usSurfaceViewer (Ver. 1.18)
説明書

目的

usSurfaceViewerは、ZEMAXユーザ定義面DLLの確認用に作成したものです。
usSurfaceViewerはZEMAXユーザ定義面DLLを扱いますが、ZEMAXとは無関係に動作します。

ZEMAXには、ユーザ定義面DLLのデバッグ確認機能はありません。
例えば、ZEMAX光線追跡を確認する場合、確認したい位置に光線を通すのは困難です。
usSurfaceViewerは、マウス操作で光線入射位置を変えることができ、確認を容易にします。

機能概要

ZEMAXユーザ定義面DLLを読み込み、
(1) 関数 UserDefinedSurface (ZEMAXからコールされる唯一の関数) を呼び出し、
呼び出し前後の引き数を比較できます。
(2) 1光線について光線追跡を行います。
入射光線の通過位置および方向余弦をマウス操作で変更できます。
(3) 光線と面との交点での、サグ断面図を表示します。
また、入射出射軌跡に沿ったサグ断面図を表示します。
(4) 3Dモデルを生成し、シェーディング図を表示します。
この3Dモデルは、STL形式でファイル保存できます。
(5) ユーザ定義面DLLが 勾配屈折率 の場合、
カーソル x , y , z 各方向の n , dn/dx , dn/dx , dn/dx 変化を表示します。
(6) C言語を真似たプログラム言語でsag関数 z = f(x , y) を定義できます。
ユーザ定義面DLLが無くても、動作します。

注意制限事項(機能)

(1) 描画の分解能を変えることができますが、どの程度細かく描画できるかは環境に依存します。
(メモリ実装量に依存します)
(2) シェーディング図の描画は、OpenGLを使用しています。
OpenGLは、環境により描画結果に異常が発生することがあります。
(3Dモデルに存在しない線が描画されたり等)
グラフィックボードの設定を変えることで、描画異常を回避できることもあります。

注意制限事項(使用について)

usSurfaceViewerの使用は自由です。
usSurfaceViewerの配布は、無償配布であれば自由です。
無償ダウンロードでも、サイトアクセスが有償であったり、
他のソフトをダウンロードすることが前提である場合など、除きます。
メディア収蔵など、商用配布の場合は、事前通知ください。

usSurfaceViewerの使用については、何の補償もしません。
usSurfaceViewerの使用について、ZEMAX関係各社に問い合わせることは止めて下さい。

その他

バージョンアップなどのメンテナンスは、以下ホームページで行います。
http://homepage2.nifty.com/yees/

連絡先
yees@nifty.com

インストール　アンインストール

インストーラはありません。
usSurfaceViewerVxxx.zip を解凍してください。
フォルダusSurfaceViewerVxxxごと、適当な場所に置いてください。

レジスリは使用しておりません。 代わりに iniファイルを使用します。
usSurfaceViewer初回起動時に、usSurfaceViewer.ini を同じフォルダに生成します。
usSurfaceViewer.ini には、ウインドウ位置とサイズを記録しています。
usSurfaceViewer.ini は、いつでも削除可能です。

アンインストールは、フォルダusSurfaceViewerVxxxごと削除して下さい。

ファイル構成

usSurfaceViewer_w32.exe　：32bit用 実行ファイル
usSurfaceViewer_x64.exe　：64bit用 実行ファイル
usSurfaceViewer説明.pdf　：本ファイル

使用準備

（重要）
usSurfaceViewer で扱うユーザ定義面DLLは、
usSurfaceViewerと同じフォルダにコピーしてください。

usSurfaceViewerとは別のフォルダにあるDLLを開くには、
そのフォルダのパスをWindows環境変数PATHに設定しておく必要があります。

ZEMAXのユーザ定義面DLLの置かれているフォルダに、usSurfaceViewer_xxx.exeをコピーして
使用する方法も考えられますが、Program Filesフォルダは特別なフォルダですので推奨しません。

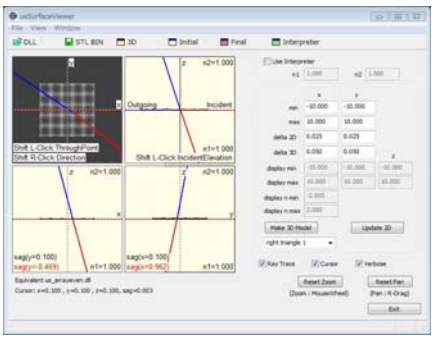
起動

使用環境に対応する実行ファイルを選んで起動して下さい。
usSurfaceViewer_w32.exe (32bit用)
usSurfaceViewer_x64.exe (64bit用)

Win7では、以下が必要かもしれません。
Microsoft Visual C++ 2008 再頒布可能パッケージ (x86)
Microsoft Visual C++ 2008 再頒布可能パッケージ (x64)

ウインド構成

メイン　ウインド



サグ断面図表示、
光線追跡操作、
描画設定
を行います。

サブ　ウインド

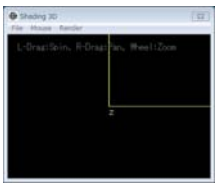
UD FD Initial

DLL 入力変数
を編集します。
また、
DLL関数の
呼び出し操作を行います。

UD FD Final

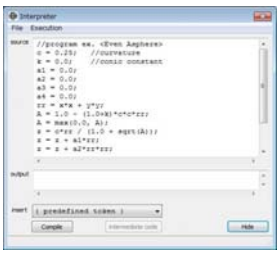
DLL 呼び出し結果を
表示します。

Shading 3D



3D シェーディング図を
表示します。

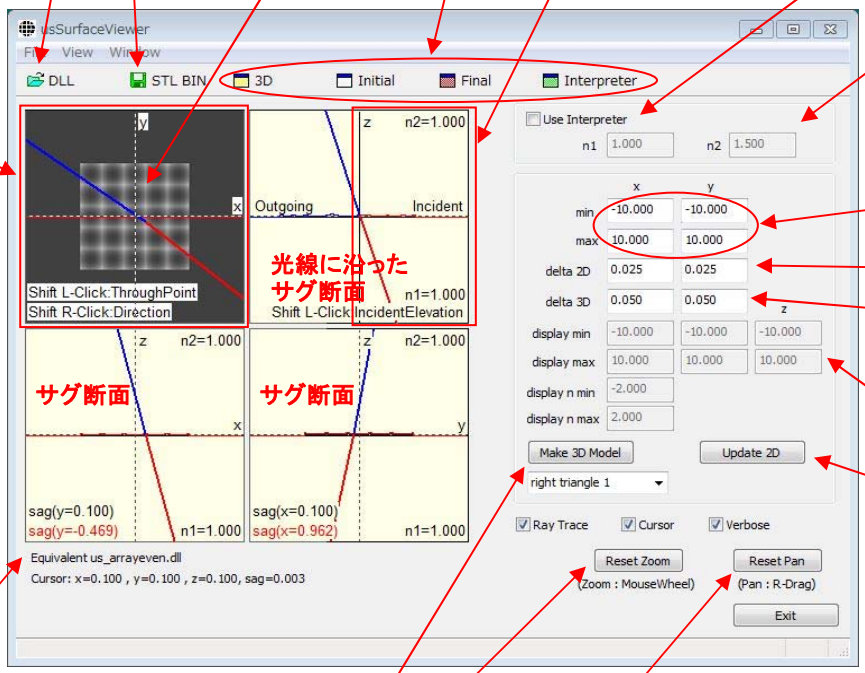
Interpreter



ユーザ定義面DLLを使用の
かわりに、
z = f(x, y) を定義します。

操作

usSurfaceViewer 起動直後、ユーザ定義面DLLの代わりに、
ZEMAXサンプル us_arrayeven.dll 相当の関数を使用します。
ユーザ定義面DLLを開いた場合、以降そのDLL内関数を使用します。



ユーザ定義面DLLを開く

面3Dモデルを STL(BIN)形式で保存
(Fileメニューから STL(ASCII)形式でも保存可能)

左クリック：
カーソル移動

閉じられたサブウインドを再表示

Shiftキー + 左クリック：光線仰角指定

ユーザ定義面DLLは
使用せず、
Interpreterウインドウの
プログラムを使用します。

プログラム使用使用時
光線追跡のn1 n2 を
指定します。

Shiftキー + 左クリック：
光線通過位置指定

光線通過位置は、
z=0平面上の(x, y)を指定し、
光線追跡結果の
面との交点とは一致しません

Shiftキー + 右クリック：
光線方向指定

光線に沿った
サグ断面

描画計算範囲

描画刻み（サグ断面）

描画刻み
（シェーディング図）

描画表示範囲
（サグ断面）

入力数値を反映し
描画更新（サグ断面）

光線追跡
入射光線：赤色
出射光線：青色
で表示

光線追跡計算
エラーの場合
破線表示

光線追跡時
DLL関数のリターン値（エラーコード）
を表示

表示拡大縮小
マウスホイール
でも可

表示移動
マウス右ボタン ドラッグ
でも可

Verbose
チェックを外したとき
表示を簡素にします

Cursor
チェックを外したとき
カーソルを非表示

3Dモデル生成し シェーディング図に表示

3Dﾓﾃﾞﾙは、以下選択可能です。
- right triangle 1　（直角三角形 1）
- right triangle 2　（直角三角形 2）(1との違いは、三角形の向き)
- equilateral triangle 1　（正三角形 1）
- equilateral triangle 2　（正三角形 2）(1との違いは、三角形の向き)

xy平面(z=0)を分割する三角形を選択します。
各三角形の頂点のサグ(z値)を求め、3Dモデルを生成します。

Shading3Dウインドで、ワイヤーフレーム表示すれば確認できます。
(メニューRender の Line を選択)

例のような矩形レンズアレイであれば、
right triangle 1　または　right triangle 2 が適当です。
レンズ面形状に応じ、試してください。

(注) ZEMAXのシェーディング描画が
どのような3Dモデルを使っているかは不明です。

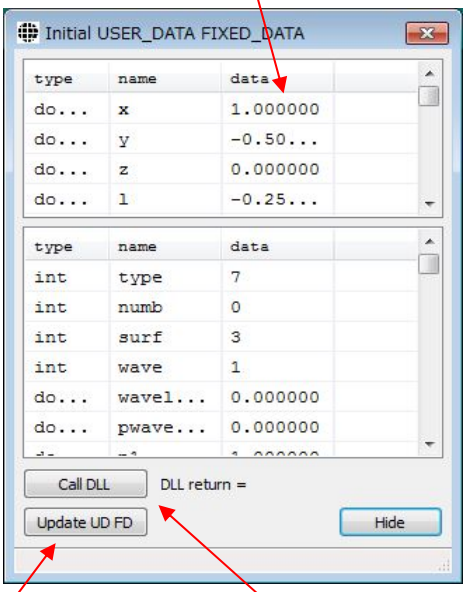
DLLを読み込んだときの、FIXED_DATAバージョン判定

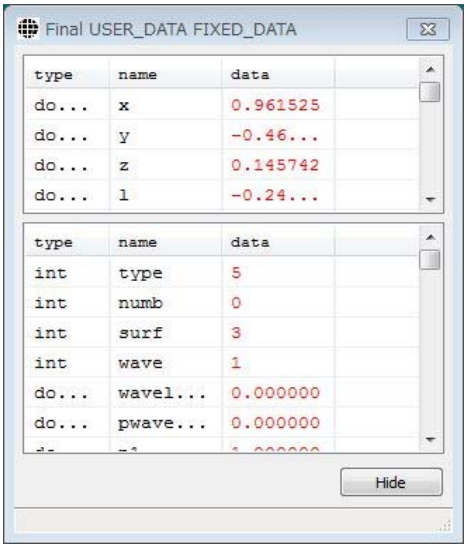
①DLL内に関数名UserDefinedSurfaceが存在すれば
UserDefinedSurface(USER_DATA *UD, FIXED_DATA *FD)
として、面定義関数とする。
↓見つからなければ
②DLL内に関数名UserDefinedSurface2が存在すれば
UserDefinedSurface2(USER_DATA *UD, FIXED_DATA *FD2)
として、面定義関数とする。
↓見つからなければ
③DLL内に関数名UserDefinedSurface3が存在すれば
UserDefinedSurface3(USER_DATA *UD, FIXED_DATA *FD3)
として、面定義関数とする。
↓見つからなければ
④DLL内に関数名UserDefinedSurface4が存在すれば
UserDefinedSurface4(USER_DATA *UD, FIXED_DATA *FD4)
として、面定義関数とする。
↓見つからなければ
⑤DLL内の1番目の関数を
UserDefinedSurface(USER_DATA *UD, FIXED_DATA *FD)
として、面定義関数とする。

操作（ DLL関数の数値確認 ）

UD FD Initial ウインド および UD FD Final ウインド は、
usersurf.h 内で定義されている構造体
　USER_DATA
　FIXED_DATA
を表示するものです。FIXED_DATAについては、
　FIXED_DATA
　FIXED_DATA2
　FIXED_DATA3
　FIXED_DATA4
に対応しております。

各 deta は、
1回クリックで選択し、
もう一度クリックで編集可能となります。





Update UD FD は、
data を変更したとき、
メインウインドの描画に反映させます。
(メインウインドの Update2D ボタンと同機能です)

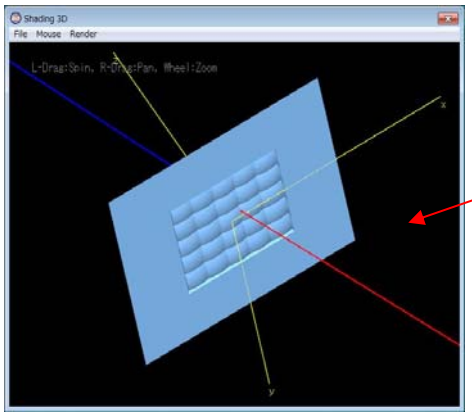
Call DLL は、
DLL関数を呼び出し、結果を Final ウインドに反映します。

DLL関数の引数は、
USER_DATA構造体、FIXED_DATA構造体 各ポインタです。
DLL関数の実行結果は、構造体要素変数を書き換え、返されます。

Initial ウインド内容を、
USER_DATA構造体、FIXED_DATA構造体にコピーし、
DLL関数を呼び出し後、
USER_DATA構造体、FIXED_DATA構造体内容を
Final ウインドに表示します。

(注)
FIXED_DATA の n1 , n2 は、
それぞれ、
　光線追跡 入射光側の屈折率
　光線追跡 出射光側の屈折率
です。
ZEMAXの光線追跡は、面の配置により自動的に
屈折率が設定されるので、
ZEMAXと比較確認する際、注意が必要です。

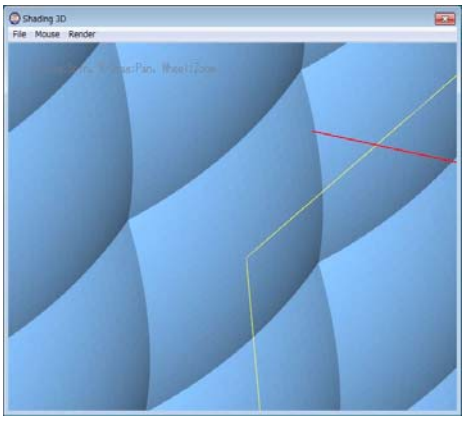
Shading 3D ウインドウ
メインウインドウの「Make 3D Model」ボタンで作成した3Dモデルを表示します。



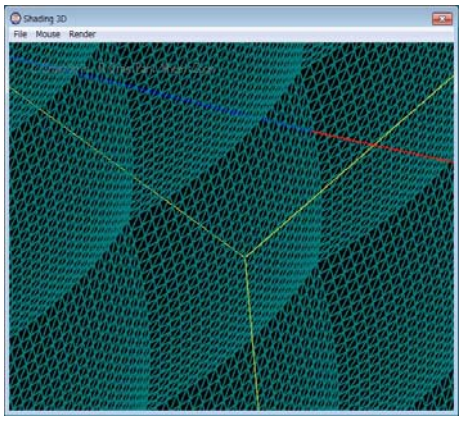
マウス 左ボタン＋ドラッグ で回転
マウス 右ボタン＋ドラッグ で移動
マウス ホイール で拡大/縮小
できます。

File メニューから
既存のSTL(BIN, ASCII)ファイルを開き、表示可能です。
ファイル先頭から256byteに、文字列 solid が存在すれば、
ASCII形式と判断

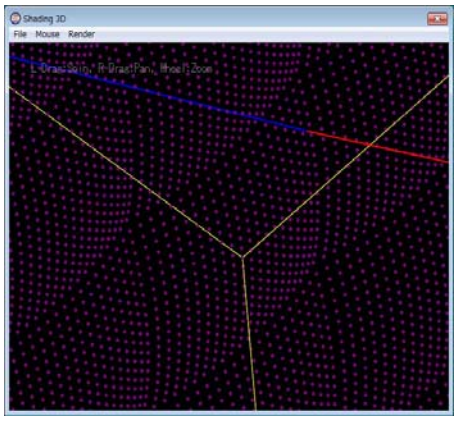
Render メニューから
レンダリング対象を選択できます。



Face のみ選択した場合。



Line のみ選択した場合。



Point のみ選択した場合。

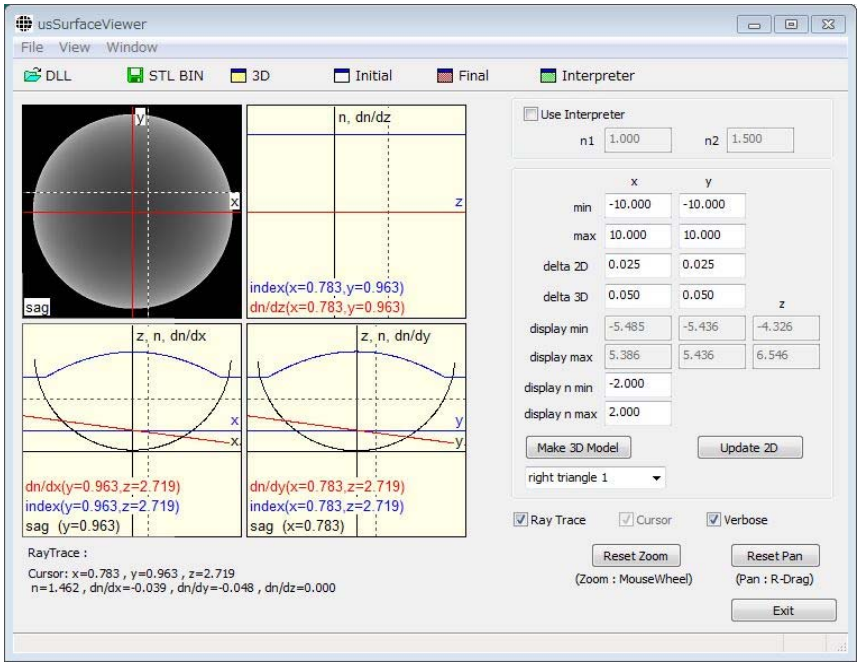
ユーザ定義DLLが勾配屈折率の場合

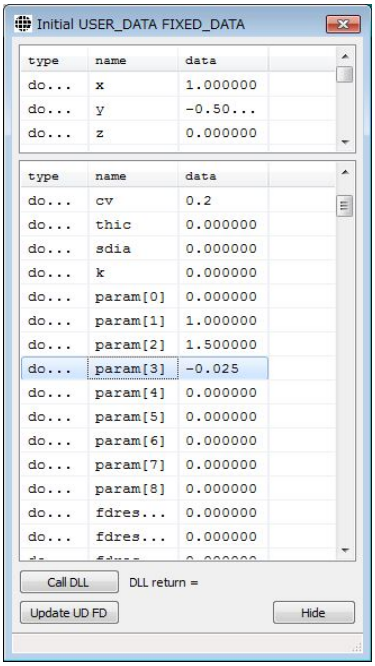
DLLは、与えられた(x, y, z)点の
- n
- dn/dx
- dn/dy
- dn/dz

を返すのみで、媒質内の光線追跡計算は行わない。

usSurfaceViewer は、カーソル x , y , z 各方向の n , dn/dx , dn/dx , dn/dx 変化を表示する。

（例） us_grin1.dll





us_grin1.dll の場合
- n = N + A･(x･x + y･y)
- dn/dx = 2.0･A･x
- dn/dy = 2.0･A･y
- dn/dz = 0

であり、
この例では、
- N : param[2] = 1.5
- A : param[3] = −0.025

を設定している。

また、
- cv = 0.2
- k = 0

を設定している。

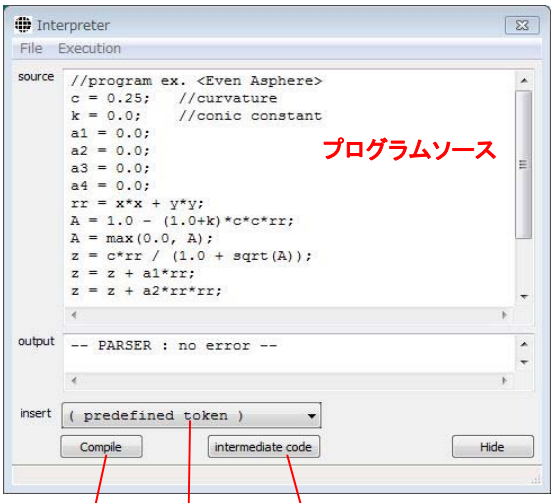
Interpreterウインドウ

ユーザ定義面DLLを使用せず、
z = f (x , y) 関数をプログラムします。

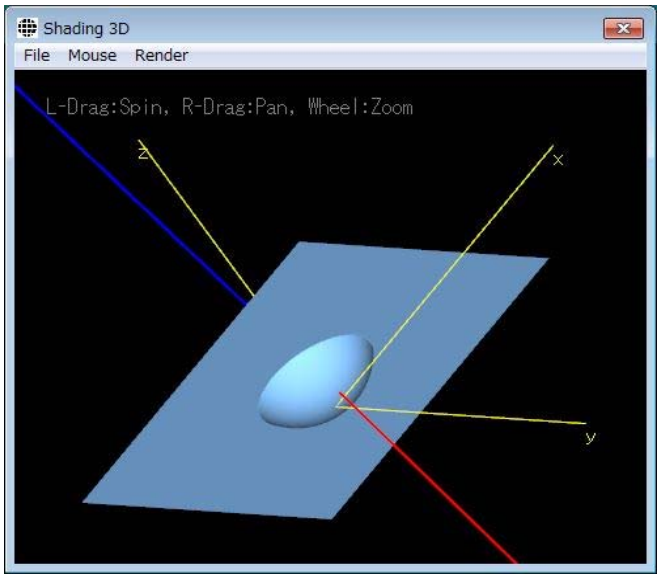
操作手順
①プログラムソースを編集
②「Compile」ボタンにより、中間コードを生成
③メインウインドウの「Use Interpreter」にチェックを入れます。
④以降、sag計算、光線追跡は中間コードを使用します。

中間コードは、x , y が与えられたときに、z(x, y) を算出するためのコードです。
usSurfaceViewer内の仮想マシンで使用します。

usSurfaceViewer起動時、
あらかじめ、ソース例が入力されます



プログラムソース



中間コードを表示します。
（デバッグ用）

予約済みの字句一覧
選択した字句を、ソースに挿入します。

中間コードを生成します。
エラーがあれば、output に表示します。

プログラム使用時の光線追跡について
多様な z(x , y) に対応するため、以下の手順で計算しています。

手順
①光線と面の交点を計算
ニュートン法でなく
2分法を採用しています。 収束判定は、1.0e-3 (交点座標精度)
↓
②交点の面法線ベクトルの計算
交点と、面上の交点近傍の2点をもとめ、法線ベクトルを計算しています。
近傍2点は、交点座標を (xc, yc, zc) とすると、
(xc+dx , yc , z((xc+dx), yc))
(xc , yc+dy , z(xc, (yc+dy)))
の2点です。
dx および dy は、メインウインドウの「delta 2D」で設定した値／10 です。
↓
③スネル法則適用し出射方向余弦を計算

**** 注意 ****

スネル法則適用は、
光線が最初に面と交差した1点のみです。

（ZEMAXシーケンシャル動作と同じです）

プログラムについて

(1) 全般

文法は、C言語と似たものとしてありますが、
if, while goto 等使用できません、また、関数定義もできません。
数式を記述するのみです。

変数名 = 数式 ;

;（セミコロン）が1行の終端です。

を複数記述します。

(6)コメント

// から行末まで、コメント文字列とします。
(C言語コメント /* ..... */ は不可です)

(2) 変数名

英数字が使用可能です。 ただし
x y z （小文字） は予約済みです。

x y は、プログラム外部から与えられるものとし、代入不可です。
最終的に計算される z を sag とします。

(3) 数値

数値は、倍精度数値に変換します。
pi または PI は円周率として扱います。

(4) 演算子

+ 加算
- 減算 （またはマイナス符号）
*****
/
== (左辺＝右辺)時(1.0)を返す、不成立時(0.0)を返す。
!= (左辺≠右辺)時(1.0)を返す、不成立時(0.0)を返す。
> (左辺＞右辺)時(1.0)を返す、不成立時(0.0)を返す。
>= (左辺≧右辺)時(1.0)を返す、不成立時(0.0)を返す。
< (左辺＜右辺)時(1.0)を返す、不成立時(0.0)を返す。
<= (左辺≦右辺)時(1.0)を返す、不成立時(0.0)を返す。
&& (左辺≠0.0)かつ(左辺≠0.0)時(1.0)を返す、不成立時(0.0)を返す。
|| (左辺≠0.0)または(左辺≠0.0)時(1.0)を返す、不成立時(0.0)を返す。

比較演算、論理演算、は浮動小数点で扱います。

比較演算結果は、成立時 (1.0) を返すので、
以下のような扱いが可能です。
x1 = (x>=-2.0) * (x<=2.0) *x;
は、x = -2.0～2.0 の範囲で、x1=x
範囲外で、x1=0.0 となります。

論理演算は、
(0.0) 以外 : true
(0.0) : false
とします。

(4) if ～ else

C言語同様形式、
if(a) . . . . ;
if(a) . . . . ; else . . . . ;
if(a) { . . . . ; . . . . ; . . . . ; }
if(a) { . . . . ; . . . . ; . . . . ; } else { . . . . ; . . . . ; . . . . ; }

但し、条件判定式は浮動小数点演算を行い、(a≠0.0)のとき if 以下を実行する。

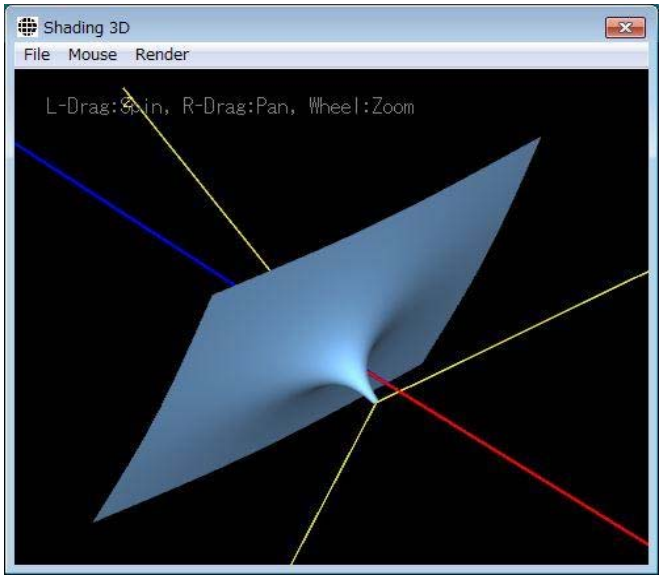
(5)算術関数 (一部C言語とは異なります)

```
sqrt( a )          computes square root
pow( a , b )       raises a number to the given power
exp( a )           returns e raised to the given power
ln( a )            computes natural (base e) logarithm
log( a )           computes natural (base e) logarithm
log10( a )         computes common (base 10) logarithm
max( a , b )       larger of two values
min( a , b )       smaller of two values
abs( a )           computes absolute value
sin( a )           computes sine
cos( a )           computes cosine
tan( a )           computes tangent
asin( a )          computes arc sine
acos( a )          computes arc cosine
atan( a )          computes arc tangent
sinh( a )          computes hyperbolic sine
cosh( a )          computes hyperbolic cosine
tanh( a )          computes hyperbolic tangent
```

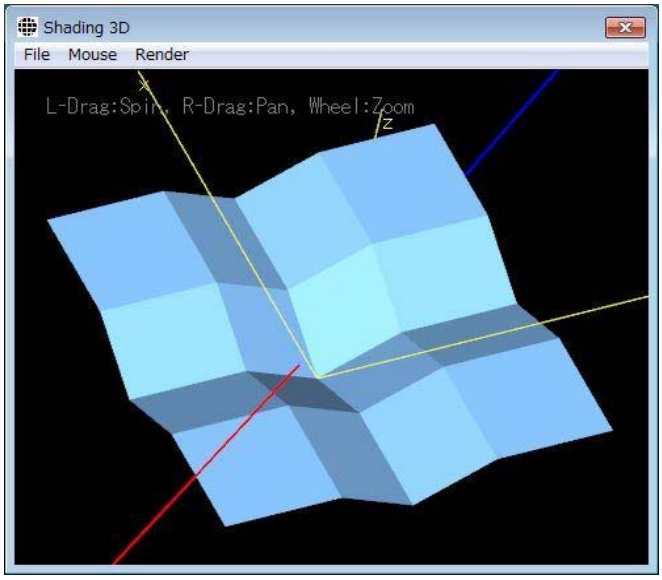
プログラム例

（とうてい レンズ面とは . . . ）

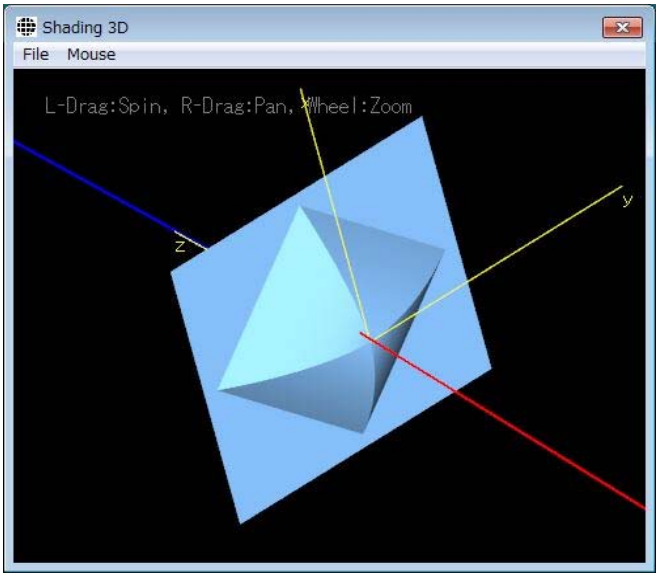
```
r = sqrt(x*x+y*y);
a = r / 2;
z = log(a) + 2.7;
z = max(0.0, z);
```



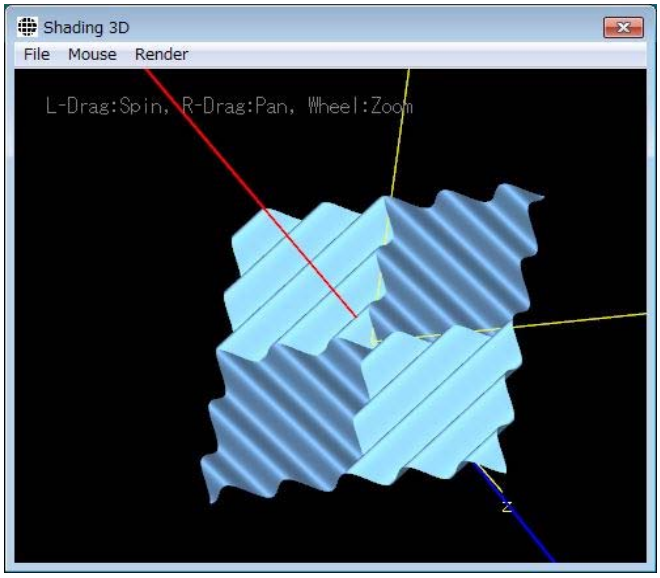

```
x1 = min(2, abs(x/2));
y1 = min(2, abs(y/2));
z = x1 + y1;
z = min(4, z);
```



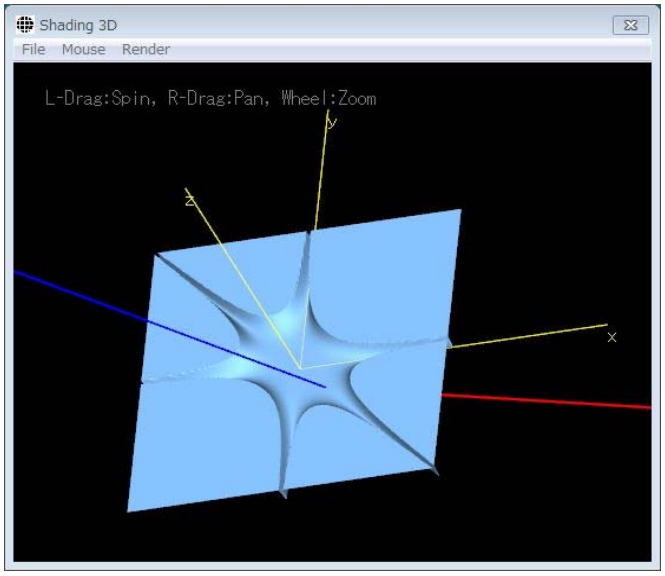

```
x1 = abs(x/4);
y1 = abs(y/4);
z = x1 + y1;
z = min(4, pow(z, 1.7));
```



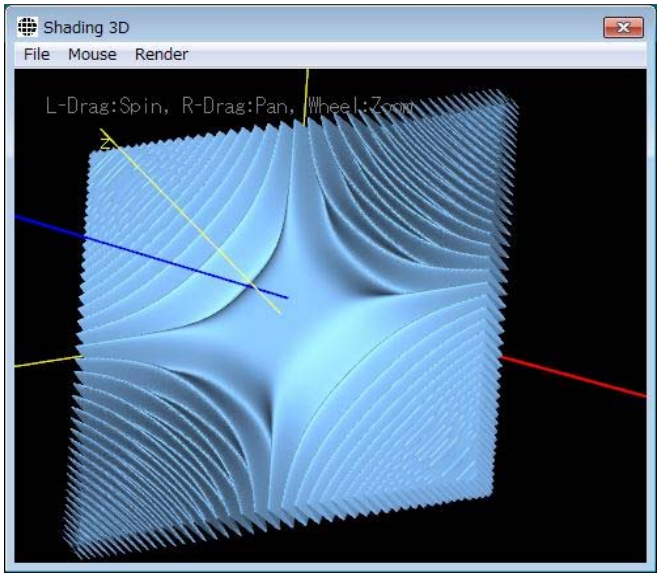
```
x1 = abs(x/4);
y1 = abs(y/4);
z = x1 + y1;
z = min(4, cos(z*pi*2));
```




```
r = abs(x+y);
a = r / 2 * pi * x/4 * y/4;
z = cosh(a);
z = min(2, z);
z = max(-2, z);
z = z - 1.0;
```




```
r = sqrt(x*x+y*y);
a = r / 2 * pi * x/4 * y/4;
z = cos(a);
```



```
//program ex. <array Asphere>
c = 0.25; //curvature
k = 0.0; //conic constant
//-----------------------------------------
W = 4.0;
H = 6.0;
hW = W/2;
hH = H/2;
//-----------------------------------------
x1 = x;  y1 = y;
s1 = (x1>-hW)*(x1<hW)*(y1>-hH)*(y1<hH);
rr1 = x1*x1 + y1*y1;
a = 1.0 - (1.0+k)*c*c*rr1;
a = max(a, 0.0);
z1 = c*rr1 / (1.0 + sqrt(a));
z1 = z1 * s1;
//-----------------------------------------
x2 = x-W;  y2 = y;
s2 = (x2>-hW)*(x2<hW)*(y2>-hH)*(y2<hH);
rr2 = x2*x2 + y2*y2;
a = 1.0 - (1.0+k)*c*c*rr2;
a = max(a, 0.0);
z2 = c*rr2 / (1.0 + sqrt(a));
z2 = z2 * s2;
//-----------------------------------------
x3 = x+W;  y3 = y;
s3 = (x3>-hW)*(x3<hW)*(y3>-hH)*(y3<hH);
rr3 = x3*x3 + y3*y3;
a = 1.0 - (1.0+k)*c*c*rr3;
a = max(a, 0.0);
z3 = c*rr3 / (1.0 + sqrt(a));
z3 = z3 * s3;
//-----------------------------------------
z = z1 + z2 + z3;
z = z + 2.0 * !(s1 || s2 || s3);
z = min(z, 2.0);
```