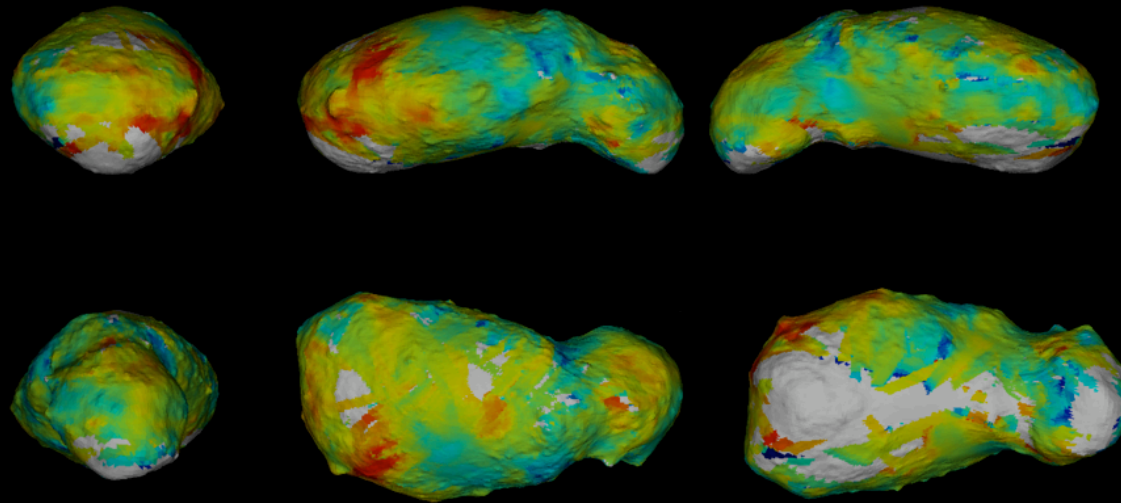
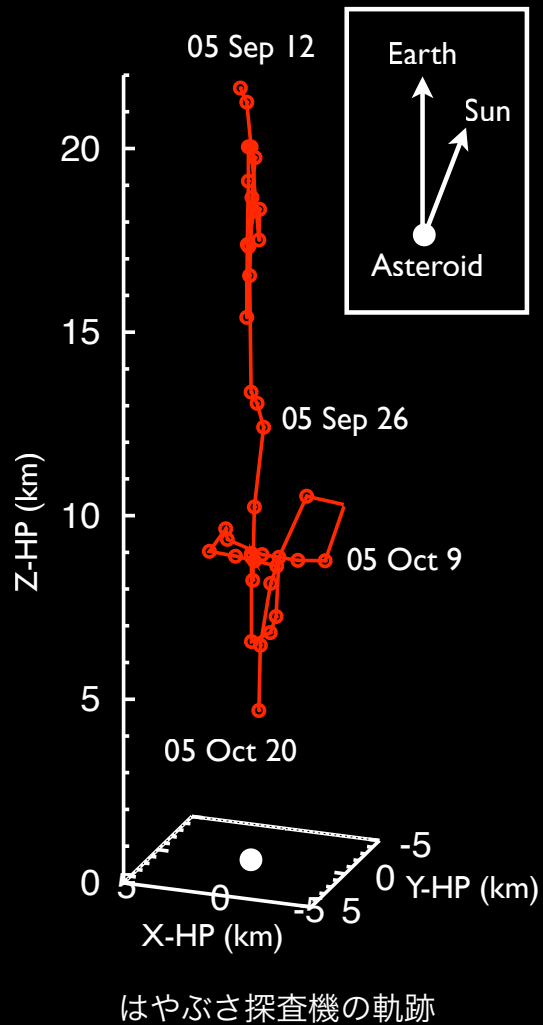


第一回 惑星探査データ解析実習会
小惑星探査機はやぶさのデータ解析



NIRS : 分光データ処理 イトカワ編
北里 宏平 (神戸大学)

NIRS : イトカワ観測



NIRS 観測期間

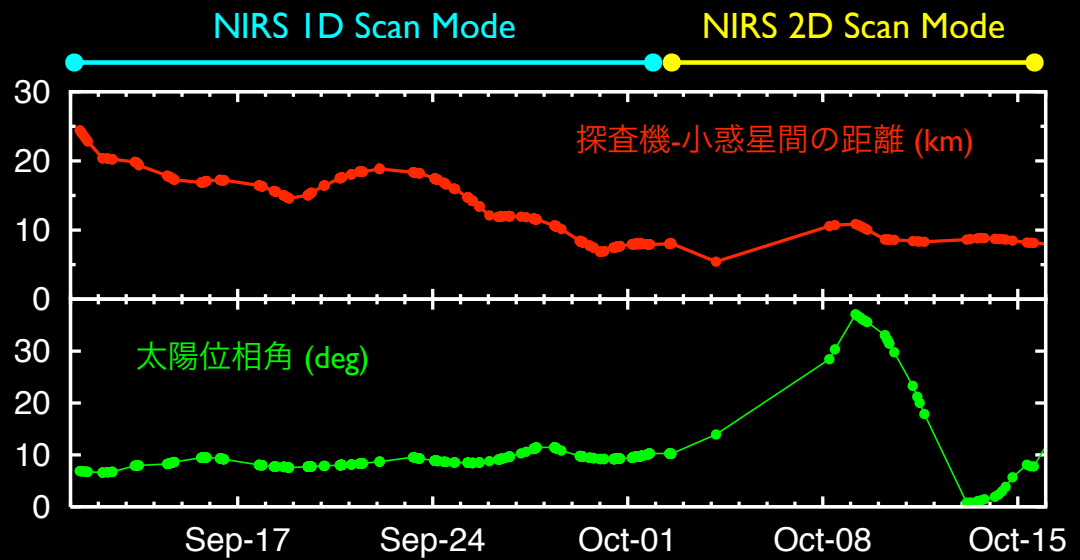
2005年 9月10日～11月24日 (56日間)

総スペクトル数 : ~80,000 本

観測パラメタ

積分時間 : 0.82 ~ 26.21 sec

検出器温度 : 一定 (~258 K)



NIRS : 観測の可視化

Spacecraft View

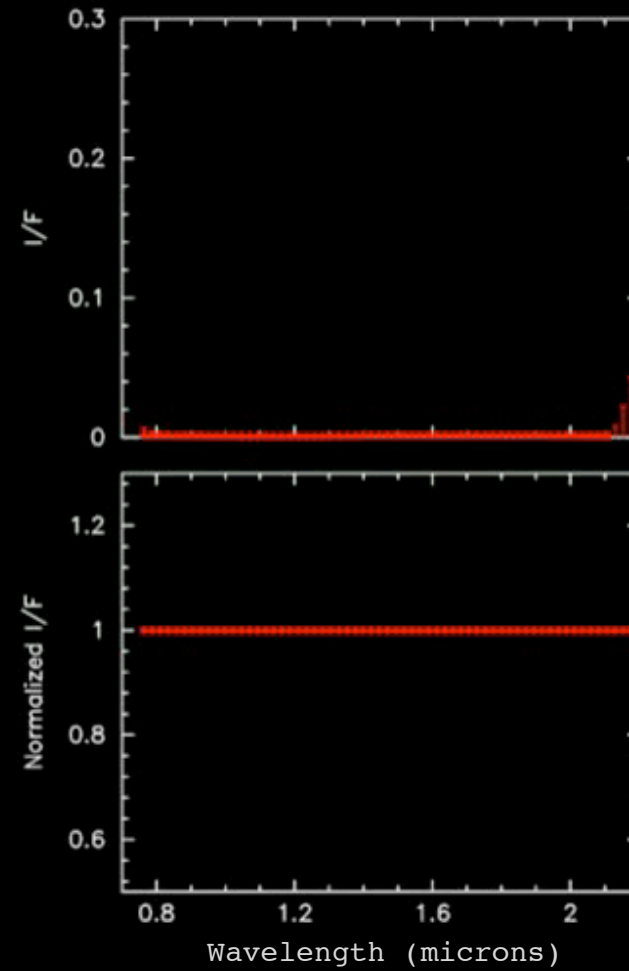
Time :2380634781:194 (s/c clock)
2005-09-16 00:00:31 (utc)



Lighting condition :

i = --- e = --- g = ---

NIRS spectrum

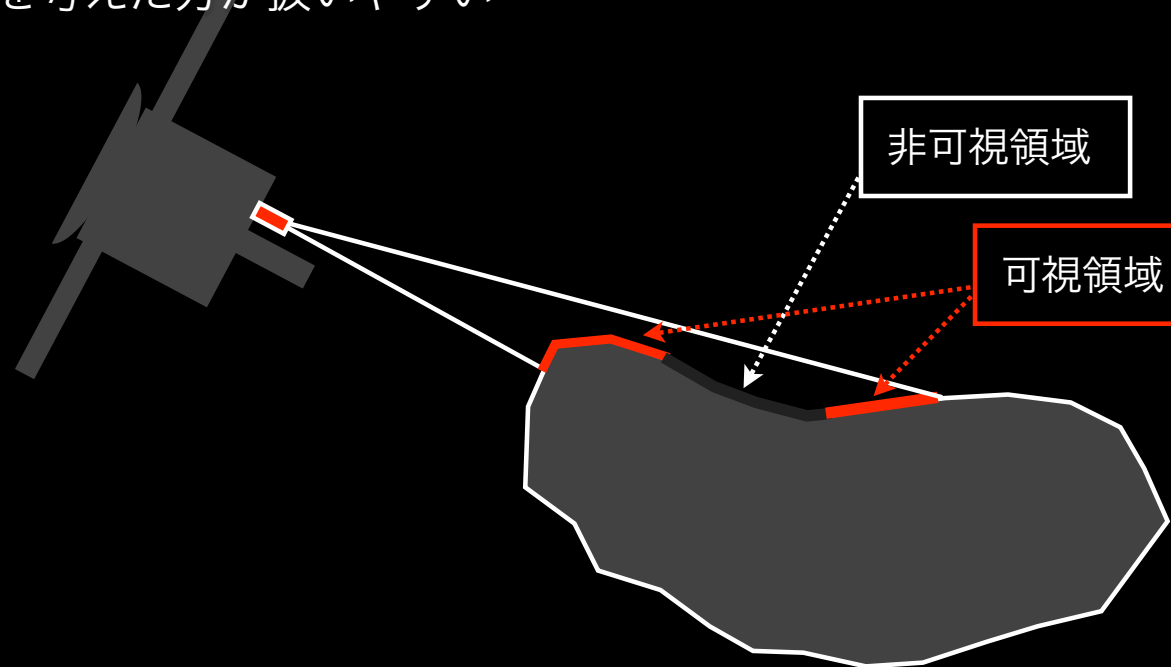


反射スペクトルのマッピング

不規則形状天体のフットプリントの扱い

(視野に対して地形の起伏のスケールが大きい場合)

- フットプリント領域の分割が起こり得るので 経緯度扱いは困難
 - 視野中心の日照条件が全体を代表しているとは限らない
- ▶ フットプリントの輪郭より 視野内にみえるプレート
を考えた方が扱いやすい



視野内に含まれるプレートの抽出

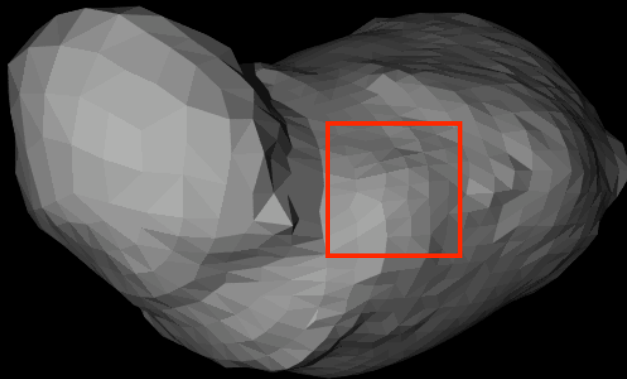
viewgeo.c

時間 (ephemeris time) を指定して NIRS の視野内に含まれるイトカワモデルのプレートとそれらのプレート毎の日照条件 (入射角, 出射角, 位相角) を計算する

— 使用している関数

platelib.a: PNTOCC (Plate occlusion processor)

視野内に含まれるプレートを抜き出す関数



視野に含まれているプレートが
わかったら, それらのプレートに
観測データの値も付与する

イトカワのスペクトルマップ作成

イトカワ観測データのマッピングの流れ

1. NIRS FITS から反射スペクトルに変換
(mkspc.pl を使って一括処理)
2. フットプリント・日照条件を計算
(mkplt.pl を使って一括処理)
3. 集計・データベース作成 (plinteg.pl)

イトカワのスペクトルマップ作成に必要なデータ

NIRS : イトカワ観測データ (20050916/*fits, 1287本)

SPICE : generic_kernels (LSK, PCK, SPK)

HAYABUSA (FK, SCLK, PCK, IK=nirs11.ti,

SPK=hayabusa_itokawarendezvous_v01.bsp)

Itokawa (PCK, SPK, PLATE)

NIRS FITS から反射スペクトルに変換

mkspc.pl の編集

% vi Tansaku_kitazato_v0308/bin/mkspc.pl

```
#!/usr/bin/env perl
use Math::Trig;
```

```
# set the absolute paths
```

```
$PATH_SPICE_KERNEL = "/home/xxx/kernels";
```

```
$PATH_NIRS_BIN      = "/home/xxx/bin";
```

```
$PATH_NIRS_CAL      = "/home/xxx/nirs_calib";
```

```
$target = itokawa; # target name or id number
```

```
...
```

一括変換処理

% cd Tansaku_kitazato_v0308/itokawa

% tar xvfz 20050916.tar.gz

% cd 20050916

% ../../bin/mkspc.pl *.fits

```
2380635849_lv11.0.fits -> 2380635849.spc
```

```
2380637996_lv11.0.fits -> 2380637996.spc
```

```
2380640144_lv11.0.fits -> 2380640144.spc
```

```
2380642291_lv11.0.fits -> 2380642291.spc
```

```
...
```

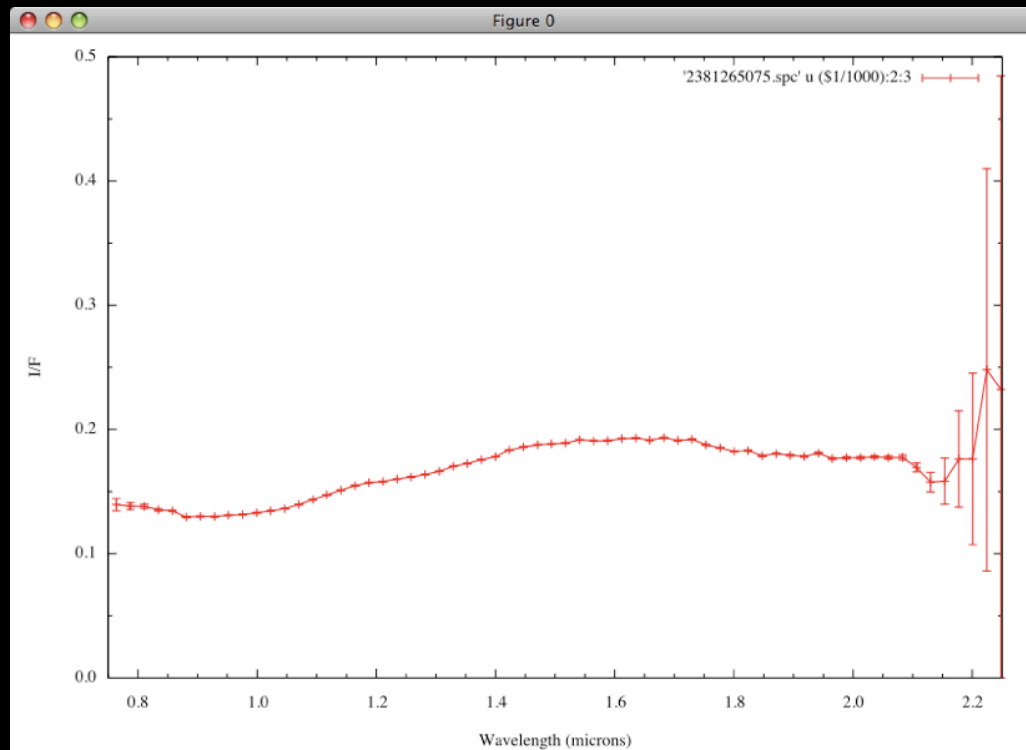
反射スペクトルデータの確認

```
# gnuplotで簡単なアニメーションを作ってみる
```

```
% mkgp.pl # spc.gp が作られる
```

```
% gnuplot
```

```
gnuplot> load "spc.gp"
```



フットプリント・日照条件を計算

mkplt.pl の編集

```
% vi Tansaku_kitazato_v0308/bin/mkplt.pl
```

```
#!/usr/bin/env perl
```

```
# set the absolute paths
```

```
$PATH_SPICE_KERNEL = "/home/xxx/kernels";
```

```
$PATH_NIRS_BIN      = "/home/xxx/bin";
```

```
...
```

一括処理

```
% cd itokawa/20050916
```

```
% ../../bin/mkplt.pl *.fits
```

```
2380635849_lv11.0.fits -> 2380635849.plt
```

```
2380637996_lv11.0.fits -> 2380637996.plt
```

```
2380640144_lv11.0.fits -> 2380640144.plt
```

```
2380642291_lv11.0.fits -> 2380642291.plt
```

```
...
```


視野内に含まれるプレート 1 枚毎の
ID, 投影面積, 入射角, 出射角, 位相角
が書かれている

集計・データベース作成

```
# プレート毎にスペクトル情報をコンパイル  
# (視野面積に対する投影面積の割合で平均化)
```

```
% cd itokawa/20050916
```

```
% plinteg.pl # 集計結果が plt.dat に出力される
```



形状モデルのプレート 1 枚毎の
ID, 観測回数, 各ピクセルの平均反射率
が書かれている

スペクトルマップの可視化

POV-Ray (Persistence of Vision Raytracer)

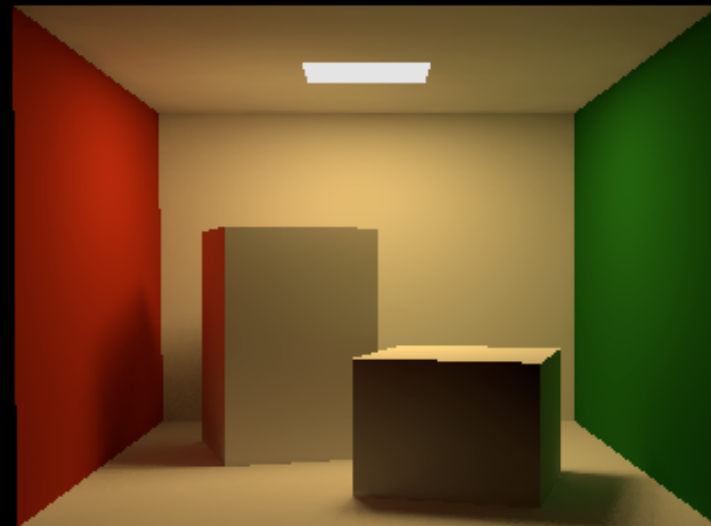
光源から放射された光の軌跡をシミュレートすることによりシーン画像を作成する本格的なレイトレーシングソフトウェア

- オープンソース
- マルチプラットフォーム対応
- 複雑な設定なしにレイトレースしてくれる

モデラーではなくレンダラー
簡単な言語で記述・実行

最低限必要な設定

- camera (カメラ)
- light_source (光源)
- object (物体)



POV-Ray demo

POV-Ray サンプル

- テキストエディタで新規ファイルを開いて下記の内容を記述
- ファイル名を `sample.pov` にして保存

```
camera{  
    location <0,0,-3>  
    look_at <0,0,0>  
    angle 90  
}  
  
light_source {  
    <100,100,-100>  
    color rgb <1,1,1>  
}  
  
object {  
    sphere {<0,0,0>, 1}  
    pigment {color rgb <1,0,0>}  
}
```

カメラの設定

視点の位置 (xyz)

注視点の位置 (xyz)

カメラ視野角

光源の設定

光源の位置 (xyz)

色の設定 (rgb)

オブジェクトの設定

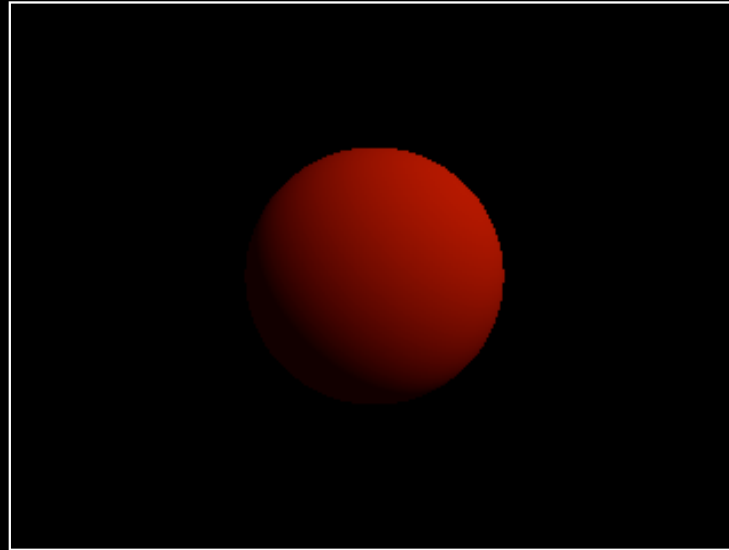
球の中心位置 (xyz), 球の直径

表面の色の設定 (rgb)

POV-Ray 実行

```
% povray +Isample.pov -D
```

```
# Povray for Windows はシーンファイルを開いて RUN
```



プレートモデルのレンダリング

itokawa_temp.pov の中身

```
#include "itokawa_temp.inc"

camera {
    perspective
    location < 0, -10000, 0 >
    right < -1.33, 0.0, 0.0 >
    up < 0.0, 1.0, 0.0 >
    sky < 0.0, 0.0, 1.0 >
    look_at < 0.0, 0.0, 0.0 >
    angle 4.0
}

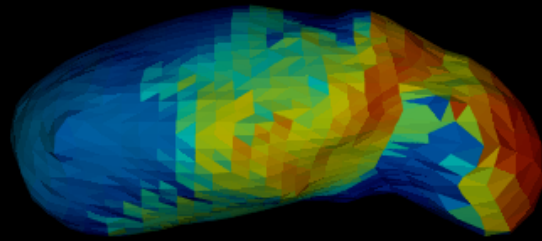
light_source {
    < 0, -10000, 0 >
    color rgb < 1.0, 1.0, 1.0 >
    parallel
    point_at < 0.0, 0.0, 0.0 >
}
```

インクルードファイル (データ挿入)
プレートモデルのオブジェクト設定

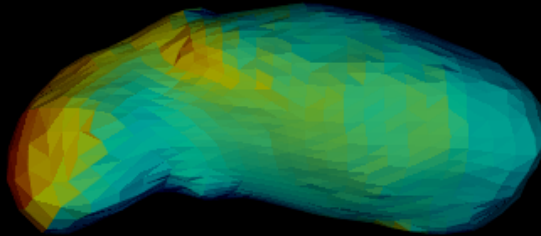
```
polygon{
    4, <x1,y1,z1>,<x2,y2,z2>,
        <x3,y3,z3>,<x1,y1,z1>
    pigment{color rgb<r,g,b>}
}
```

別の方向からの絵を作るには
カメラと光源の位置をかえる

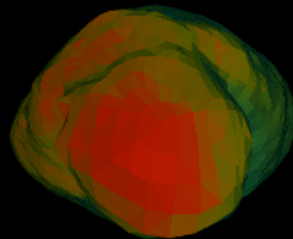
いろんな方向からレンダリング



-Y



+Y



+X

POV-Ray inc ファイルの作成

plt.dat にあるプレートの値（観測回数およびピクセルの反射率）で
色付けして povray のインクルードファイルを作成

```
% vi Tansaku_kitazato_v0308/bin/mkpovinc.pl
```

```
#!/usr/bin/env perl

# set the absolute paths
$PATH_SPICE_KERNEL = "/home/xxx/kernels";

$pix    = 0; # pixel number of nirs for drawing
$max    = 30; # maximum value of data range
$min    = 1; # minimum value of data range
...
$pix += 2;
```

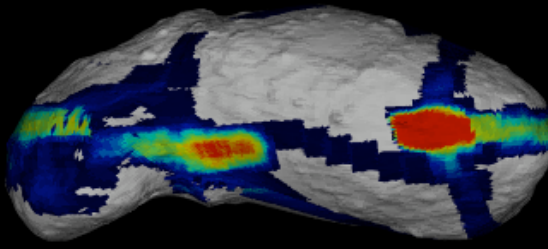
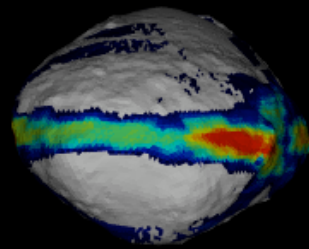
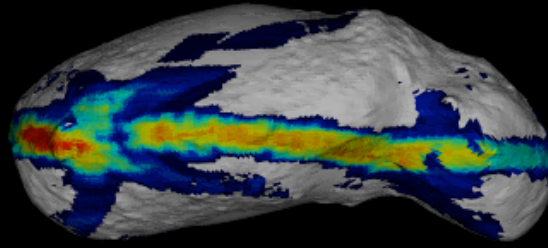
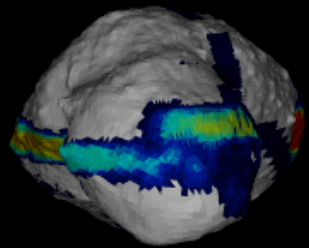
```
% cd itokawa/20050916
% ../../bin/mkpovinc.pl
```

plt.inc が作られる

POV-Ray 実行

```
% cp ../../povray/itokawa_temp.pov . # need to be edited
% cp ../../povray/makefile.pov . # need to be edited

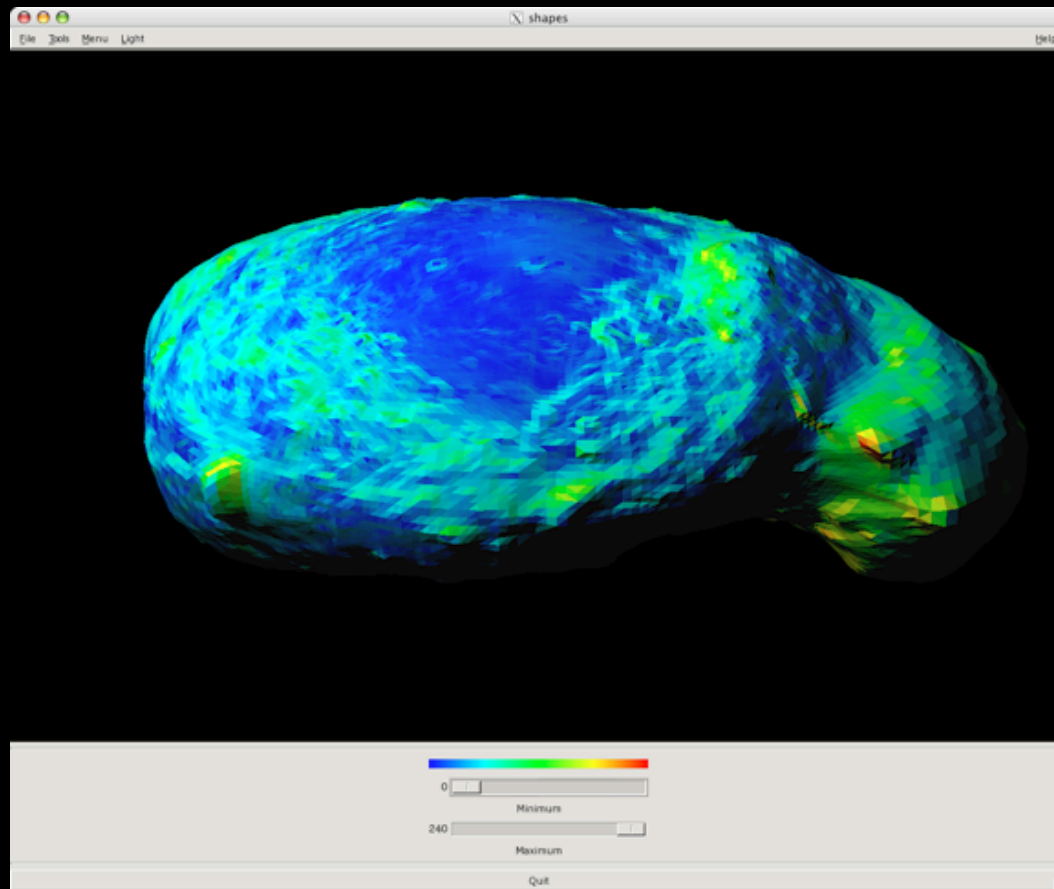
# itokawa_temp.pov を povray で実行
% make -f makefile.pov
# シーン画像ができる
```



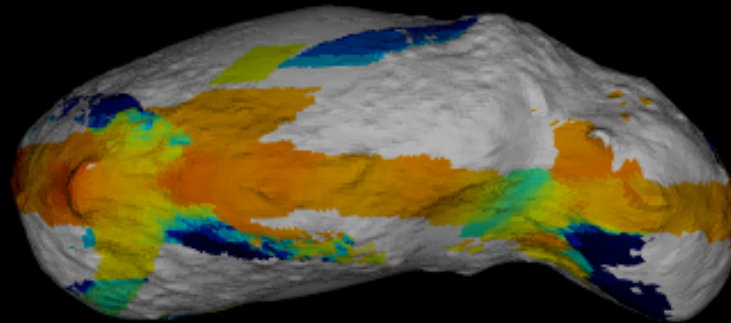
1 30
観測回数

Aizu 3D-GIS

スペクトルマップデータを Aizu 3D-GIS で表示してみる



反射率でレンダリングすると



0.08 0.22



30-ch の反射率



NIRS : 解析フローチャート

