1　Cindy3D

Cinderella でレイトレーシングを用いた空間図形を描くには，同梱されている Cindy3D プラグインを用いる。Initailiazarion スロットの先頭に次の 1 行を書く。

```
use("Cindy3D");
```

これで Cindy3D の各関数が使えるようになる。スクリプトを書いて実行すると別ウィンドウが開いて描画がされる。

【重要な注意】
Cindy3D は，ファイルメニューの「HTML に書き出す」で HTML ファイルに書き出しても CindyJS では使えない。CindyJS で 3D を扱うには，HTML ファイルを直接編集する必要がある。

1.1　設定

以下の関数は初期値の設定なので，最初に一度だけ実行すればよく， Initialization スロットに `use("Cindy3D");` に続いて記述すればよい。

色の初期設定：`color3d([R,G,B])`
すべてのオブジェクトの表示色の初期値を RGB 値に設定する。
【例】 `color3d([0.8,0.8,0]);` 表示色を少し暗い黄色に設定する。

点の色の初期設定：`pointcolor3d([R,G,B])`
点の表示色の初期値を RGB 値に設定する。

線の色の初期設定：`linecolor3d([R,G,B])`
線の表示色の初期値を RGB 値に設定する。

面の色の初期設定：`surfacecolor3d([R,G,B])`
面の表示色の初期値を RGB 値に設定する。

透明度の初期設定：`alpha3d(<real>)` または `surfacealpha3d(<real>)`
面の透明度の初期値を設定する。引数は 0 以上 1 以下の実数。

光沢の初期設定：shininess3d(<real>)
　すべてのオブジェクトの光沢の初期値を<real>に変更する。

点の光沢の初期設定：pointshininess3d(<real>)
　点の光沢の初期値を<real>に変更する。

線の光沢の初期設定：lineshininess3d(<real>)
　線の光沢の初期値を<real>に変更する。

面の光沢の初期設定：surfaceshininess3d(<real>)
　面の光沢の初期値を<real>に変更する。

サイズの初期設定：size3d(<real>)
　点と線のサイズの初期値を<real>に変更する。

点のサイズの初期設定：pointsize3d(<real>)
　点のサイズの初期値を<real>に変更する。

線の太さの初期設定：linesize3d(<real>)
　線のサイズの初期値を<real>に変更する。

1.2　描画関数

　以下の関数は Draw スロットに記述する。

Cindy3D の描画開始：begin3d()

Cindy3D の描画終了：end3d()
　Cindy3D の描画関数の使い始めと使い終わりを宣言する。Cindy3D の描画関数は、begin3d() から end3d() までの間に書かれたものが実行される。

現在の描画設定の保存：gsave3d()
　現在の描画に関する諸設定をスタックに保存する。

描画設定の復帰：grestore3d()
　スタックに保存された描画に関する諸設定を呼び出して、その状態に復帰する。

点を描く：draw3d(<point>)

座標 `<point>` に点を打つ。

修飾子	値	効果
size	＜ real ＞	点の大きさを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する

線を描く：`draw3d(<point1>,<point2>)`

線分，反直線，直線を描く。線の種類は type 修飾子で指定する。指定がなければ線分が描かれる。２つの引数は、線の種類に応じて解釈される。

線種	point1	point2
線分	始めの端点	終わりの端点
反直線	始点	通る点
直線	直線上の点	直線上のもう一つの点

修飾子	値	効果
type	＜ string ＞	segment,ray,line のいずれか
size	＜ real ＞	線の太さを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	線の色を RGB 値でを指定する
shininess	＜ real ＞	光沢を指定する

【例】次の例は 3 種類の線を描画する。

・ (0,0,0) と (1,0,0) を端点とする線分

```
draw3d([0,0,0],[1,0,0],color->[1,0,0])
```

・ (0,0,0) と (0,1,0) を端点とする緑色の線分

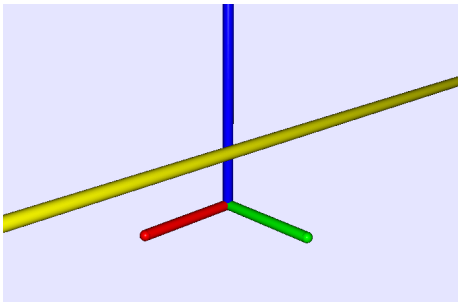
```
draw3d([0,0,0],[0,1,0],type->"segment",color->[0,1,0])
```

・(0,0,0) を始点として、(0,0,1) を通る青の半直線

```
draw3d([0,0,0],[0,0,1],type->"ray",color->[0,0,1])
```

・ (1,1,1) と (2,1,1) を通る黄色の直線

```
draw3d([1,1,1],[2,1,1],type->"line",color->[1,1,0])
```



点を結ぶ：`connect3d(<list>)`

`<list>`で与えられた各点を線分で結ぶ。

修飾子	値	効果
size	＜ real ＞	線の太さを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する

多角形を描く：`drawpoly3d(<list>)`

`<list>`で与えられた各点を線分で結んで多角形を描く。

修飾子	値	効果
size	＜ real ＞	線の太さを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する

多角形の面を描く：`fillpoly3d(<list>)`

`<list>`で与えられた各点を線分で結んで多角形の面を描く。

修飾子	値	効果
size	＜ real ＞	面の大きさを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する
alpha	＜ real ＞	透明度を指定する

法線ベクトルを指定して多角形の面を描く：fillpoly3d(<list1>,<list2>)

ユーザー定義の法線ベクトルによる多角形の面を描く。法線ベクトルは、光の当たり方を計算するものである。

<list1>は多角形の頂点の座標，<list2> は多角形の各頂点の法線ベクトル。 <list1> と <list2> の長さは一致する必要がある。

修飾子	値	効果
size	＜ real ＞	面の大きさを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する
alpha	＜ real ＞	透明度を指定する

円盤を描く：fillcircle3d(<point>,<vec>,<real>)

<point>を中心、<vec>を法線ベクトル、<real>を半径とする円盤を描く。

修飾子	値	効果
size	＜ real ＞	面の大きさを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する
alpha	＜ real ＞	透明度を指定する

【例】座標軸と円盤を描く。

Initialization スロットに次のコードを書く。以後の例も同様。

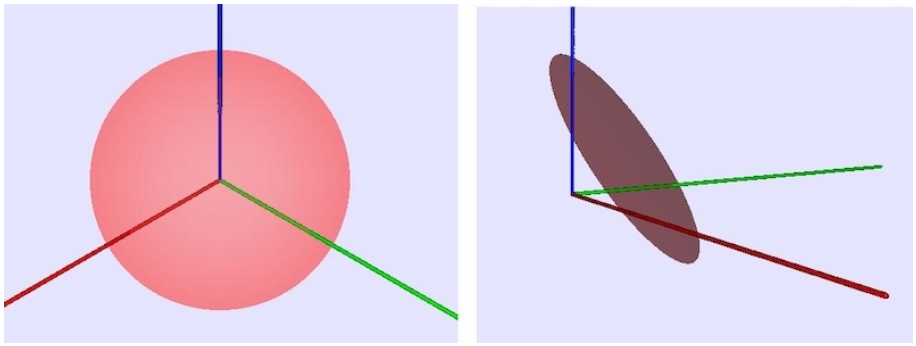
```
use("Cindy3D");
background3d([0.9,0.9,1]);
renderhints3d(quality->4);
lookat3d([4,4,4],[0,0,0],[-1,-1,0]);
```

draw スロットに次のコードを書く。

```
renderhints3d(quality->4);
begin3d();
draw3d([0,0,0],[3,0,0],color->[1,0,0],size->0.3);  // 座標軸
```

```
draw3d([0,0,0],[0,3,0],color->[0,1,0],size->0.3);
draw3d([0,0,0],[0,0,3],color->[0,0,1],size->0.3);
fillcircle3d([sqrt(3)/6,sqrt(3)/6,sqrt(3)/6],[1,1,1],
  sqrt(3)/2,color->[1,0,0],alpha->0.3);
end3d()
```

`renderhints3d(quality->4)` はレンダリングの品質を指定する関数。（後述）
左が実行結果。右はマウスで画面上をドラッグし，回転したもの。



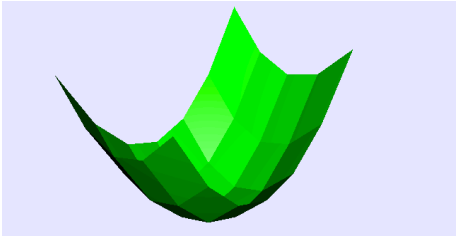
球面を描く：`drawsphere3d(<point>,<real>)`
`<point>`を中心、`<real>`を半径とする球面を描く。

修飾子	値	効果
size	＜ real ＞	面の大きさを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する
alpha	＜ real ＞	透明度を指定する

網目上の曲面を描く：`mesh3d(<int1>,<int2>,<list>)`
曲面を、m 行 n 列の格子点でできる網目状に区切る。（メッシュモデル）
`<int1>` 格子点の行数 m
`<int2>` 格子点の列数 n
`<list>` 格子点のリスト
リストは 2 次元の格子点のリストを平坦化したもので，リストの長さは m × n 。

【例】図は、５行７列の格子点からなる放物線状の面。修飾子の効果を比較するため、かなり荒い網目にしてある。

```
begin3d();
pt=apply(-2..2,s,
       apply(-3..3,t,
         y = s/3;
         x = t/3;
         z = x^2+y^2;
         (x,y,z);
       );
   );
pt=flatten(pt,levels->1);
mesh3d(5,7,pt);
end3d()
```

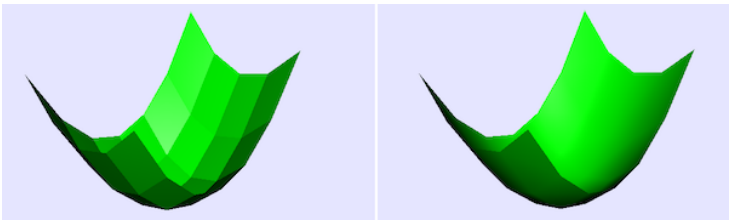


apply 関数のネストにより作成した格子点のリストを、flattenn() により平坦化して引数に与えている。

修飾子	値	効果
nomaltype	＜ string ＞	法線のタイプを指定する。値は ”perface” , ”pervertex”
topology	＜ string ＞	topology を指定する。値は ”open”, ”closerows”, ”closecolumns”, ”closeboth” のいずれか
size	＜ real ＞	面の大きさを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する
alpha	＜ real ＞	透明度を指定する

「normaltype」修飾子は，各面の法線の計算方法を指定する。指定がなければ「perface 」として処理される。

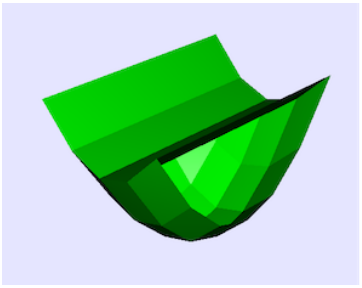
nomaltype	説明
perface	各面の法線はその面上の三角形の法線。その結果、面の端で光の当たり方が不連続になり、格子構造が明らかになる。
pervertex	各面の法線はその面上の三角形の頂点の 3 本の法線をとり、その一次結合によって計算される。その結果、面の端での光の当たり方が連続的になり、格子構造が見えなくなる。



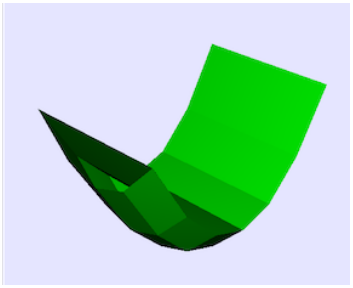
perface　　pervertex

`topology` 修飾子は面の端の状態を指定する。指定がなければ、open として処理される。

topology	説明
open	端点もしくは面の端の点までが面になる。その結果、(m-1) × (n-1) 個の矩形ができる。 面は両サイドと 1 つの境界を持つ。
closerows	各行の最初と最後の頂点が結合されて対応する面ができる。その結果、(m-1) × n 個の矩形ができる。面は両サイドと 2 つの境界を持つ。
closecolumns	各列の最初と最後の頂点が結合されて対応する面ができる。その結果、m × (n-1) 個の矩形ができる。面は両サイドと 2 つの境界を持つ。
closeboth	各行の最初と最後および各列の最初と最後の頂点が結合されて対応する面ができる。その結果、m × n 個の矩形ができる。面は両サイドを持ち境界はない。



closerows



closecolumns

法線ベクトルを指定して網目を描く：`mesh3d(<int1>,<int2>,<list1>,<list2>)`

ユーザー定義による法線ベクトルによって網目を描く。

`<int1>` 格子点の行数： m　`<int2>`：格子点の列数 n　`<list>`：格子点のリスト

`<list2>` 各格子点での法線ベクトルのリスト。

リストの長さはいずれも m × n 。

修飾子	値	効果
topology	＜ string ＞	topology を指定する。 値は open, closerows, closecolumns, closeboth のいずれか
size	＜ real ＞	面の大きさを指定する
color	[＜ real ＞, ＜ real ＞, ＜ real ＞]	色を RGB で指定する
shininess	＜ real ＞	光沢を指定する
alpha	＜ real ＞	透明度を指定する

1.3　光の当て方と表現

背景の色を設定する：`background3d(<colorvec>)`

背景の色 `<colorvec>` を RGB 値で設定する。

カメラの位置：`lookat3d(<point1>,<point2>,<vec>)`

３Ｄグラフィクスの表示は、空間に置いたカメラで物体を写していると考える。この関数は、カメラの位置と向きを設定する。カメラにはレンズがついていて、レンズの向き（視線の方向）で物を見ていると考える。

`<point1>`：カメラの位置　`<point2>`：回転の中心　`<vec>`：視線の方向

画角の設定 ：`fieldofview3d(<real>)`

カメラの画角を設定する。実際のカメラの画角と同じ。画角が小さいと望遠（被写体が大きく写る)、大きいと広角（小さく写る）になる。初期設定は 45° 。

カメラ深度の設定：`depthrange3d(<real1>,<real2>)`

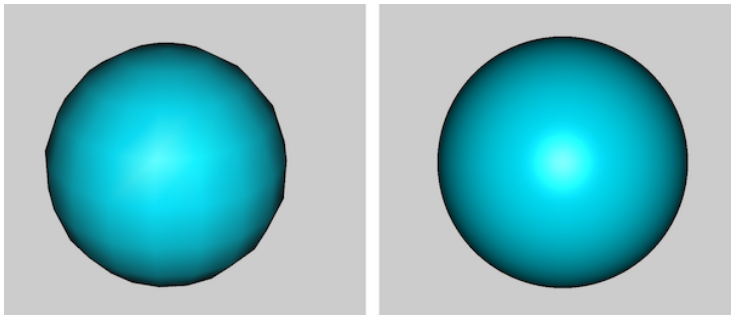
カメラの最小・最大深度を設定する。カメラの深度とは、カメラ面（カメラを通り、視線方向に垂直な面）と点の距離。カメラ深度に入らないものは表示されない。第 1 引数が最小値、第 2 引数が最大値。

レンダリングのヒントを設定する：`renderhints3d()`

レンダリングの過程における様々な比率のヒントを設定する。

修飾子	値	効果
quality	＜ int ＞	品質レベルを選ぶ。値は 0 以上 8 以下
renderMode	＜ string ＞	レンダリングモードを指定する。 値は ”sinple” か ”raycated”
sampleRate	＜ int ＞	ピクセルごとのサンプル数を設定。1 以上の整数。
screenError	＜ real ＞	ピクセルにおける最大の screen space error を 設定する。値は 0 より大きな実数

”quality” 修飾子は規定の品質レベルのいずれかを選ぶ。レベル 0 は最低の品質だが、最小のリソースですむ。最大の品質は 8 で、非常によい品質だが多くのリソースを必要とする。あらかじめ品質レベルが設定されているのは、個々のレンダリングヒントを操作することなく、全体の品質を簡単に管理できるようにするため。要請された品質レベルがサポートされない（例えばハードウェアの制限やリソースの制約のため）とき、Cindy3D は下位のレベルに移行するかもしれない。



`quality->1`　　`quality->4`

”renderMode” 修飾子は、オブジェクトのレンダリングについて指定する。”simple”の場合は、すべてのオブジェクトは三角形の網目としてレンダリングされる。このモードではシェーディングは頂点ごとに行われ、上図左のように粗削りになる。 ”raycasted”の場合は、点・直線・球面はレイ・キャスティングを用いた連続面としてレンダリングされる。また、シェーディングは点ごとに行われる。上図右のようになる。 ”raycasted” モードでは高品質が得られるがハードウェアの条件によっては時間がかかる。

”screenError” 修飾子は、スクリーン・スペース・エラーをデティール・アルゴリズムのレベルに設定する。 ”simple” モードでは、 点・直線・球面は三角形の網目で近似される。最適化のために、小さい、あるいは遠いオブジェクトはレンダリング時間を節約するために少しの三角形網目にする。これを ”level of detail”と呼ぶ。 Cindy3D は、それぞれのプリミティブに対して、異なる三角形網目ごとに決められた定数を用いている。あるプリミティブに対しどの定数を用いるかは、各網目をスクリーン上に仮想的に投影し、ピクセルごとに最大の三角形の大きさを計測して決定される。それから、最大と予測される三角形サイズが「screenError」の下にある最も小さな網目が、プリミティブを生成するために使われる。これは、「screenError」の低い値がより高い品質となることを意味する。この修飾子は ”raycasted” レンダリングモードのもとでは無効になる。

”samplingRate” 修飾子は、オブジェクトのシルエットの滑らかさに影響する。サンプリング・レートは、出力イメージの各々のピクセルに対するサンプルの数を定める。最終的なピクセルの色はそれらのサンプルの平均値。サンプリングレートが高いほど、記憶領域と時間を消費するがオブジェクトシルエットはより滑らかになる。要請されたサンプリング・レートをサポートできないとき（例えばハードウェアの制限やリソースの制約のための)、Cindy3D は低いサンプリング・レートに移行するかもしれない。

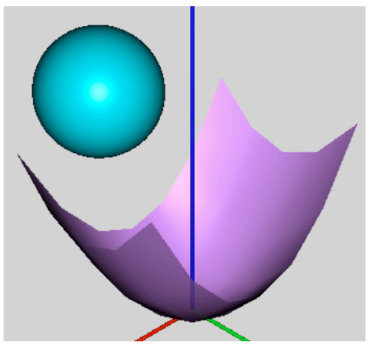
※訳者の実験では、品質は quality の例で示した 2 通りくらいで、数値を変えてもあまり変化はなかった。（ハードウェアの制限などによるかもしれない）renderMode の ”simple” , ”raycasted” の違いも同様。あとの 2 つの修飾子の効果については翻訳時点では不明だった。デフォルトでは `quality->1` なので、 `renderhints3d(quality->4)` もしくは `renderhints3d(renderMode->"raycasted")` で運用するのがよさそうだ。

点光源の設定：`pointlight3d(<int>)`

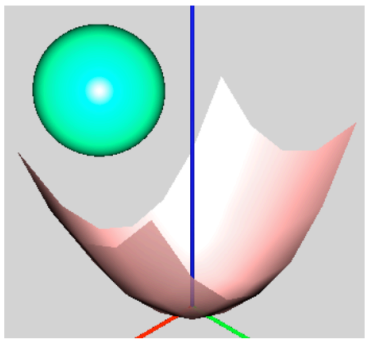
`<int>` は，光源の番号で 0 以上 7 以下の整数。

点光源を発生または修正する。指定された光源がすでに存在するならば修飾子によって指定された状態に修正し、利用可能にする。そうでなければ、指定された点光源を作る。修飾子がなければ初期値が使われる。点光源は 8 つまで作ることができ、それぞれに番号を振る。

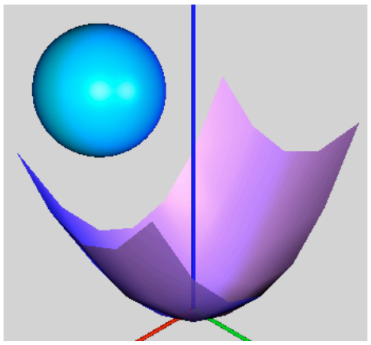
修飾子	値	効果
ambient	[R,G,B]	周囲の色を RGB 値で指定された色にする (初期値は [0,0,0])
diffuse	[R,G,B]	拡散する光の色を RGB 値で指定された色にする (初期値は [1,1,1])
specular	[R,G,B]	反射光の色を RGB 値で指定された色にする (初期値は [1,1,1])
position	＜ point ＞	点の位置 (初期値は [0,0,0])
frame	＜ string ＞	位置がカメラフレームに依存するか、絶対位置かを指定する。値は”camera” か ”world” で、初期値は ”camera”



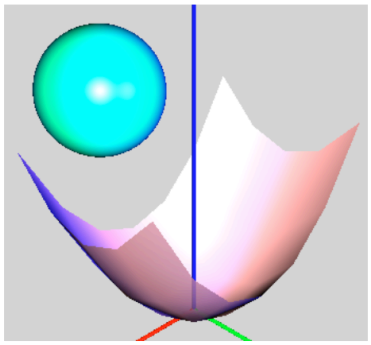
点光源の指定なし



点光源 1 `diffuse->[1,1,0]`



点光源 2 `position->[8,0,0],diffuse->[0,0,1])`



点光源 1 と 2

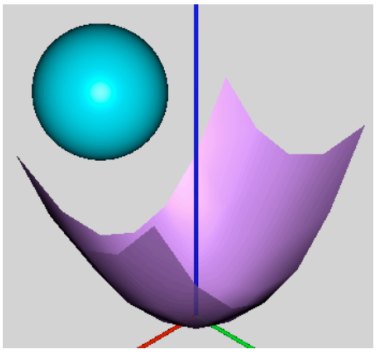
方向光源の設定：`directionallight3d(<int>)`

`<int>`は，方向光源の番号で 0 以上 7 以下の整数。

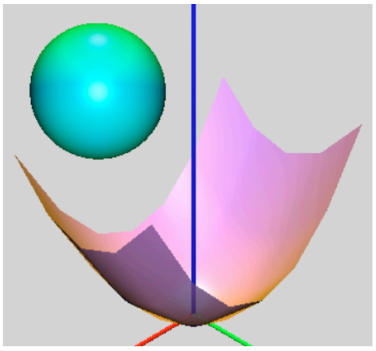
方向光源を発生または修正する。 指定された方向光源がすでに存在するならば修飾子に

よって指定された状態に修正し、利用可能にする。そうでなければ、指定された方向光源を作る。修飾子がなければ初期値が使われる。

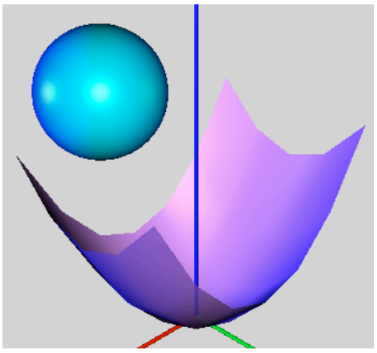
修飾子	値	効果
ambient	[R,G,B]	周囲の色を RGB 値で指定された色にする (初期値は [0,0,0])
diffuse	[R,G,B]	拡散する光の色を RGB 値で指定された色にする (初期値は [1,1,1])
specular	[R,G,B]	反射光の色を RGB 値で指定された色にする (初期値は [1,1,1])
direction	＜ vec ＞	光の方向 (初期値は [0,-1,0])
frame	＜ string ＞	方向がカメラフレームに依存するか、絶対的かを指定する。値は”camera” か ”world” で、初期値は ”camera”



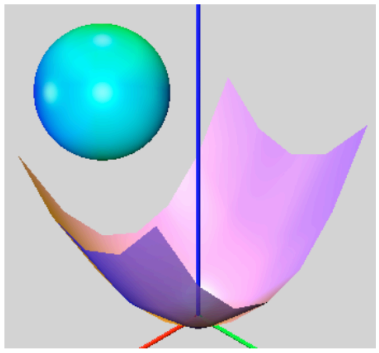
方向光源の指定なし



方向光源 1 `diffuse->[1,1,0]`



方向光源 2 `direction->[8,0,0],diffuse->[0,0,1])`



方向光源 1 と 2

スポットライトの設定：`spotlight3d(<int>)`

`<int>`は，光源の番号で 0 以上 7 以下の整数。

スポットライトを発生または修正する。 指定された光源がすでに存在するならば修飾子によって指定された状態に修正し、利用可能にする。そうでなければ、指定された光源を作る。修飾子がなければ初期値が使われる。

修飾子	値	効果
ambient	[R,G,B]	周囲の色を RGB 値で指定された色にする (初期値は [0,0,0])
diffuse	[R,G,B]	拡散する光の色を RGB 値で指定された色にする (初期値は [1,1,1])
specular	[R,G,B]	反射光の色を RGB 値で指定された色にする (初期値は [1,1,1])
position	＜ point ＞	点の位置 (初期値は [0,0,0])
direction	＜ vec ＞	光の方向 (初期値は [0,-1,0])
cutoffAngle	＜ real ＞	スポットコーンのカットオフ角。ラジアンで指定。 0 から $\frac{\pi}{2}$ 初期値は $\frac{\pi}{4}$
exponent	＜ real ＞	減衰指数。値は 0 以上 128 未満。初期値は 0
frame	＜ string ＞	方向がカメラフレームに依存するか、絶対位置かを指定する。値は”camera” か ”world” で、初期値は ”camera”

光源を無効にする：`disablelight3d(<int>)`

`<int>`は，光源の番号で 0 以上 7 以下の整数。

与えられた番号の光源を無効にする。