

可視化の基礎演習

担当教員 大野

概要

- 3次元ベクトルデータの可視化
 - OpenDXの起動
 - Generalファイル作成
 - Visualization Networkの例
 - 流線(Streamline, Tube), 矢印
 - フォルダはdx_11th。デスクトップにコピーすること。
MyDocumentにはコピーしないこと

参考

Mike Bailey's OpenDX Page: <http://web.engr.oregonstate.edu/~mjb/opendx/>

D. Thompson, J. Braun, and R. Ford, "OpenDX: Paths to Visualization", Vis Inc., 2001.

OpenDXの起動 -1

X serverが必要

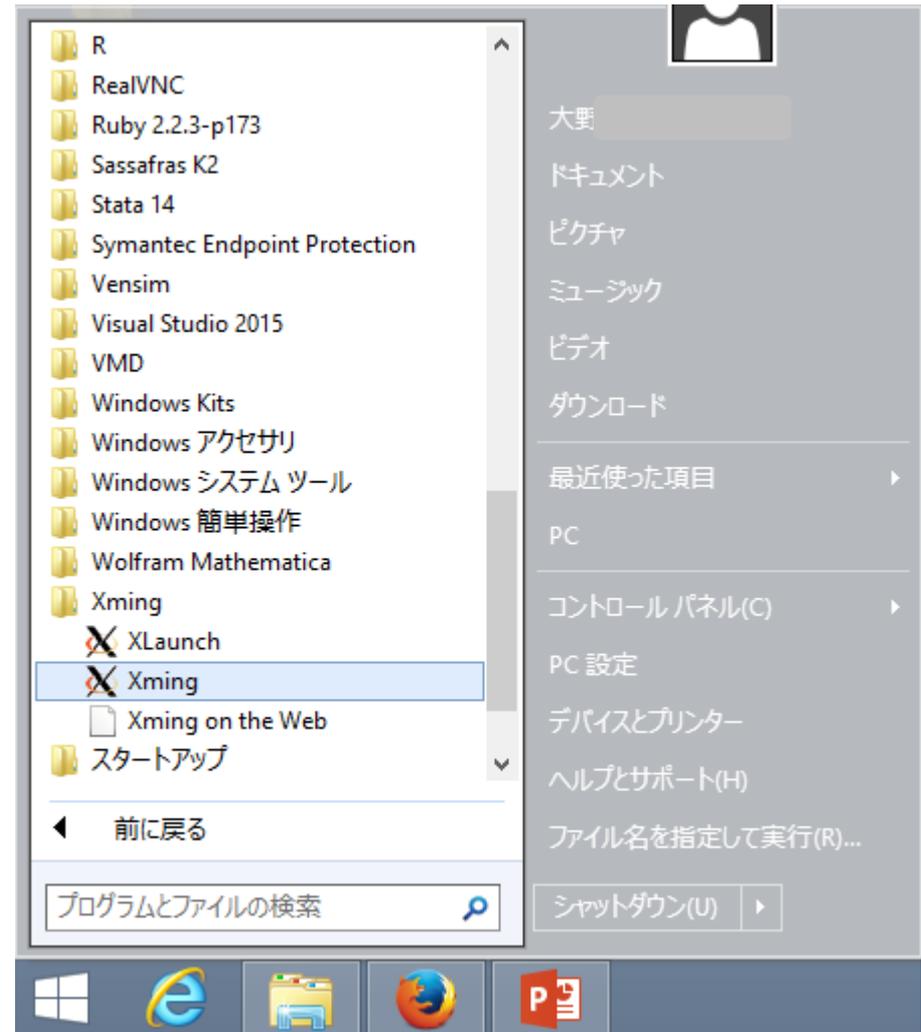
スタートボタン

→すべてのプログラム

→Xming

→Xming

でXサーバを起動



OpenDXの起動 -2

X server起動後

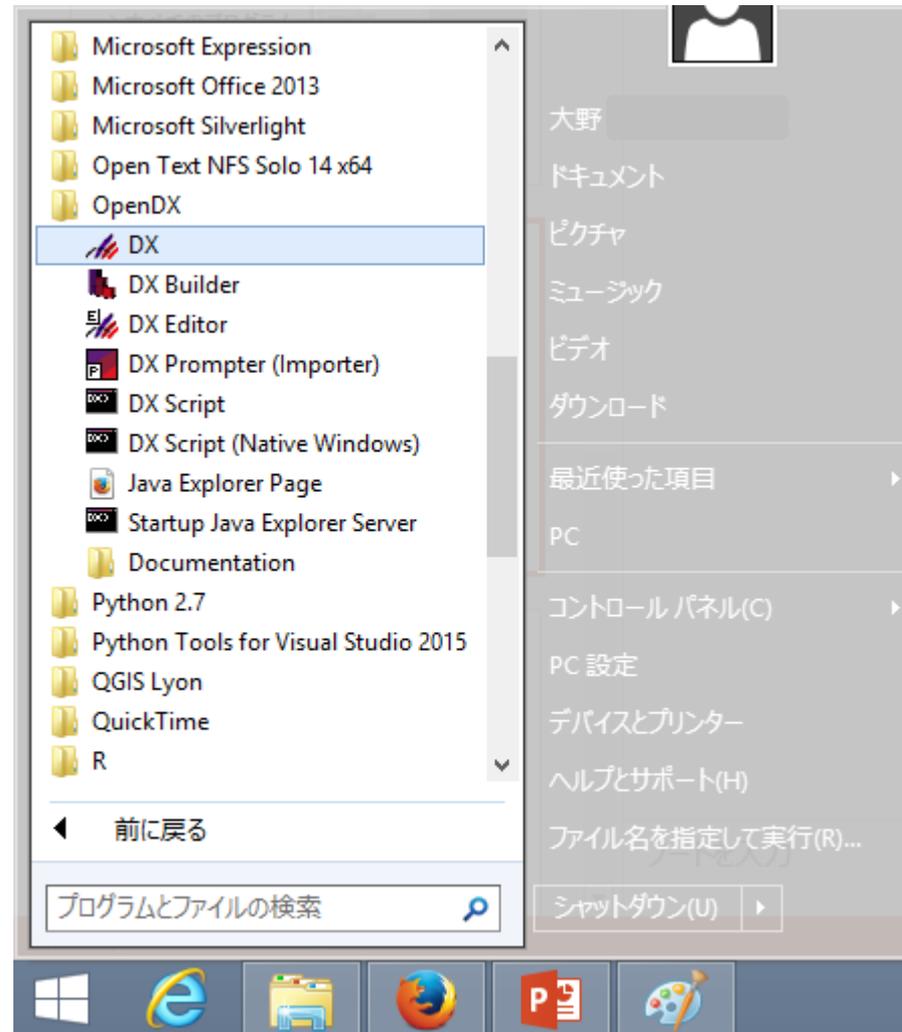
スタートボタン

→すべてのプログラム

→OpenDX

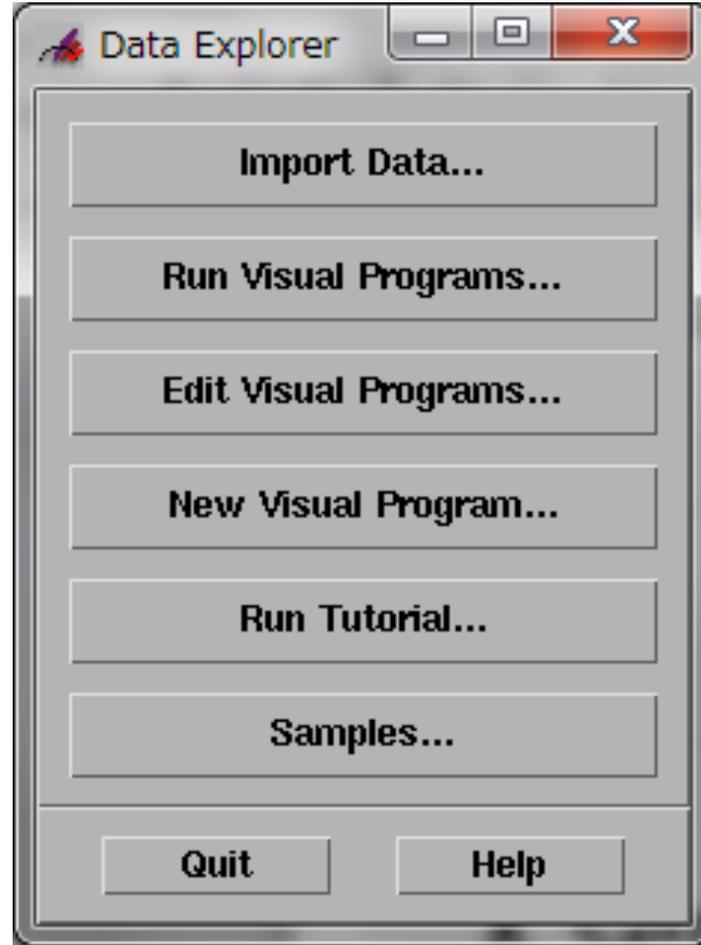
→DX

でOpenDXを起動



OpenDXの起動 -3

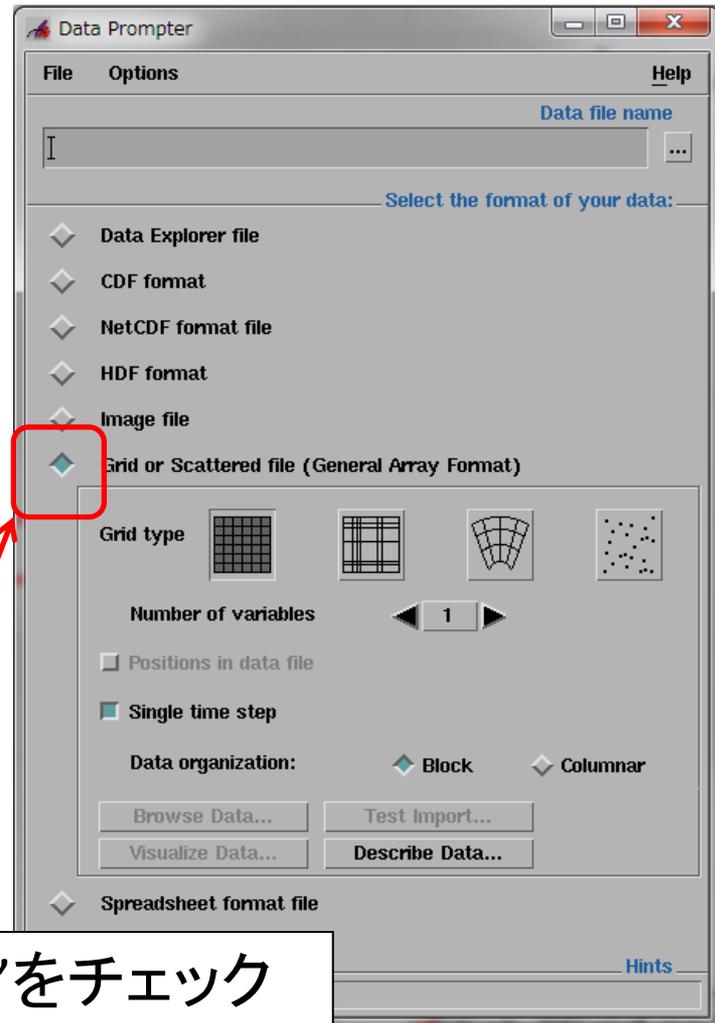
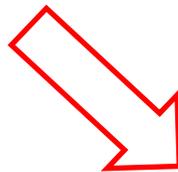
これが出ればOK



General ファイル作成 -1



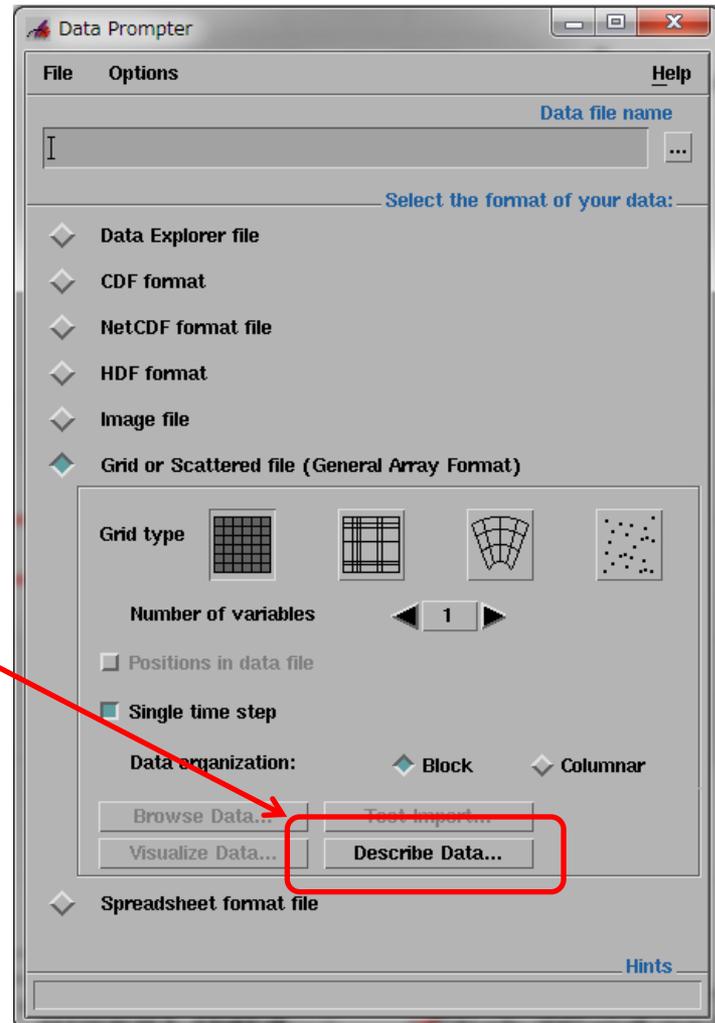
“Import Data”



“Grid or Scattered File”をチェック

General ファイル作成 -2

“Describe Data”をクリック



General ファイル作成 -3

Data Prompter: vect_general.png

File Edit Options

Data file: Z:\Desktop\dx_11th\vect.dat

Header # of bytes

Grid size: 100 x 75 x 120

Data format: Binary (IEEE) Least Significant Byte First

Data order: Row Column

Vector interleaving: $X_0Y_0, X_1Y_1, \dots, X_nY_n$

Grid positions

origin, delta	-5, 0.1
origin, delta	-3.75, 0.1
origin, delta	-6, 0.1
origin, delta	0, 1

Field list: field0

Field name: field0

Type: float

Structure: 3-vector

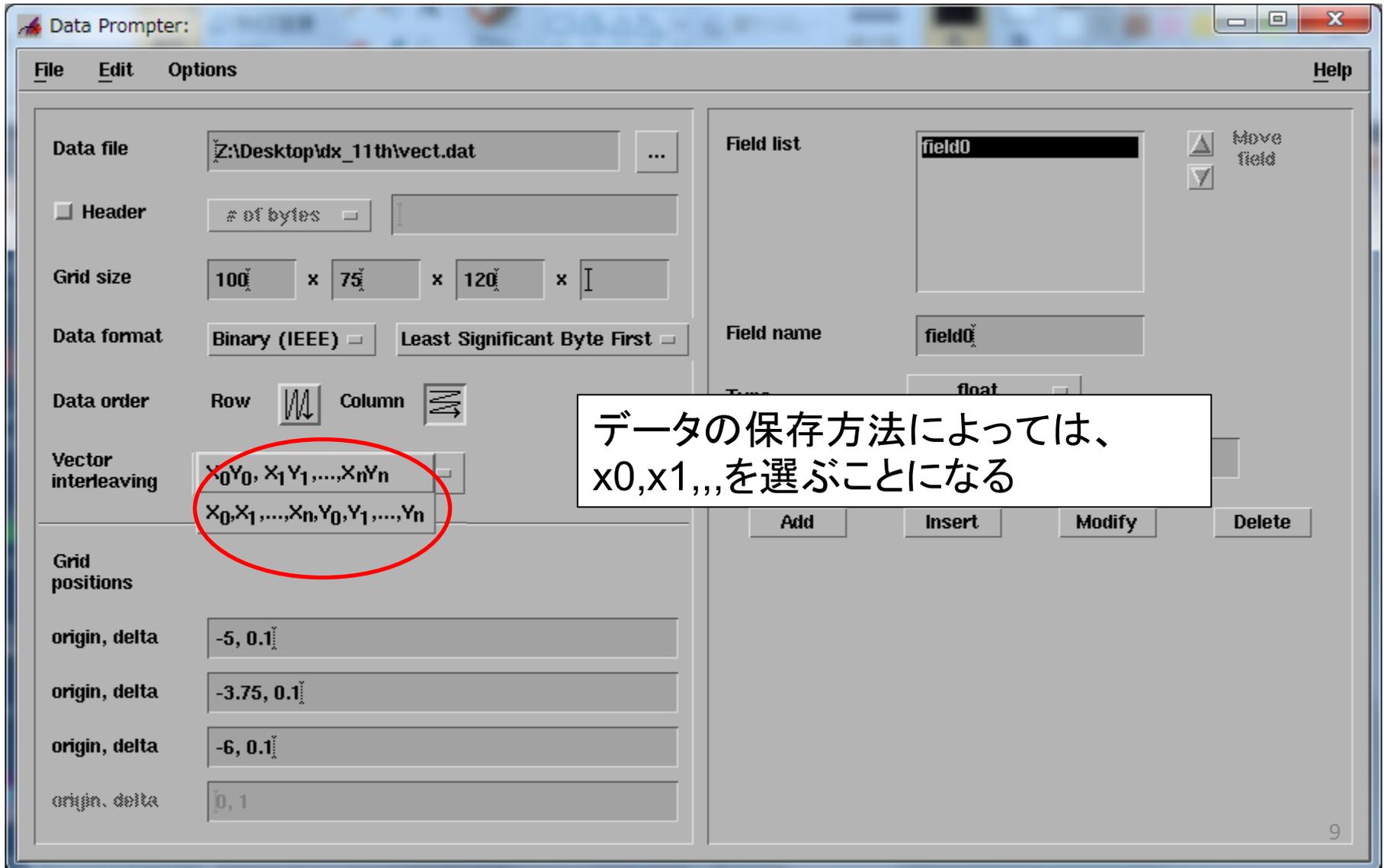
string size

Add Insert Modify Delete

Structureを3-vectorに変えた後 Modifyをクリック

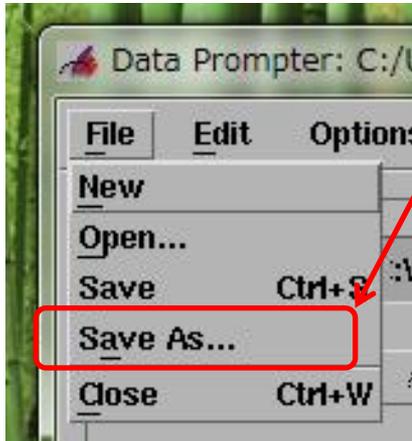
8

General ファイル作成 -3A



データの保存方法によっては、
x0,x1,,,を選ぶことになる

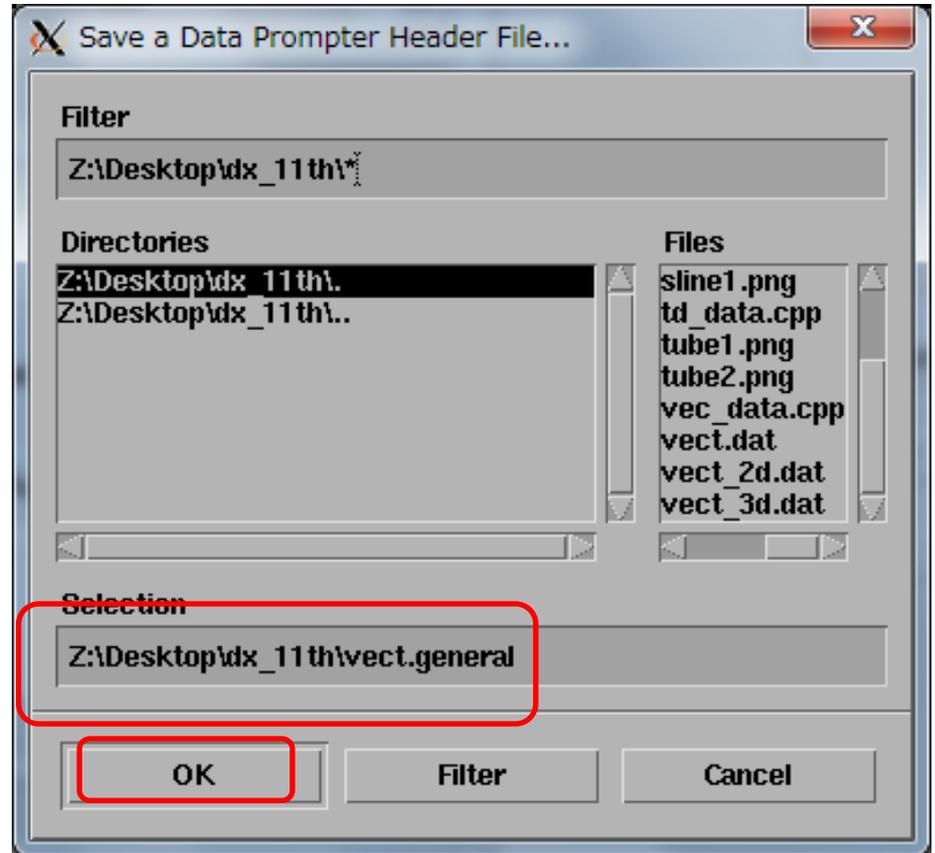
General ファイル作成 -4



File->Save as



vect.generalとして保存



General ファイル作成 -5

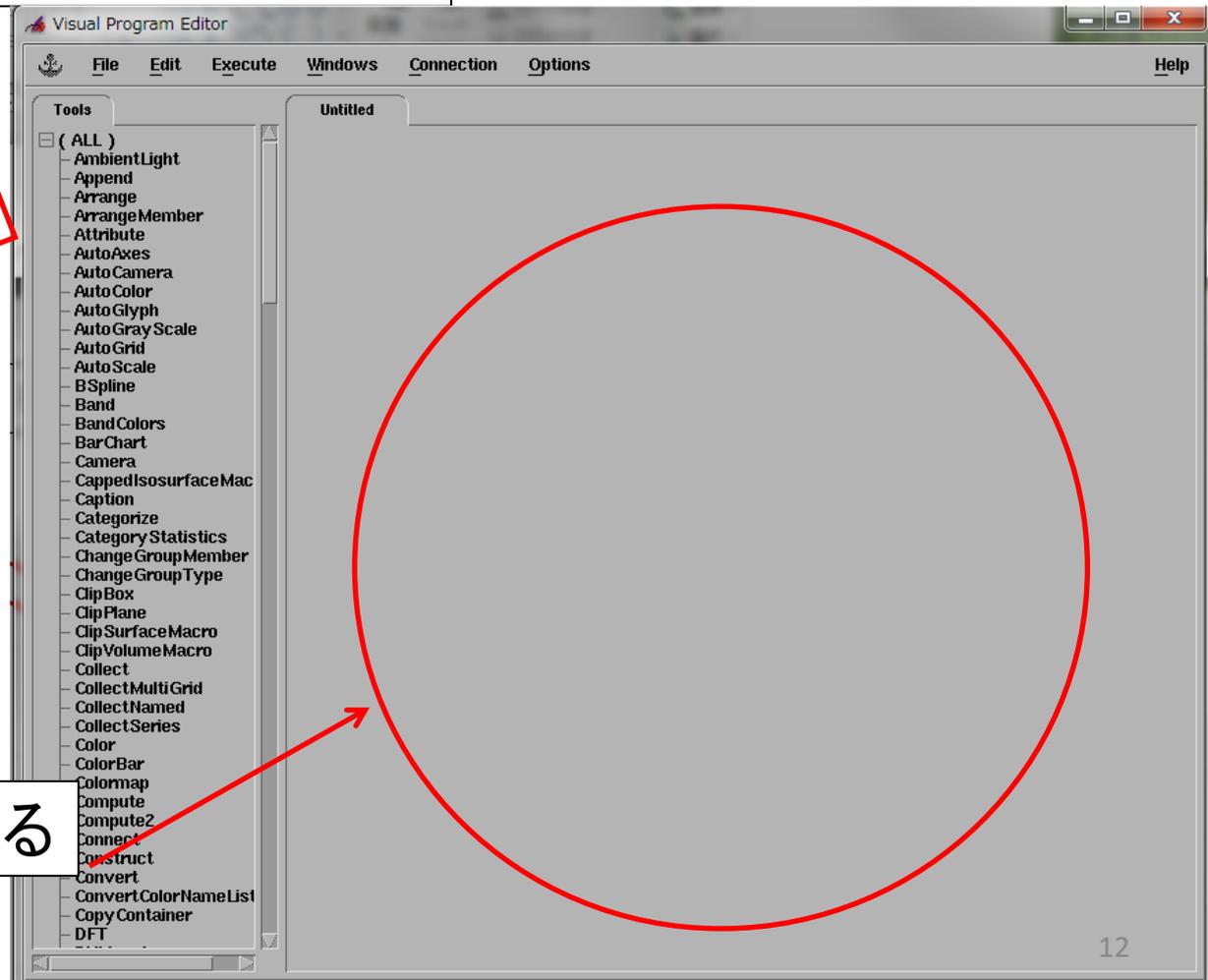
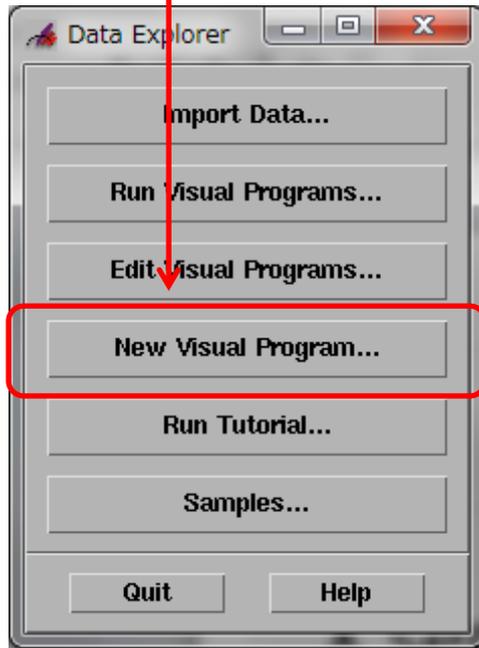
vect.generalの中身

```
file = Z:\Desktop\dx_11th\vect.dat
grid = 100 x 75 x 120
format = lsb ieee
interleaving = record-vector
majority = column
field = field0
structure = 3-vector
type = float
dependency = positions
positions = regular, regular, regular, -5, 0.1, -3.75, 0.1, -6, 0.1

end
```

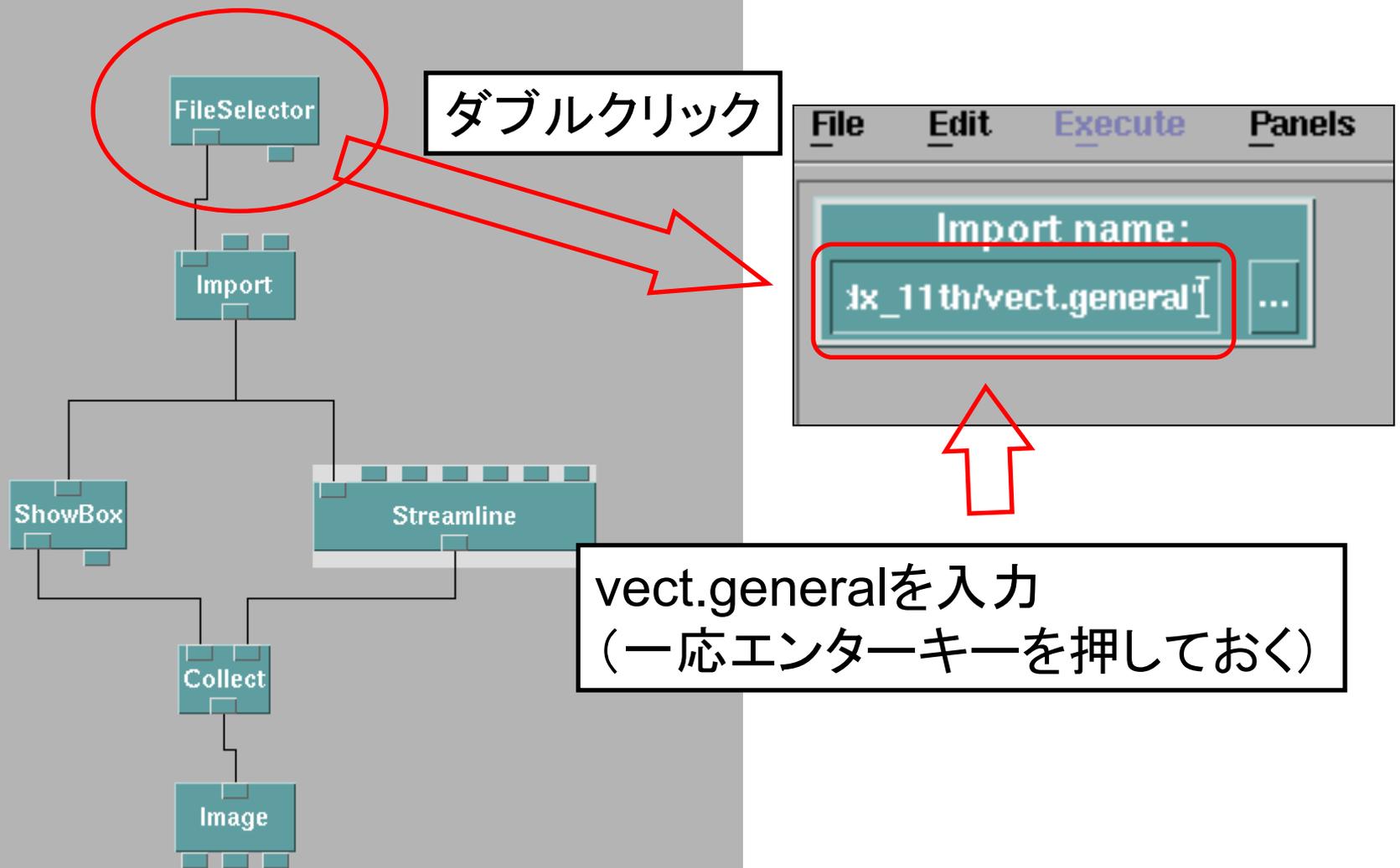
Network作成準備

“New Visual Program”をクリック

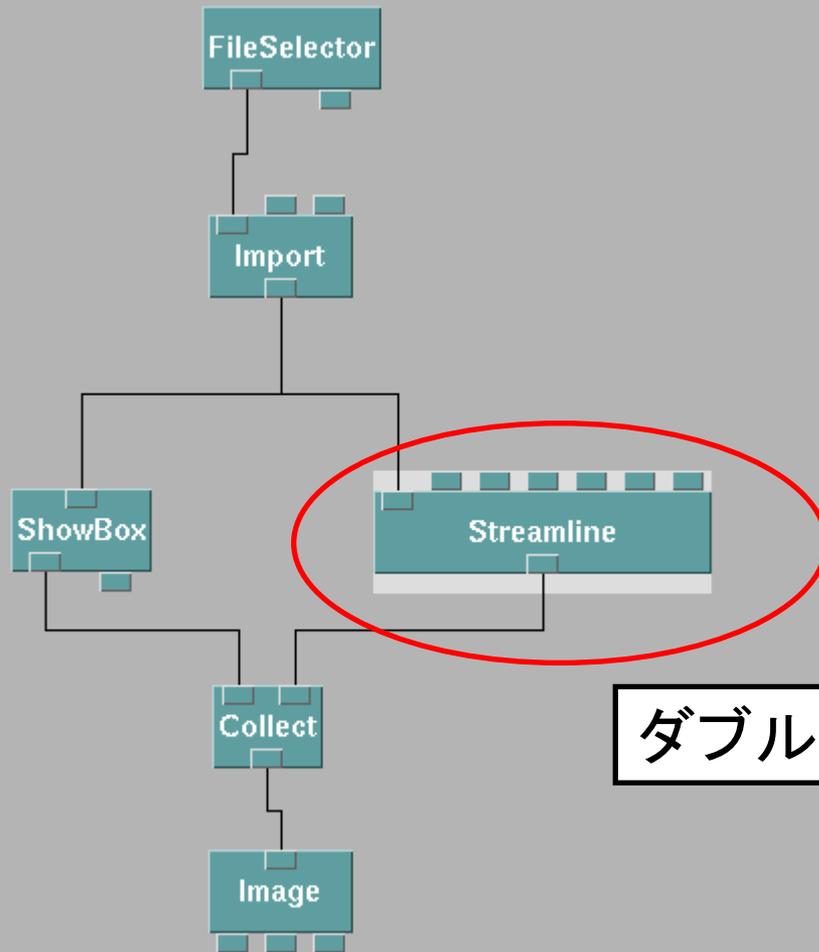


ここにネットワークを作る

StreamLine -1



StreamLine -2



流線の出発点を指定する

ダブルクリック

StreamLine -3

流線の出発点を入力

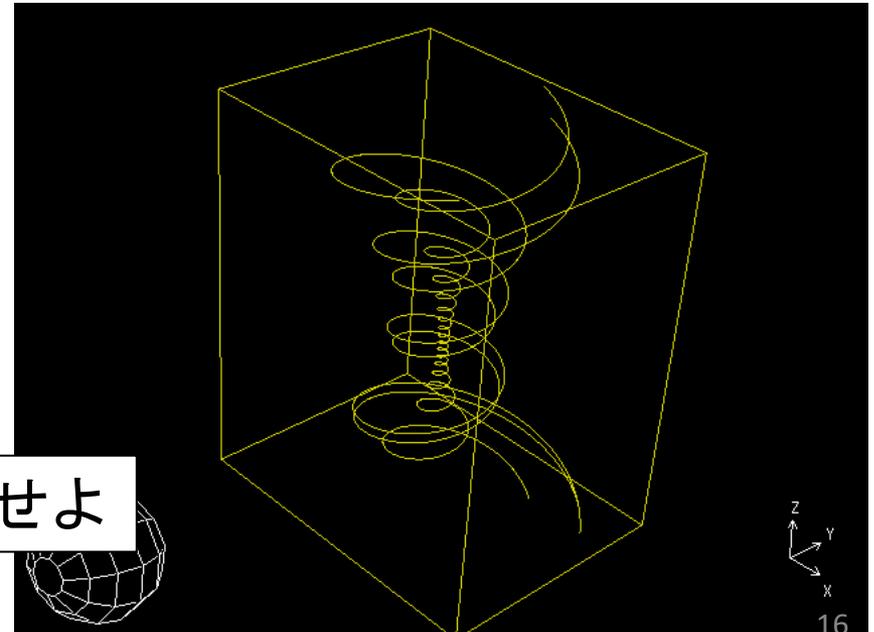
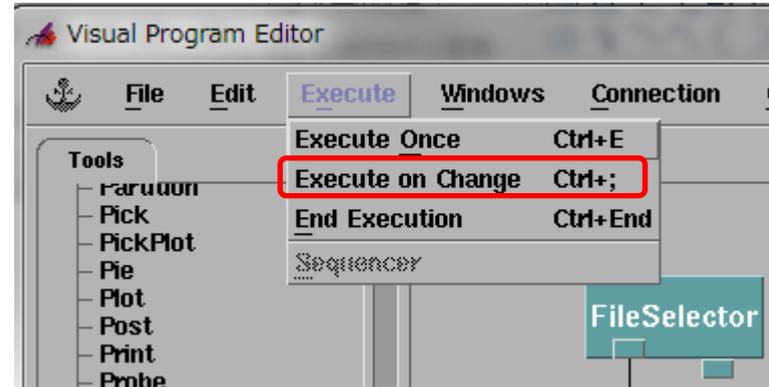
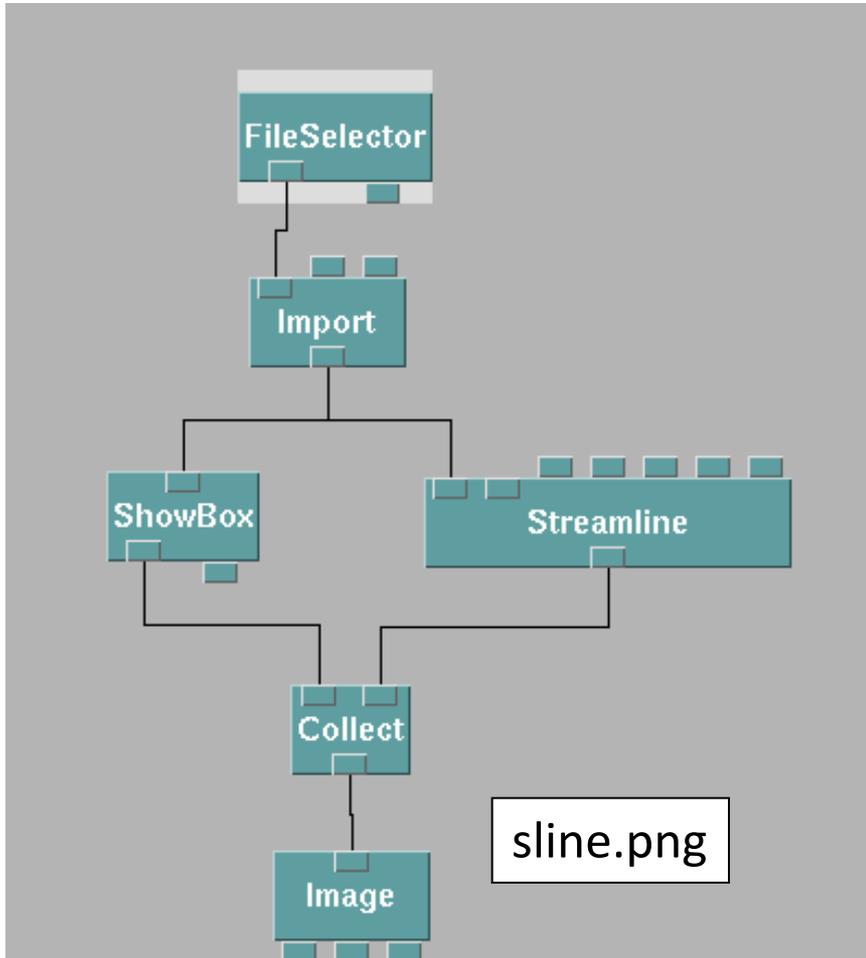
位置

$\{[2,2,-6],[3,3,-6],[4,2,-6]\}$

Example

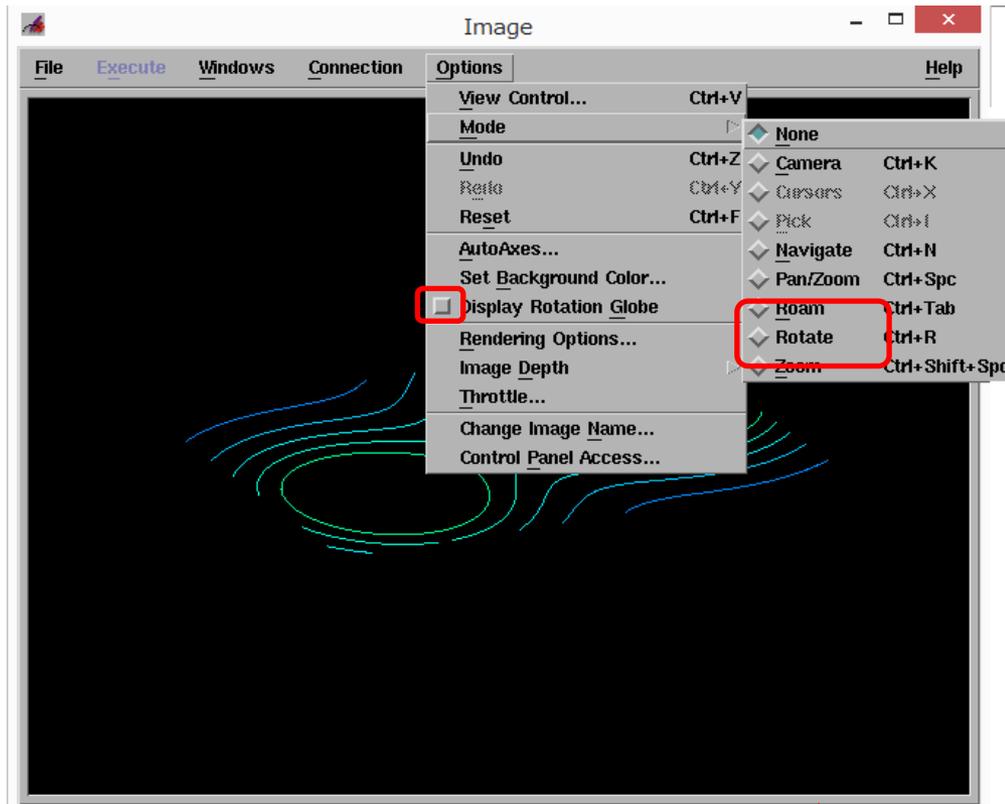
- 1.“Streamline” moduleをダブルクリック
- 2.上記のように編集
- 3.OKをクリック

StreamLine -4



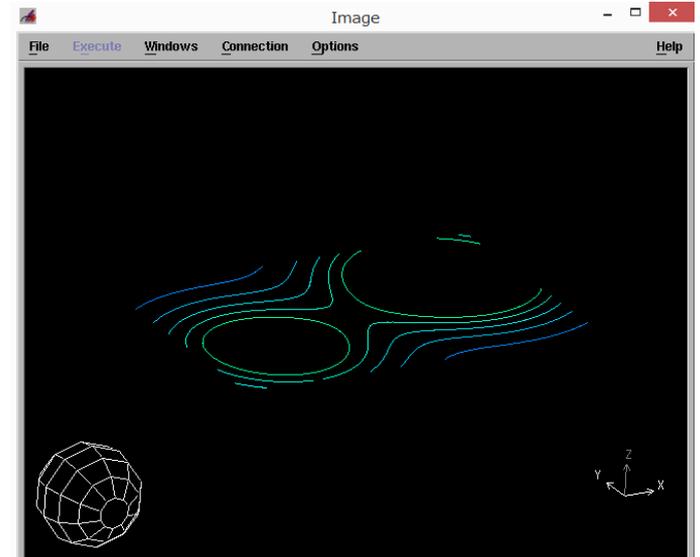
Exercise 1 上記ネットワークを作成せよ

Windowの操作:再掲



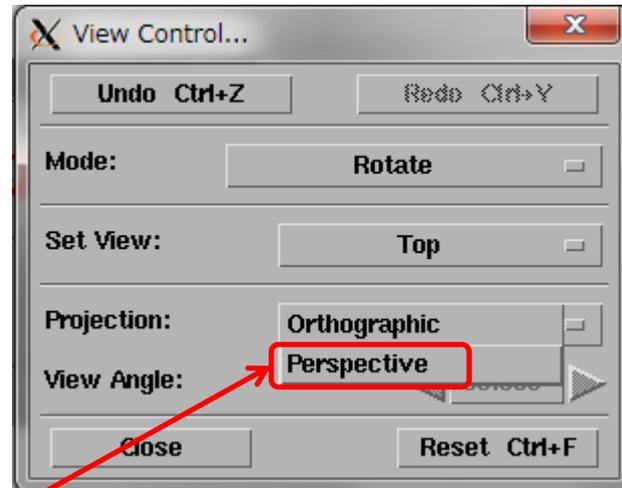
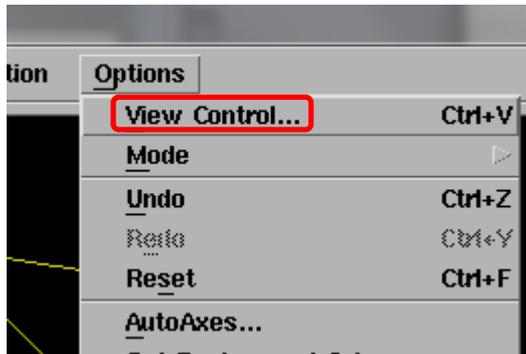
1. Option
2. Mode
3. “Rotate”をチェック
4. (“Display Rotation Globe”もチェック)

Zoom も試してみるとよい



マウスで回せる

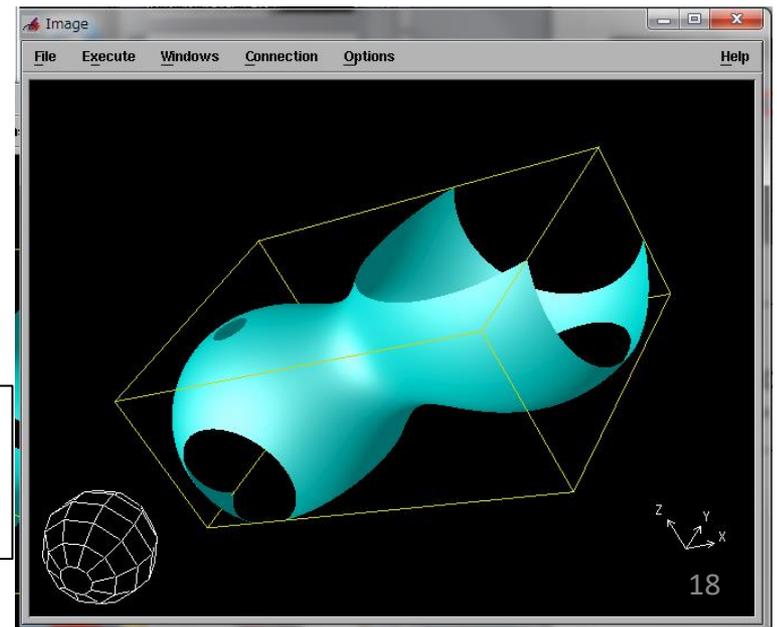
Windowの操作:再掲



1. Option
2. View Control
3. "Perspective"

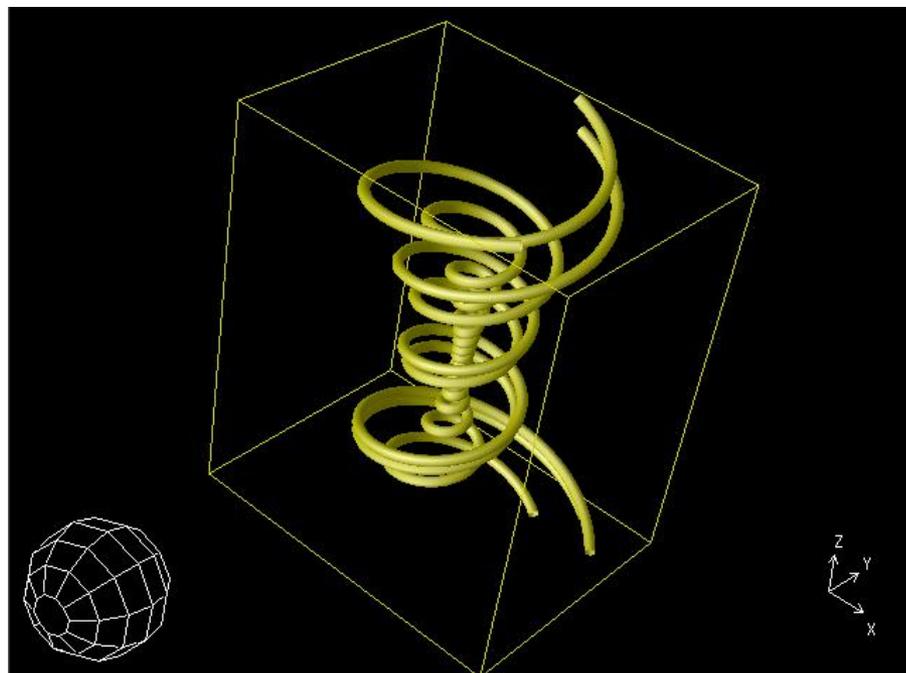
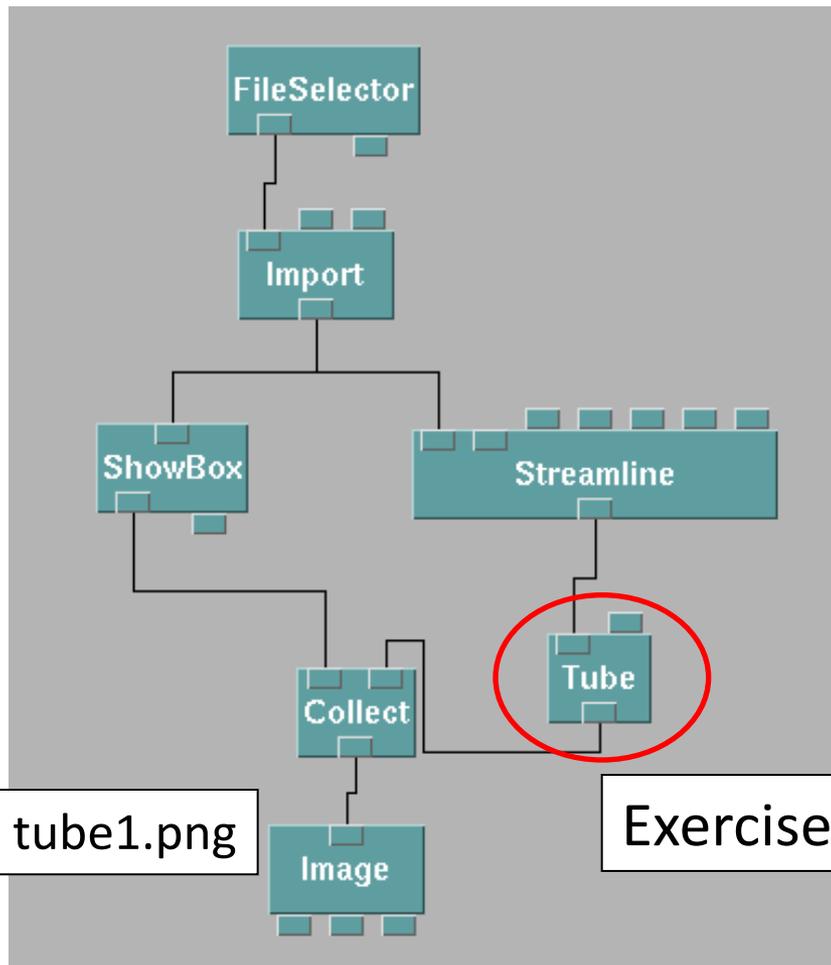


透視射影になる
Ortho->Perspective



Tube -1

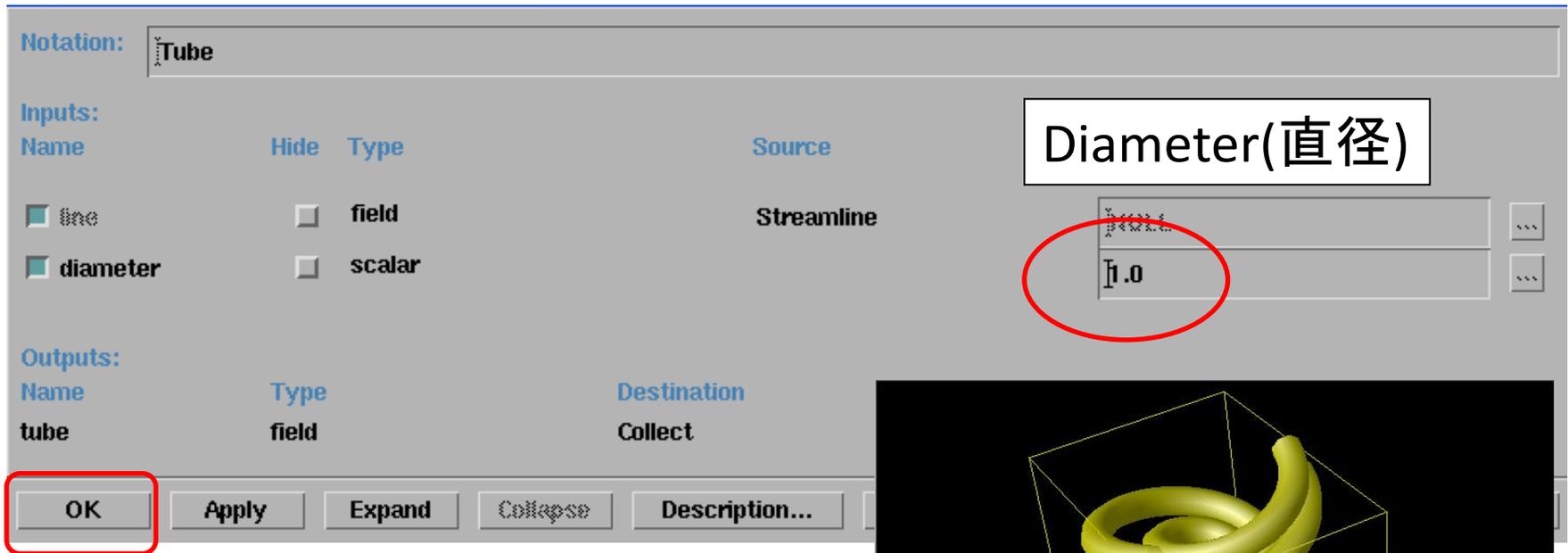
流線をTubeにして迫力を出す



Exercise 2 上記ネットワークを作成せよ

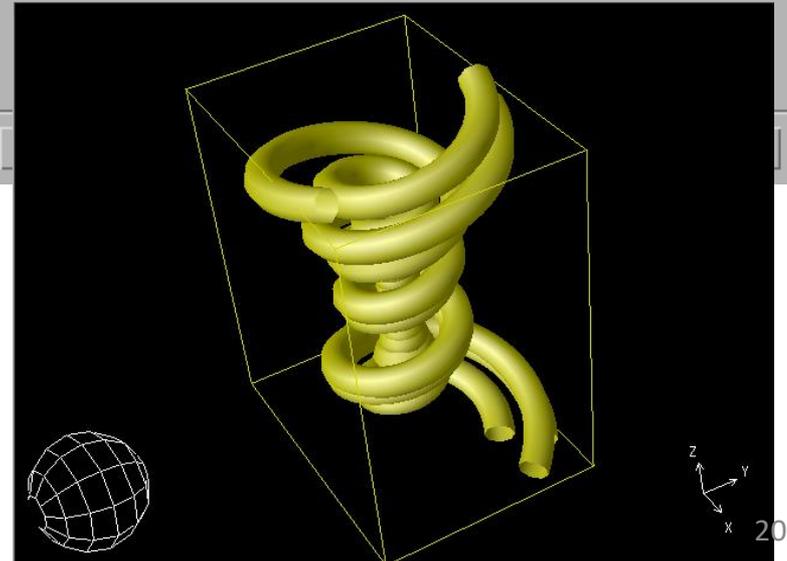
Tube -2

チューブの直径を変えられる

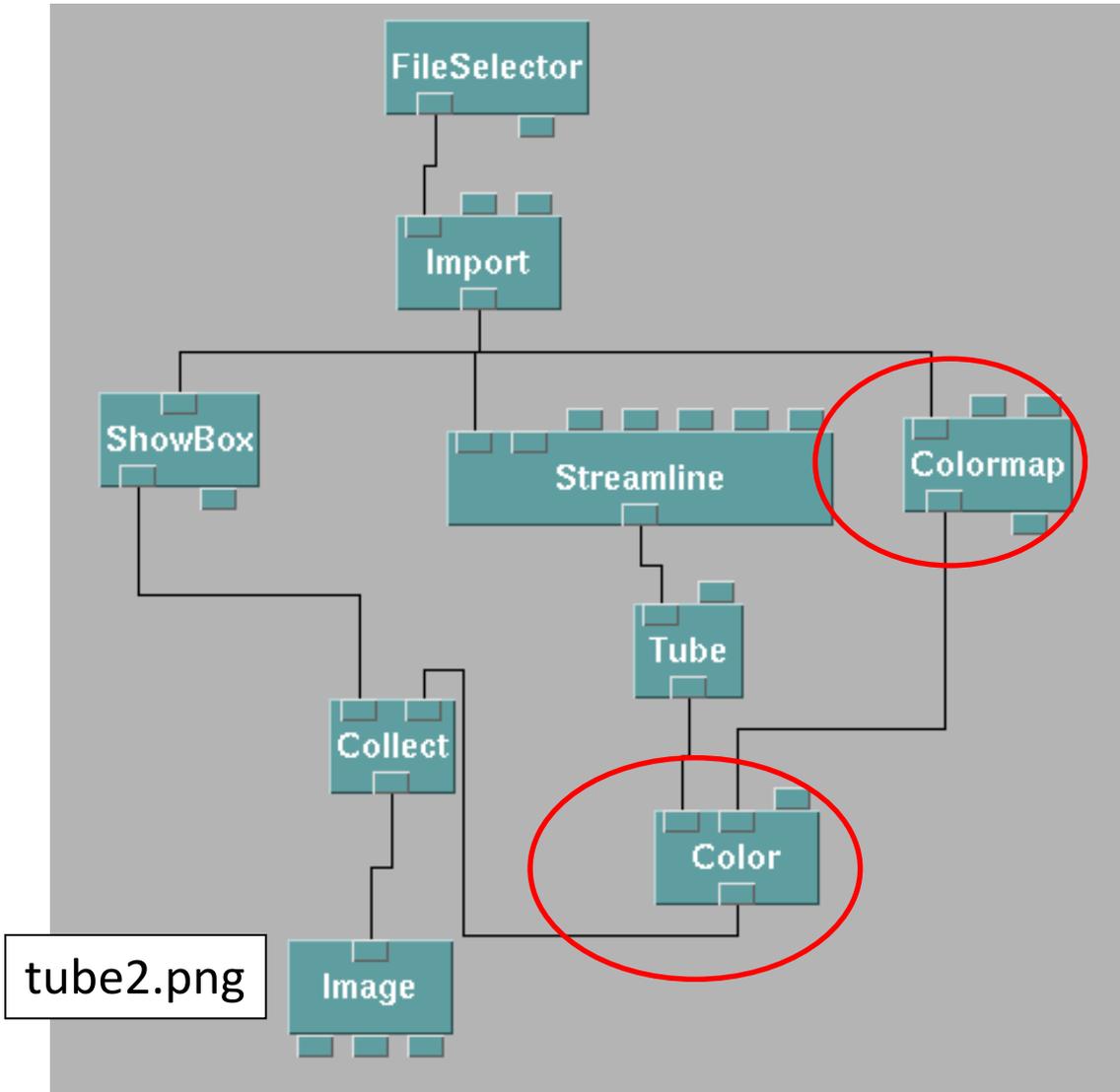


Example

- 1.“Tube” moduleをダブルクリック
- 2.上記のように編集
- 3.OKをクリック

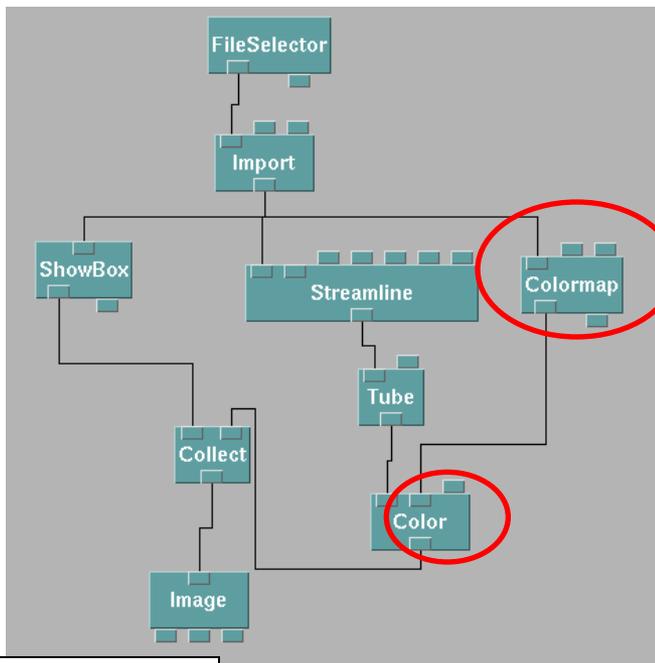


Tube -3

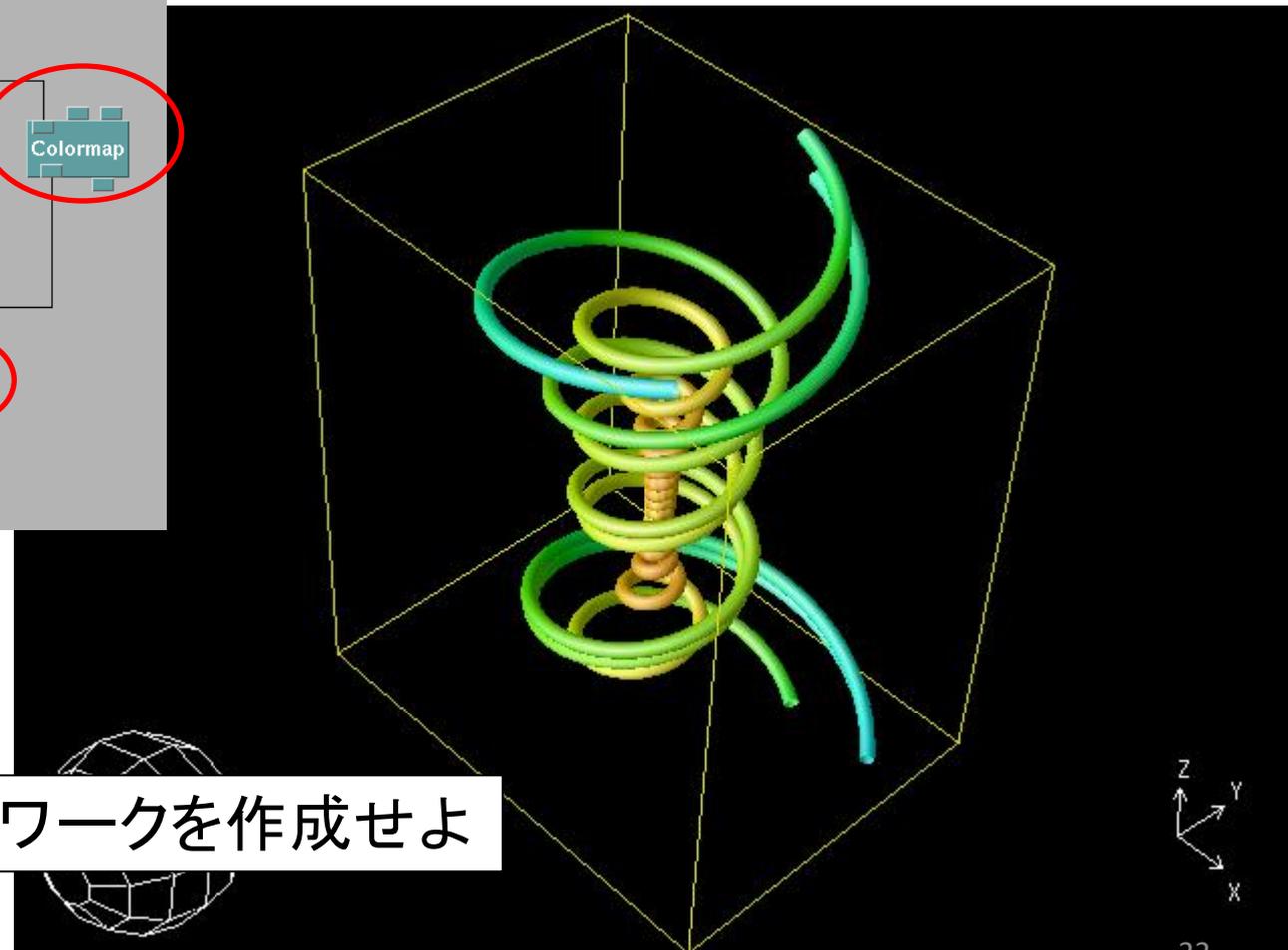


Tubeにベクトル場の大きさに応じて色付けする(流線でもできる)

Tube -4



tube2.png



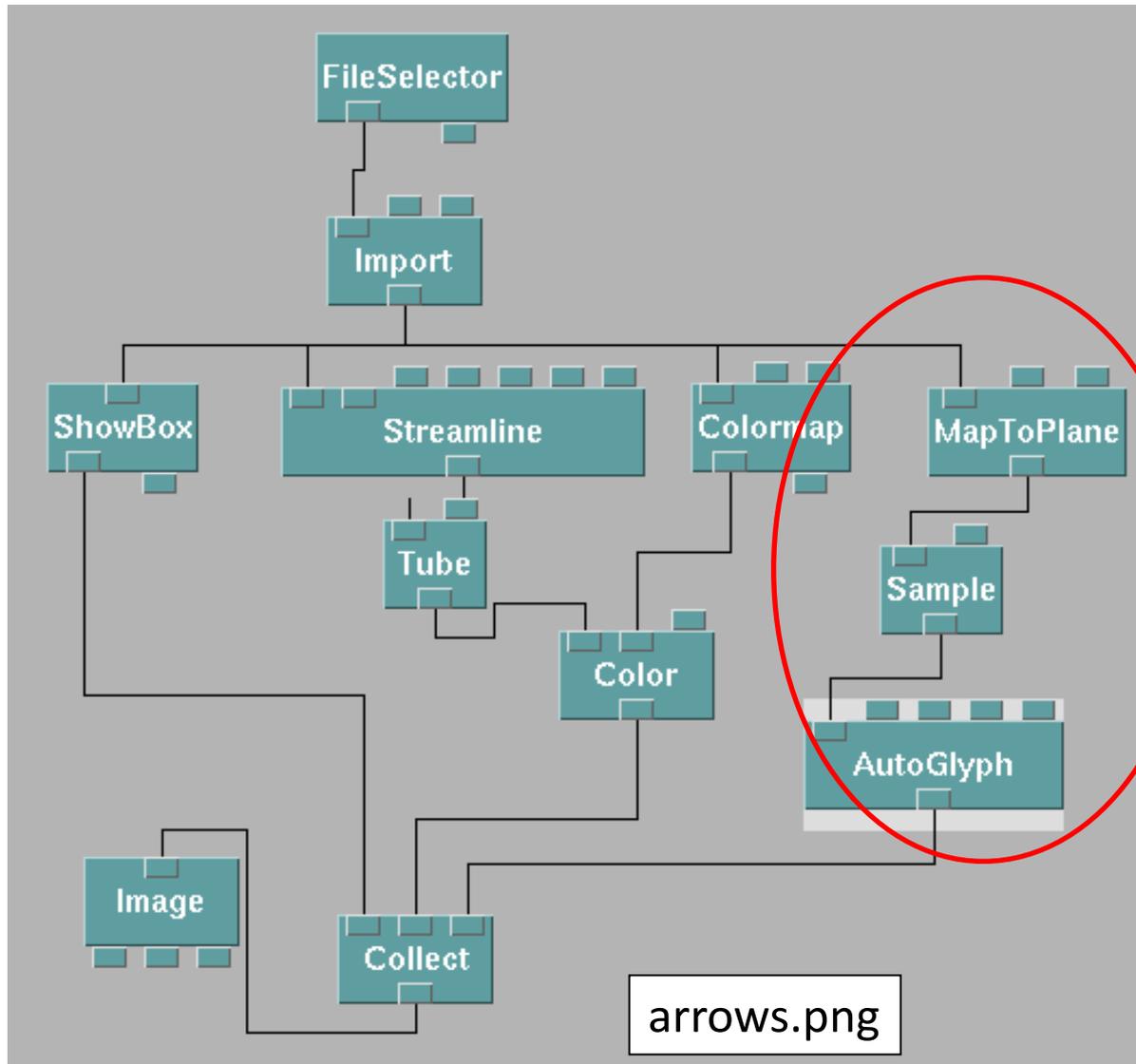
Exercise 3 上記ネットワークを作成せよ

Arrows -1

ベクトル場を矢印で表現する;

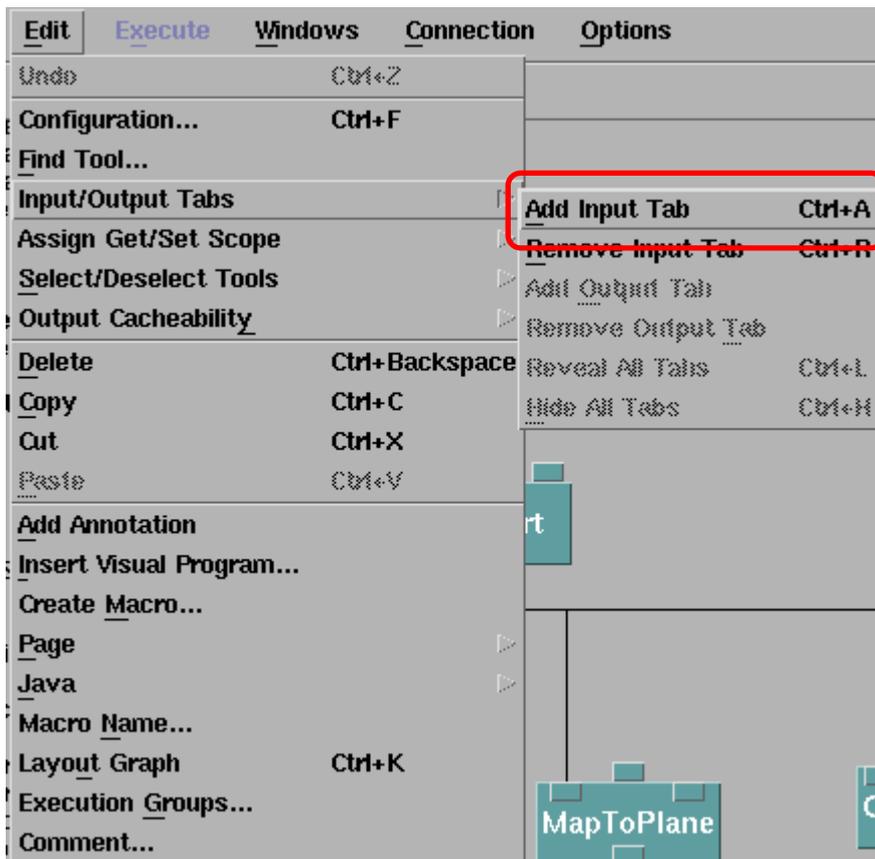
ただし、データ全体に矢印を表示すると、かえってわかりにくいので、一部分にのみ矢印を配置する

Arrows -2

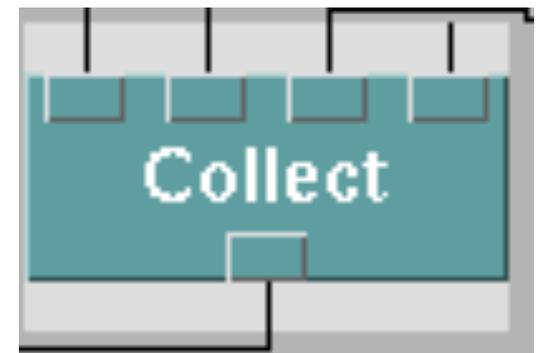


第10回Appendix -1

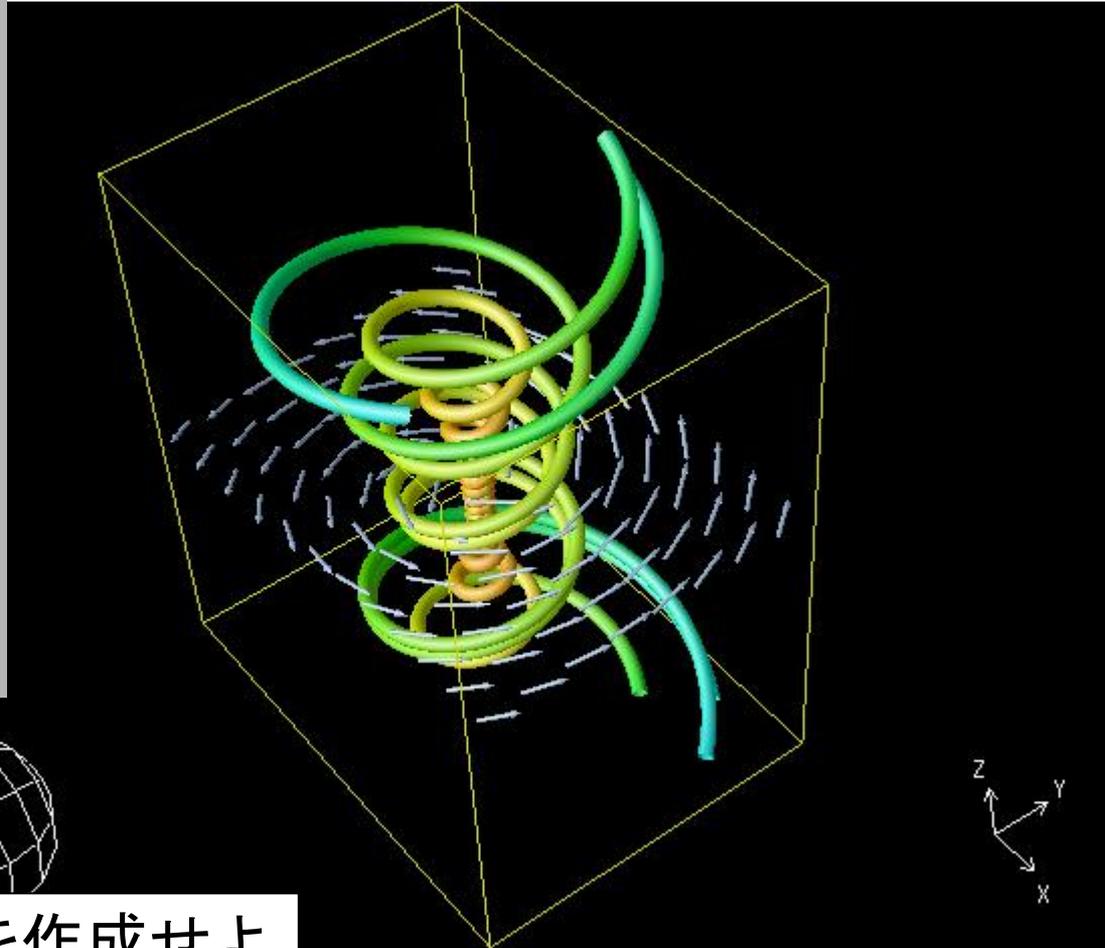
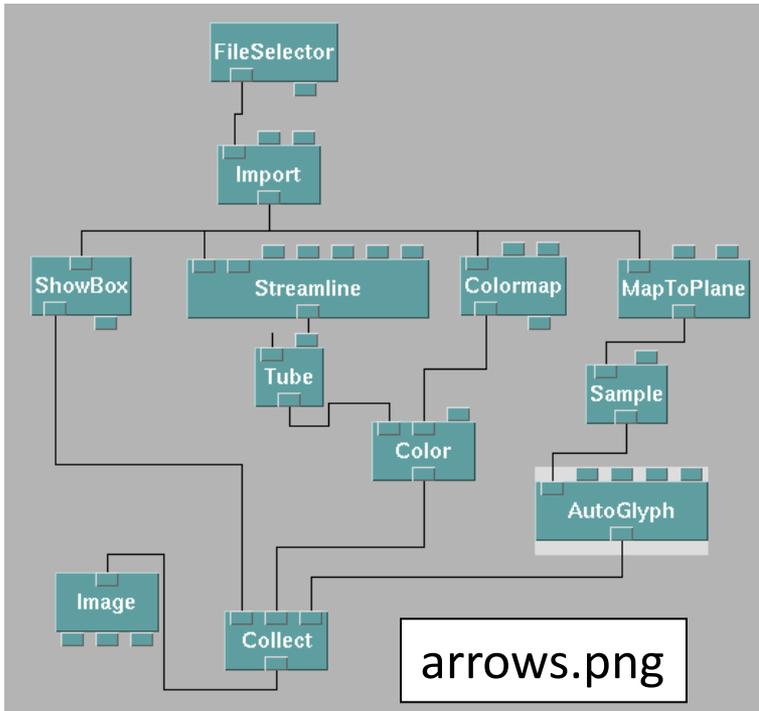
- Collectの入りを増やす



1. “Collect” moduleをクリック
2. Edit -> “Input/Output Tabs” -> “Add Input Tabs”



Arrows -3

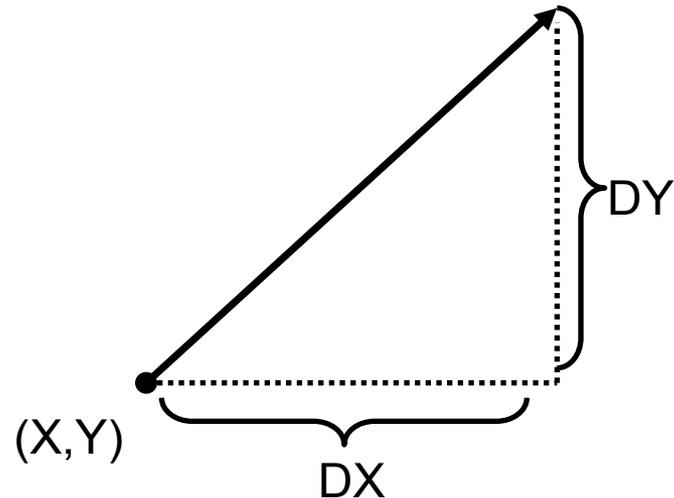


Exercise 4 上記ネットワークを作成せよ

Appendix A-1

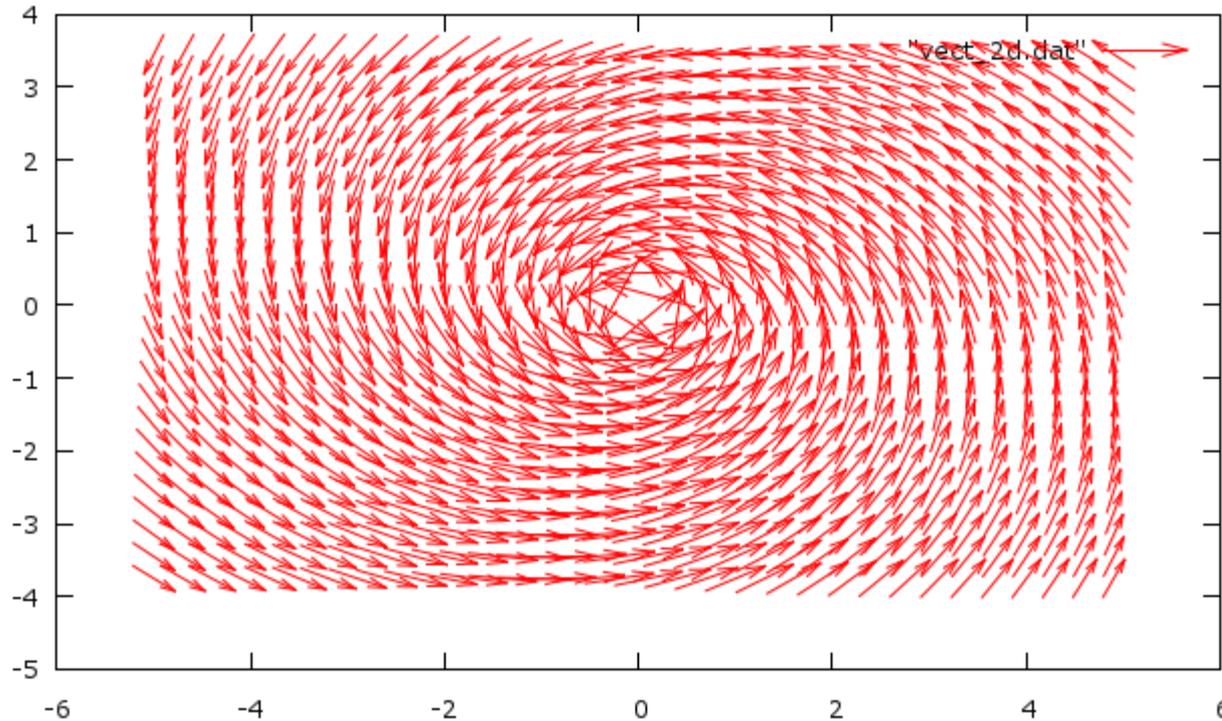
- Gnuplotによるベクトル場の可視化
- 2次元ベクトル場
データフォーマット

```
X1 _ Y1 _ DX1 _ DY1 ↵  
X2 _ Y2 _ DX2 _ DY2 ↵  
...
```



Appendix A-2

```
gnuplot> plot "vect_2d.dat" with vectors
```

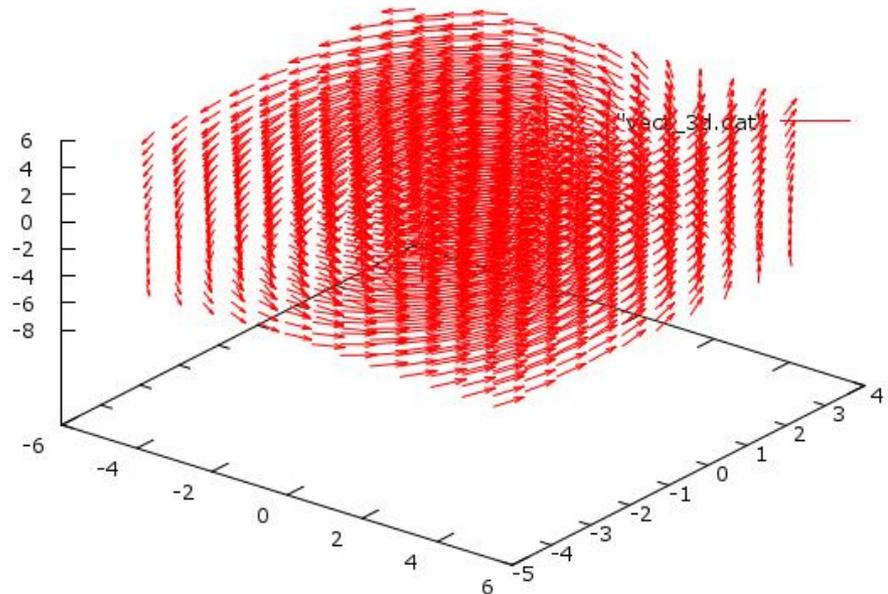


矢印の長さを短くしたほうがよさそう

Appendix A-3

3次元ベクトル場
データフォーマット

```
X1 _ Y1 _ Z1 _ DX1 _ DY1 _ DZ1 ↵  
X2 _ Y2 _ Z2 _ DX2 _ DY2 _ DZ2 ↵  
...
```



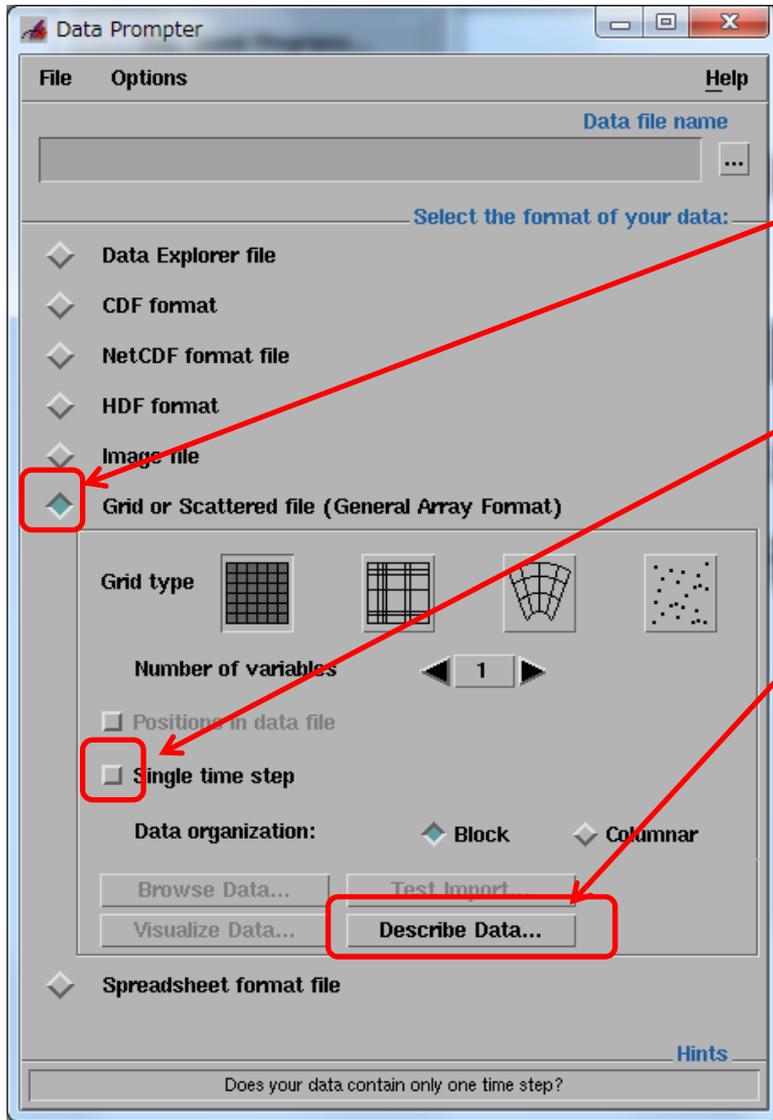
```
gnuplot> splot _ "vect_3d.dat" _ with _ vectors ↵
```

vect_2d.dat, vect_3d.datは、gp_data.cppで作成した。
いずれも矢印の長さを短くしたほうが良さそう。
gp_data.cppをどう変えればよいか考えてみよう。

Appendix B-1

- OpenDXによる時系列データの取り扱い
- scal.datを使う(80x70x60, T=1~20)
 1. .generalファイル作成法
 2. サンプルネットワーク
 3. 連番画像
 - 連番画像保存
 - 連番画像処理

Appendix B-2



1. Grid or Scattered fileをクリック
2. “Single time step”を解除
3. Describe Dataをクリック

Appendix B-3

The screenshot shows the 'Data Prompter' application window. The 'Data file' field contains 'Z:\Desktop\idx_11\thscal.dat'. The 'Series' section has 'n' set to 20, 'start' to 1, and 'delta' to 1. A red box highlights these three fields, with a red arrow pointing to a text box containing the Japanese text '時刻数を入れる' (Enter the number of time points). Another red box highlights the 'Data file' field, with a text box below it containing 'scal_td1.png'. A third red box highlights the 'File' menu, with a text box below it containing 'File -> Save asで scal_td.generalとして保存' (Save as scal_td.general via File -> Save as).

Data Prompter: File Edit Options Help

Data file: Z:\Desktop\idx_11\thscal.dat

Header: # of bytes: []

Grid size: 80 x 70 x 60 x []

of Points: []

Data format: Binary (IEEE) Least Significant Byte First

Data order: Row Column

Field interleaving: Block Vector interleaving $X_0Y_0, X_1Y_1, \dots, X_{n-1}Y_{n-1}$

Series: n: 20 start: 1 delta: 1

Series interleaving: $F_1^S, F_2^S, F_1^S, F_2^S, \dots, F_1^S, F_2^S$

Series separator: # of bytes: []

Grid positions: Completely regular

regular origin, delta: -4, 0.1

regular origin, delta: -3.5, 0.1

regular origin, delta: -3, 0.1

regular origin, delta: 0, 1

Field list: field0 Move field

Field name: field0

Type: float

Structure: 時刻数を入れる

Dependency: Layout skip width: [] Block skip # elem: [] width: []

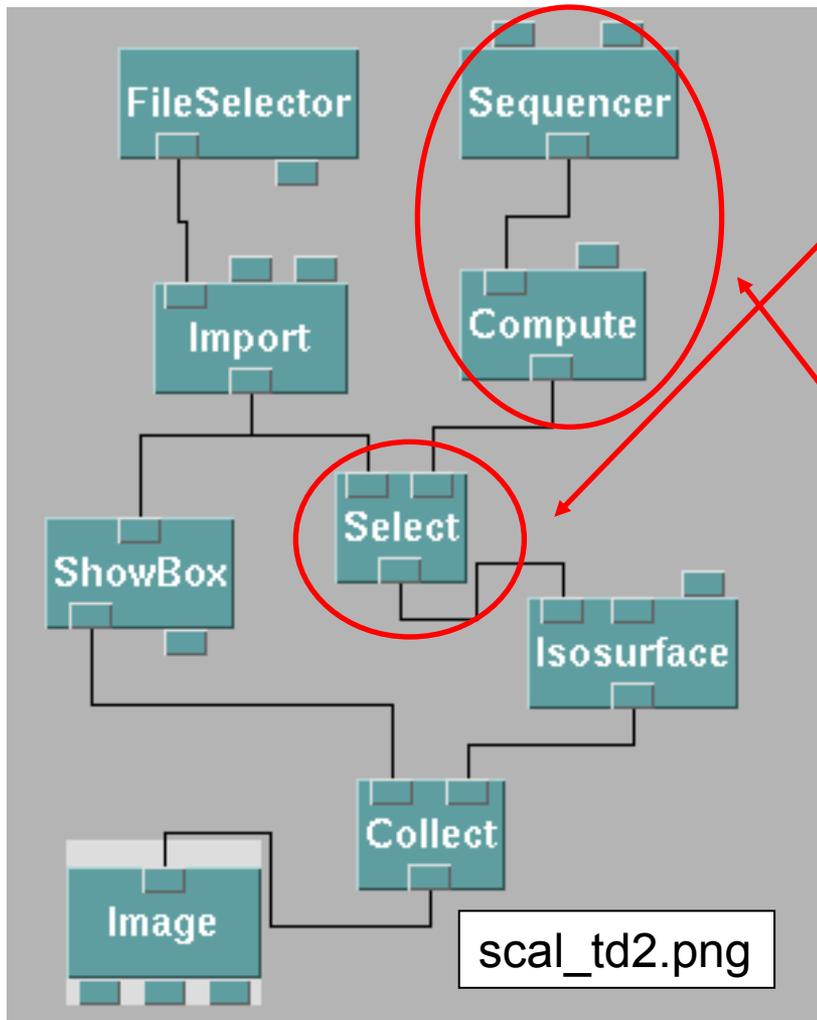
Add Insert Modify Delete

Record Separator: same between all records: # of bytes: [] between records: # of bytes: []

scal_td1.png

File -> Save asで scal_td.generalとして保存

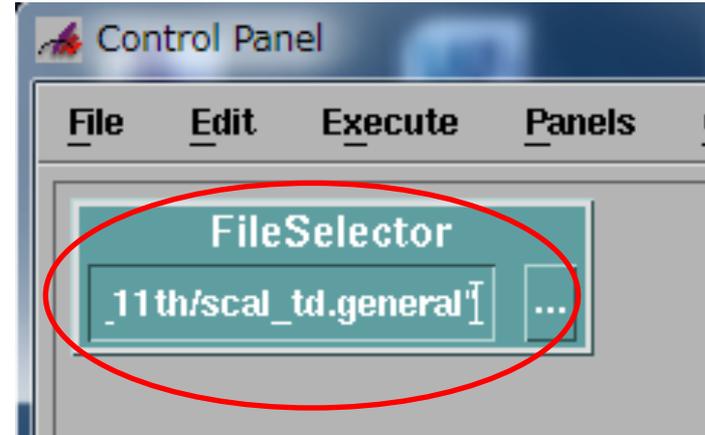
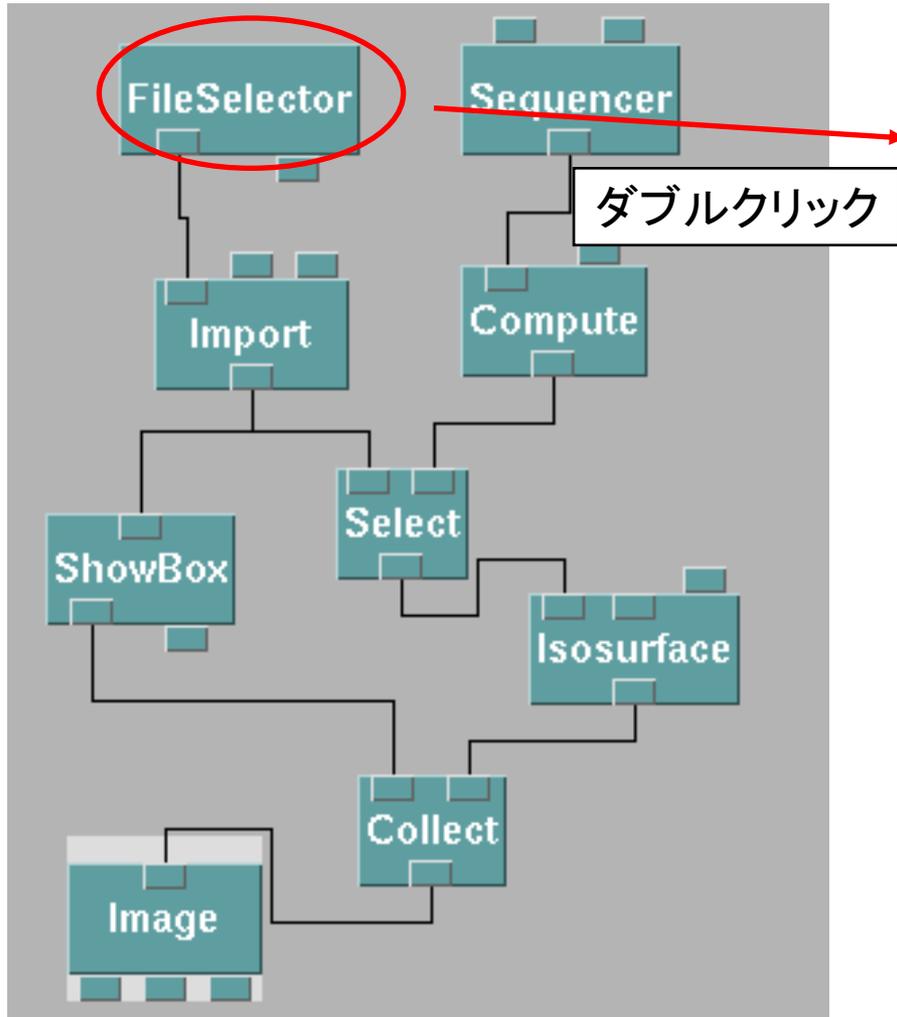
Appendix B-4



Selectで時刻を選ぶ

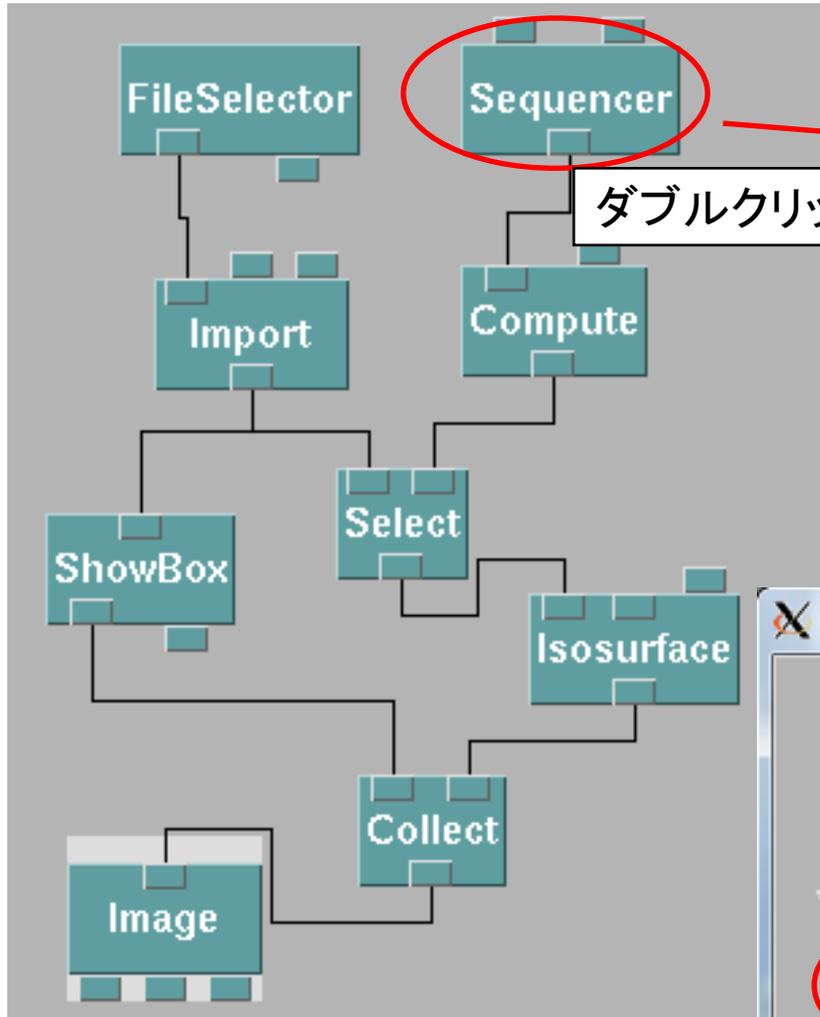
SequencerとComputeで
インタラクティブに時刻を
選ぶ

Appendix B-5



scal_dt.generalをインプット

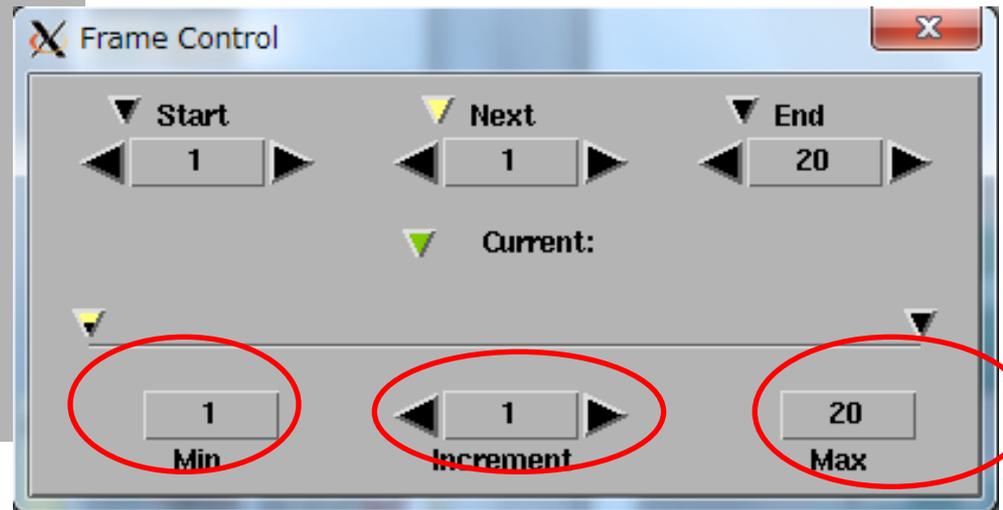
Appendix B-6



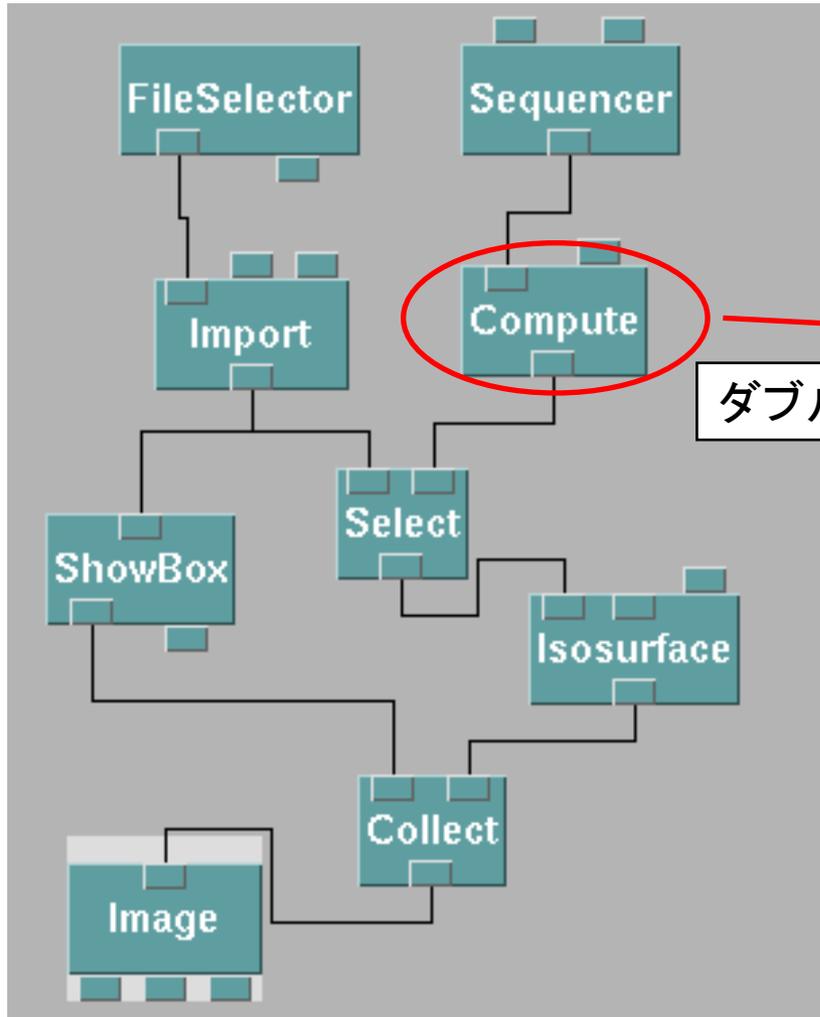
ダブルクリック



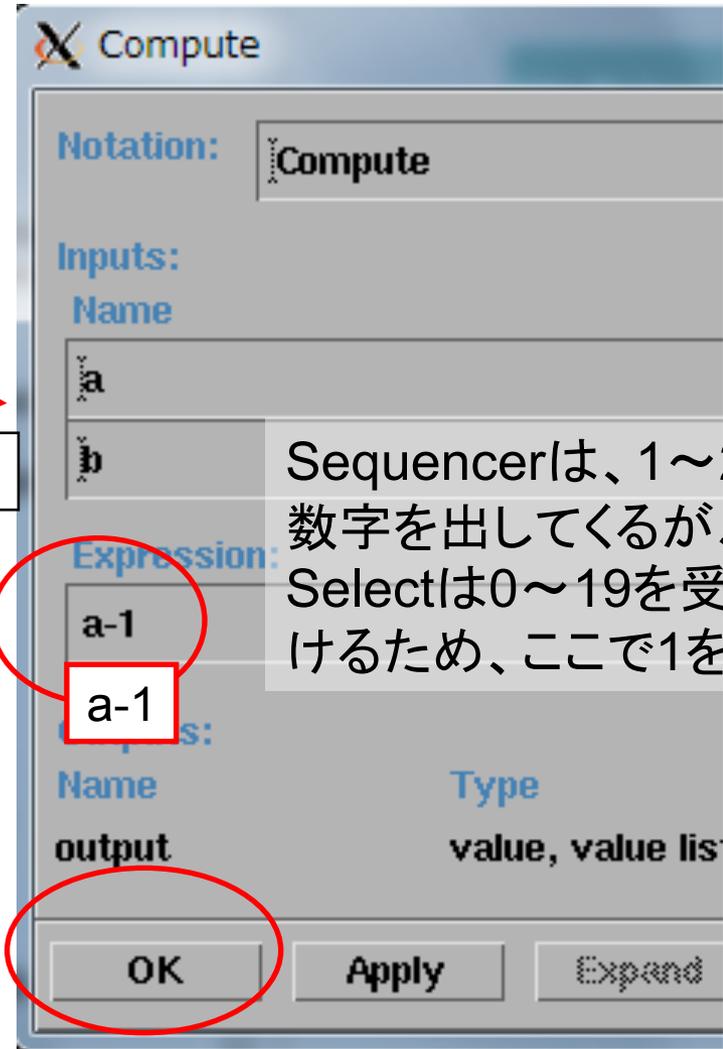
クリック



Appendix B-7

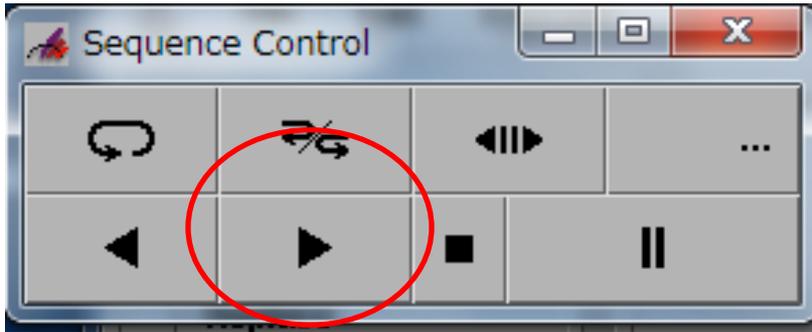


ダブルクリック

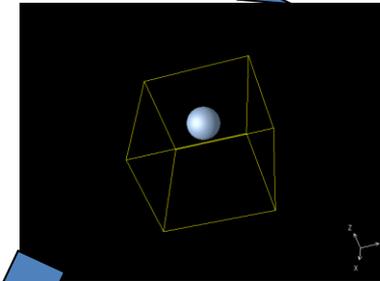
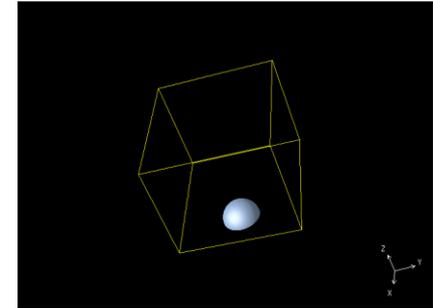


Sequencerは、1～20の数字を出してくるが、Selectは0～19を受け付けるため、ここで1を引く

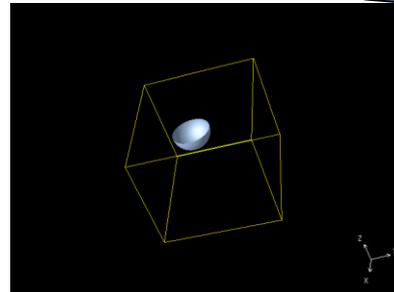
Appendix B-8



このボタンで再生できる

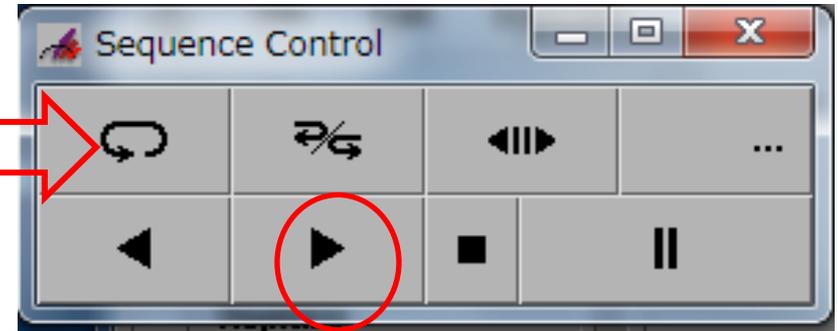
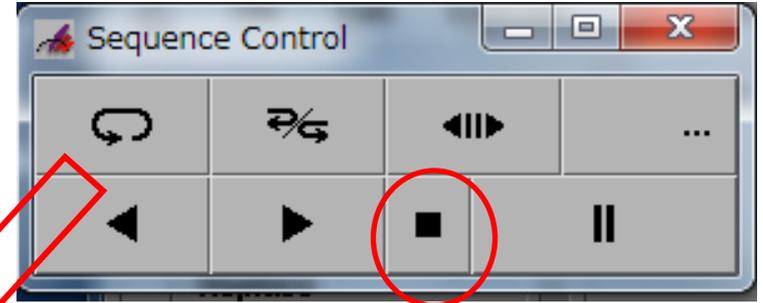
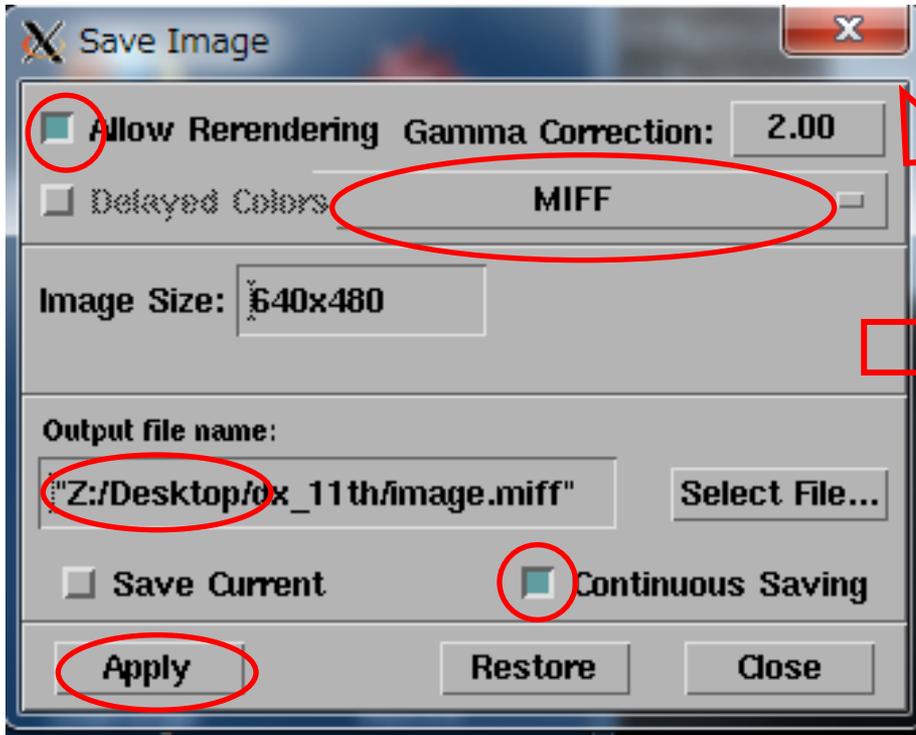


Isosurface level = 0.9



Appendix B-9

画像保存法



画像が出ている窓のfile->image save

再生が終了したら

- Continuous Saving をoff
- Close

Appendix B-10

Miff形式のファイルは、Linuxのconvertコマンドで、連番画像に分解することができる。

```
# convert _+adjoin _image.miff _image.tiff ↵
```

-> image-0.tiff, image-1.tiff が生成される。

Windows用のImageMagick(フリー)をインストールすれば、Windowsでもこのコマンドが使えるようになる。

Appendix B-11

```
xterm
Main Options  VT Options  VT Fonts
sh-4.1$ ls
image.miff
sh-4.1$ convert +adjoin image.miff image.tiff
sh-4.1$ ls
image-0.tiff  image-13.tiff  image-18.tiff  image-5.tiff  image.miff
image-1.tiff  image-14.tiff  image-19.tiff  image-6.tiff
image-10.tiff image-15.tiff  image-2.tiff   image-7.tiff
image-11.tiff image-16.tiff  image-3.tiff   image-8.tiff
image-12.tiff image-17.tiff  image-4.tiff   image-9.tiff
sh-4.1$ █
```

convertコマンドはCygwinで使用可能