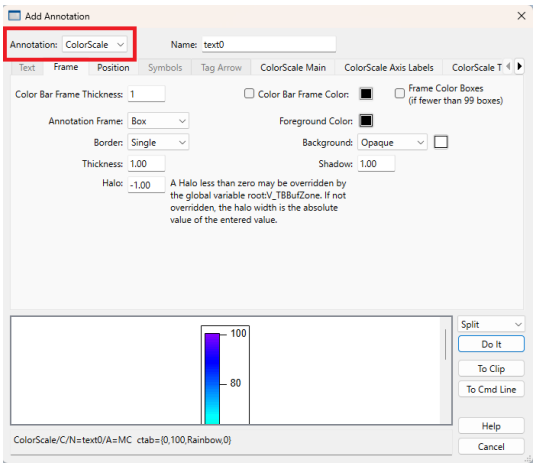


複数行はサポートされていないため、特定の文字列オブジェクトは単一行のテキストのみを生成します。

ColorScale オブジェクト

Igor Pro 6 およびそれ以前のバージョンでは、3D カラースケールグラフィックスの作成に Gizmo カラースケールオブジェクトが使われていました。
現在では、グラフと同様に標準の Igor 注釈を使うことができます。

注釈は、すべての 3D グラフィックスの前に平面上に平らに配置される 2D グラフィックスです。
一般的なラベル付け目的に適しています。
注釈としてカラースケールを作成するには、Gizmo→Add Annotation を選択し、Annotation ポップアップメニューから ColorScale を選択します。
下位互換のため、Gizmo はカラースケールオブジェクトを引き続きサポートしており、3D グラフィックスを作成する場合に便利です。
一般的なラベル付けには注釈が適しています。



このセクションの残りの部分では、旧バージョンで使われていた Gizmo ColorScale オブジェクトについて説明します。

カラースケールオブジェクトは、一連の色と数値スケールの関連付けを提供するように設計されています。
次のようなコマンドを使ってカラースケールオブジェクトを作成できます。

```
AppendToGizmo colorScale=colorScale0
```

通常、この後には ModifyGizmo コマンドを続けます。

カラースケールも ±1 の表示ボリューム座標で描画され、デフォルトでは平面状に表示されますが、正の深度値を割り当てることでカラースケールを完全な 3D オブジェクトにすることができます。

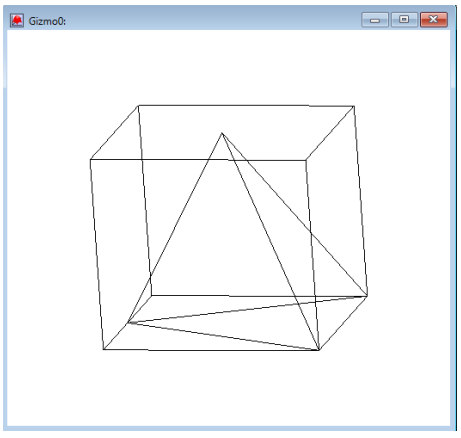
内蔵のカラーテーブルに基づく色の順序を選択するか、カラーウェーブを使って独自の順序を指定できます。

カラースケールは、表面プロットなどのウェーブベースのデータオブジェクトに関連付けることもできますが、プロット内のすべてのオブジェクトから独立して設定することも可能です。
同じプロット内で複数のカラースケールオブジェクトを作成できます。

Tetrahedron オブジェクト

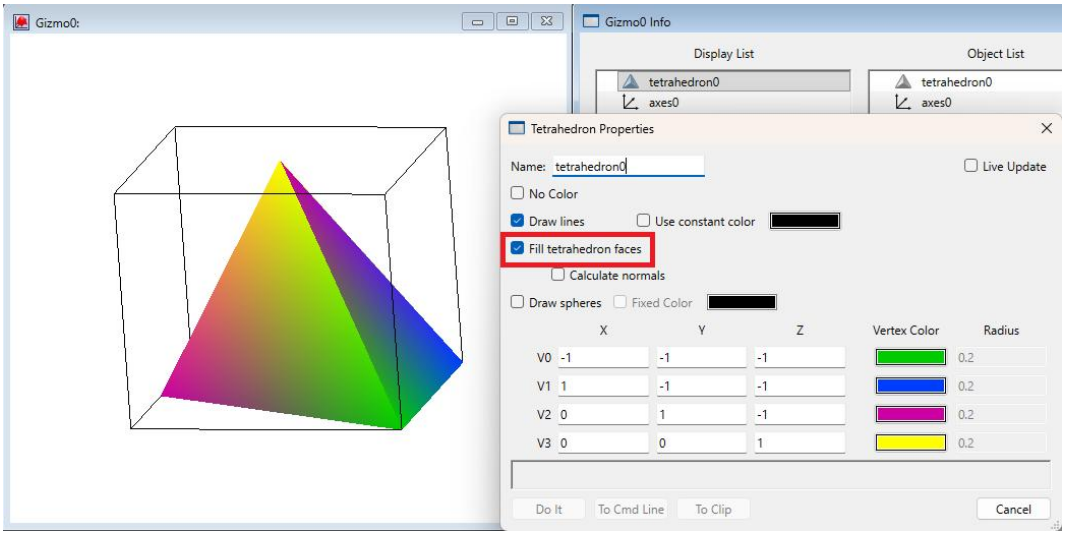
Tetrahedron（四面体）オブジェクトは４つの頂点によって定義されます。
次のコマンドは四面体を生成し、その描画スタイルを線に設定します。

```
NewGizmo
AppendToGizmo/D tetrahedron=tetrahedron0
ModifyGizmo ModifyObject=tetrahedron0,
        objectType=tetrahedron, property={vertex0,-1,-1,-1}
ModifyGizmo ModifyObject=tetrahedron0,
        objectType=tetrahedron, property={vertex1,1,-1,-1}
ModifyGizmo ModifyObject=tetrahedron0,
        objectType=tetrahedron, property={vertex2,0,1,-1}
ModifyGizmo ModifyObject=tetrahedron0,
        objectType=tetrahedron, property={vertex3,0,0,1}
```

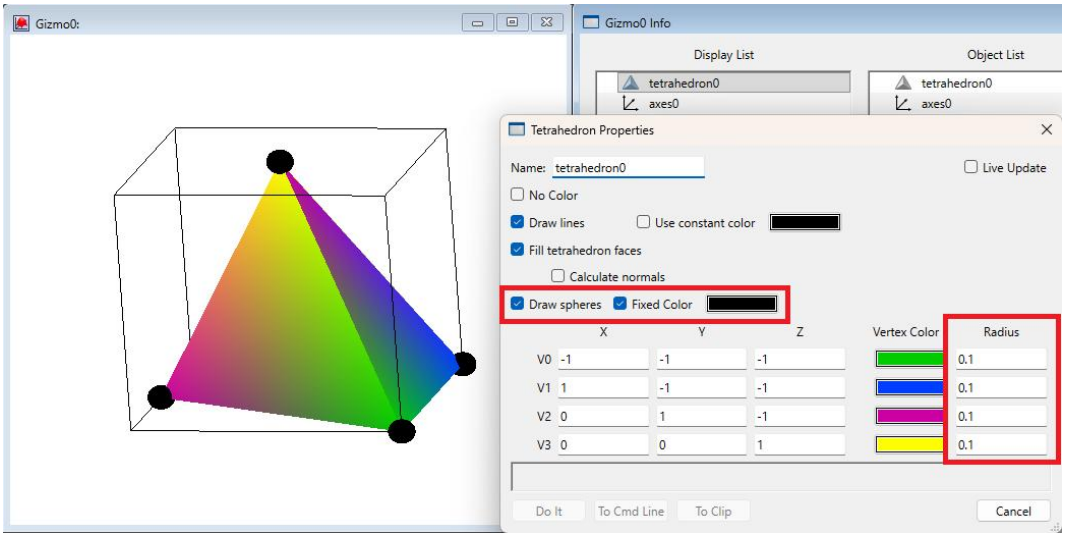


（図には軸を追加しています）

各頂点に塗りつぶしを有効化し色を指定することで、次のような四面体を生成できます。



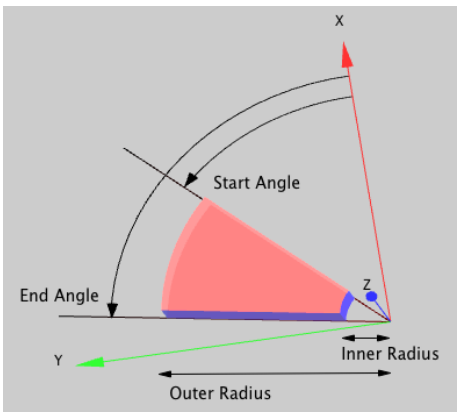
また、各四面体の頂点に球体を描画するよう指定することもできます。



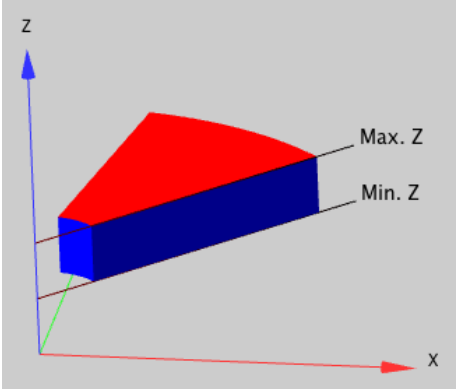
Pie Wedge オブジェクト

Pie Wedge（パイウェッジ／パイの一切れ [楔形]）オブジェクトは、２つの角度、２つの半径、２つの Z 値によって定義されます。
複数のパイウェッジオブジェクトを使うことで、様々な種類の 3D 円グラフを作成できます。

この図は上から見たパイの一切れを示しています。



この図は、横から見たパイの一切れを示しています。



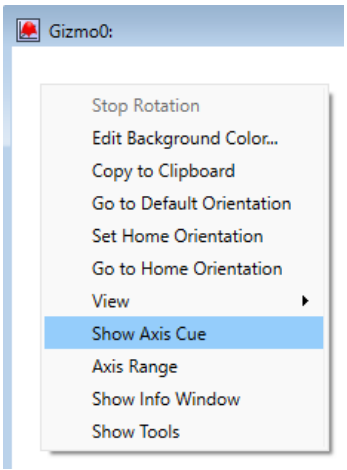
Gizmo の Axis と Axis Cue オブジェクト

Gizmo の軸オブジェクトは、主にサーフェスプロットや散布図などのウェーブベースのデータオブジェクトで使われます。
これにより、データ値の範囲を指定し、データオブジェクトのスケールを設定できます。

Axis Cue オブジェクトは、X、Y、Z 方向の向きを示します。
Axis オブジェクトは、向きと数値範囲の両方を示します。

Axis Cue オブジェクト

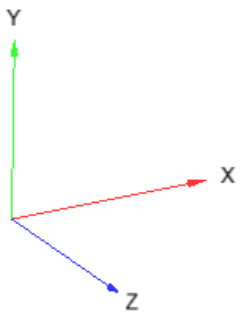
Gizmo は 2 種類の Axis Cue オブジェクトをサポートします：デフォルト Axis Cue と自由 Axis Cue です。
これらのオブジェクトは X、Y、Z 軸の向きを示します。



デフォルトの Axis Cue は、表示ボリュームの原点を中心とするトリプレット軸オブジェクトです。
X、Y、Z のキューラインは、それぞれ表示ボリュームの正の X、Y、Z 方向に伸びています。

デフォルトの Axis Cue は 1 つだけです。
これはどの方向がどちらかを示すためだけに使われます。
目盛線、目盛ラベル、軸ラベルはサポートしていません。
デフォルトの軸キューを表示するには、Gizmo 表示ウィンドウを右クリックし、Show Axis Cue を選択します。

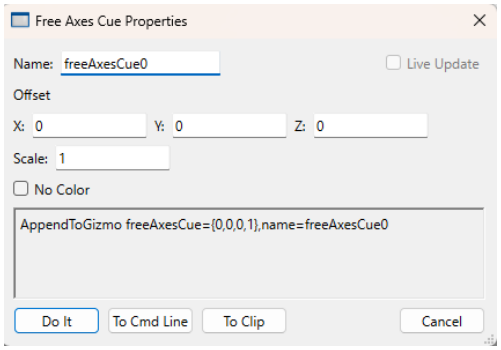
デフォルトの Axis Cue オブジェクトは、X 軸が赤、Y 軸が緑、Z 軸が青で描画されます。
3 つの軸をラベル付けする文字はすべて黒で描画されます。
内部的には、軸は発光色素材で描画されており、これにより Axis Cue を背景から区別しやすくなっています。



自由 Axis Cue オブジェクトは、原点からの任意のオフセット位置と任意のスケールで描画できます。
デフォルトでは、デフォルト Axis Cue と同じ向きになります。
回転させたい場合は回転コマンドを使ってください。

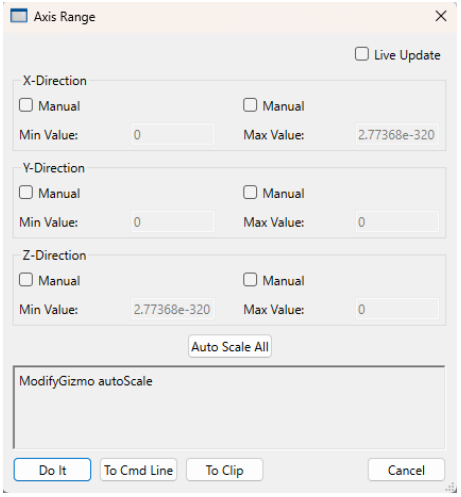
複数の自由 Axis Cue を作成でき、散布図でマーカーとして使うことも可能です。
自由 Axis Cue を追加するには、Gizmo 情報ウィンドウのオブジェクトリストにある「+」アイコンをクリックし、Free Axis Cue を選択します。

自由 Axis Cue は、Free Axis Cue ダイアログで No Color チェックボックスをオンにしてその色を無効にしない限り、デフォルトの Axis Cue と同じ色になります。
この場合、Display List の前の色属性で設定された、描画環境の現在の色になります。
また、自由軸に色を適用するには、Display List に色素材コマンドも必要です。



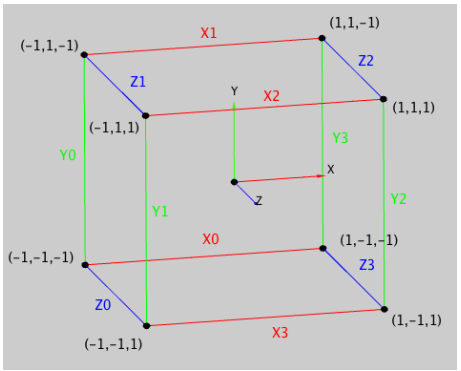
Axis オブジェクト

「Gizmo の次元」のセクションで説明したように、Gizmo の表示には、+/-1 の表示ボリュームに重ねられた軸座標系があります。
この軸座標系の範囲は、Gizmo メニューからアクセスできる Axis Range ダイアログで設定します。
散布図やサーフェスプロットなどのウェーブベースのオブジェクトは、この軸座標系に対して表示されます。
軸オブジェクトは、その視覚的な表現です。



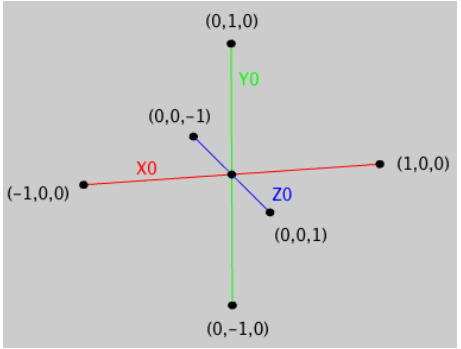
Gizmo は 3 種類の軸オブジェクトをサポートします：Box、Triplet、Custom。
Gizmo は軸オブジェクトを、Gizmo 表示ウィンドウに追加する他のオブジェクトと同様に扱います。
ただし 1 つの例外があります。
軸オブジェクトが意味を持つためには、Display List 上に少なくとも 1 つのデータオブジェクトまたは描画オブジェクトが存在している必要があります。

ボックス軸オブジェクトの頂点は、表示ボリュームの頂点に対応します。
したがって、表示ボリューム空間における各頂点の位置は、各次元で -1 または +1 となります。
この図に示されている座標は、ボックス軸の頂点の表示ボリューム座標です。



ボックス軸は、X0、X1、X2、X3、Y0、Y1、Y2、Y3、Z0、Z1、Z2、Z3 という 12 本の軸で構成されています。
各軸を個別にコントロールし、色、範囲、目盛り、目盛ラベルを割り当てることができます。
また、グリッド線を追加したり、ボックスの 6 つの面のいずれかを塗りつぶしたりすることもできます。

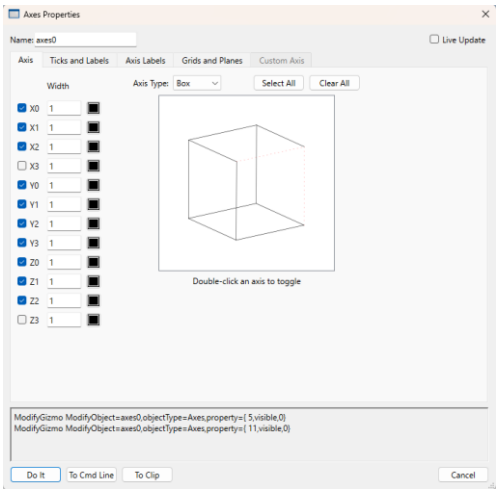
トリプレット軸は、原点で交差し、±1 表示ボリュームをスパンする 3 つの直交軸のシステムです。
各軸を個別にコントロールし、目盛、目盛ラベル、色を割り当てることができます。
軸の原点は、±1 表示キューブの中心です。



各トリプレット軸は、X0、Y0、Z0 という名前で識別されます。
右の図は、トリプレット軸の表示体積座標と各軸の名前を示しています。

カスタム軸は、-1/+1 表示ボリューム座標で指定された、選択した開始点と終了点を結ぶ空間内の単一線です。
ボックス軸やトリプレット軸が Gizmo→Axis Range で設定されたグローバル軸範囲を反映するのとは異なり、カスタム軸は独自の独立した軸範囲を持ちます。
カスタム軸には、他の軸と同様に目盛線と目盛ラベルを追加できます。

デフォルトでは、Gizmo は目盛ラベルの適切な位置を自動的に探します。
手動位置モードを使うと、それらの位置と向きをコントロールできます。



ほとんどの場合、Axes Properties ダイアログを使って、さまざまな軸の設定をコントロールします。
ダイアログを使う時には、チェックボックスで選択できるさまざまな軸を把握しておくことが重要です。
プロット内のさまざまな軸を識別しやすくするために、ダイアログの Axis タブには、Gizmo ウィンドウに表示されるのと同じ向きで軸のアウトラインが表示されます。
軸をダブルクリックすると、軸の表示を切り替えることができます。

プログラミングの詳細については、ヘルプ AppendToGizmo およびヘルプ ModifyGizmo for Axis Objects を参照してください。

Gizmo ウェーブデータフォーマット

サーフェスプロット、パスプロット、等値面、ボクセルグラムなどのオブジェクトは、ウェーブ内のデータに基づいており、したがって「ウェーブベースのデータオブジェクト」、または略して「データオブジェクト」と呼ばれます。
データを提供するウェーブは「データウェーブ」または「ソースウェーブ」と呼ばれます。

以下のセクションでは、異なるデータオブジェクトに対するデータ形式の要件について説明します。

散布、パス、リボンデータのフォーマット

散布図およびパスプロットには、各頂点の X、Y、Z 座標の 3 列を含む 2 次元ウェーブであるトリプレットが必要です。
散布図またはパスプロット用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 2 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。

リボンプロットにはトリプレットウェーブも必要です。
リボンプロット用のカラーウェーブは、散布図やパスプロットと同様で、頂点ごとに 1 行の RGBA 値を持ちます。
キーワード pathToRibbon を使った ModifyGizmo および「リボンプロット」のセクションも参照してください。

サーフェスオブジェクトデータフォーマット

データウェーブフォーマットは、サーフェスプロットの種類によって異なります。

２次元ウェーブ（別名「行列」または「Z 値の行列」）から単純なサーフェスプロットが生成されます。
このタイプのサーフェスプロットにおけるカラーウェーブは 3D RGBA ウェーブであり、ウェーブの４つのレイヤーには [0,1] の範囲の R、G、B、A 成分が含まれます。

パラメトリックサーフェスをプロットする場合、データウェーブは３つのレイヤーを含む 3D ウェーブとなります。
各行/列の位置において、レイヤー0 は X 座標、レイヤー1 は Y 座標、レイヤー2 は Z 座標を含みます。
パラメトリックサーフェスのカラーウェーブは 3D RGBA ウェーブであり、データウェーブと同じ X・Y 次元の要素数を持ちます。
レイヤー0 は赤成分、レイヤー1 は緑成分、レイヤー2 は青成分、レイヤー3 はアルファ成分を含みます。

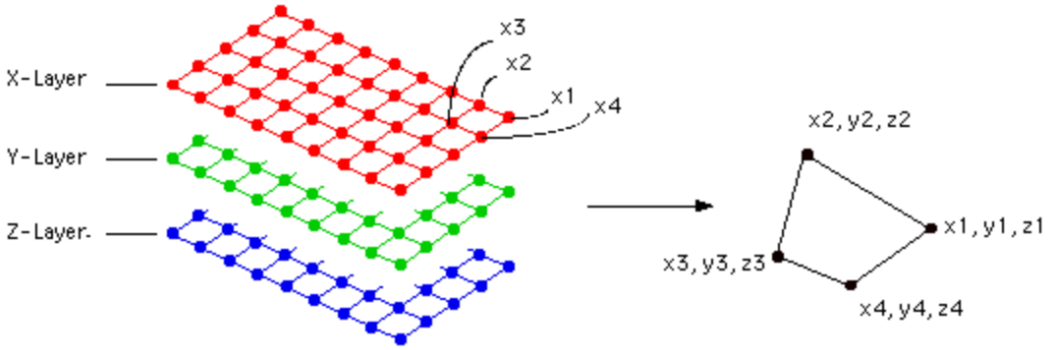
連続する四角形（Catmull-Clark B-スプラインの ImageTransform 出力など）を描画する場合、データウェーブは４行３層の 3D ウェーブとなります。
各行には四角形の４つの頂点が格納され、３層にはそれぞれ X 座標、Y 座標、Z 座標が含まれます。

分離された四角形をプロットする場合、データウェーブは２次元で、12 列からなり、反時計回りに取得した四角形頂点の X、Y、Z 値に対応します。

順次四角形および分離四角形の色ウェーブは、４列の 2D ウェーブであり、各列は [0,1] の範囲の RGBA 値に対応します。

パラメトリックサーフェスデータ形式

パラメトリックサーフェスは、通常、一組のパラメーターに対して X、Y、Z 座標が計算される方程式によって定義されます。
Gizmo でパラメトリックサーフェスを定義するには、X、Y、Z レイヤーを含む 3D ウェーブが必要です。
このウェーブの次元に関する唯一の制限は、４列であってはならないことです。
Gizmo は、四角形からパラメトリックサーフェスを構築します。
各四角形は、下図に示すように隣接する４つの頂点によって定義されます。



パラメトリックサーフェスの例は、以下のデモエクスペリメントで見つけることができます（File→Example Experiments→Visualization）。

- Mobius Demo
- Spherical Harmonics Demo
- Gizmo Sphere Demo

画像オブジェクトデータフォーマット

画像オブジェクトのデータウェーブは、次の３つの形式のいずれかです。

- Z 値の２次元行列

- ３層の符号なしバイト型の 3D RGB ウェーブ
- ４層からなる符号なしバイト型の 3D RGBA ウェーブ

3D フォーマットでは、赤成分はレイヤー0 に、緑成分はレイヤー1 に、青成分はレイヤー2 に格納されます。
アルファ成分がある場合は、レイヤー3 に格納されます。
各成分の値は 0 から 255 の範囲です。

等値面およびボクセルグラムオブジェクトデータ形式

等値面またはボクセルグラムオブジェクトのデータは、３次元体積ウェーブです。

等値面とボクセルグラムはカラーウェーブを使いません。

ウェーブ内の NaN 値

一般的に、Gizmo でプロットされるデータウェーブでは NaN の使用を避けるべきです。
NaN はサーフェスでは穴として、パスプロットでは不連続点として表示されます。
データウェーブに１つ以上の NaN 値を含むサーフェスオブジェクトは通常通り描画されますが、少なくとも１つの頂点が NaN である構成三角形は描画されません。
可能な場合は、Gizmo で欠損データを透明度を使って表示するのが最適です（「透明と半透明」のセクションを参照）。