JAERI-Tech
96-008

AVS (Application Visualization System) 利用手引書

1996年3月

増子献児*・五來一夫・加藤克海*・山崎和彦

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所技術情報部情報資料課(〒319-11 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Information Division, Department of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

© Japan Atomic Energy Research Institute, 1996

編集兼発行　日本原子力研究所
印　　刷　㈱高野高速印刷

JAERI-Tech 96-008

ＡＶＳ（Application Visualization System）利用手引書

日本原子力研究所計算科学技術推進センター
増子　献児＊・五來　一夫・加藤　克海＊・山崎　和彦

（1996年１月31日受理）

情報システム管理課では，分散処理環境の整備作業の一環として画像処理環境の整備を進めており，画像処理サーバ，画像処理ソフトウェアＡＶＳ，動画像処理システム（ビデオ装置，フレームスキャンコンバータ等）を導入している。本報告書は，整備した画像処理環境において画像処理ソフトウェアを効率的に利用するための情報及び画像処理環境を整備するための情報をとりまとめたものである。

日本原子力研究所：〒319－11　茨城県那珂郡東海村白方白根2-4
＊　高度情報科学技術研究機構

JAERI-Tech 96-008

AVS (Application Visualization System) User's Guide

Kenji MASUKO*, Kazuo GORAI
Katsumi KATO* and Kazuhiko YAMAZAKI

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received January 31, 1996)

Computer and network environment for image processing has been developed and maintained under the course of establishing a distributed processing environment by the information system operating division. We introduced a server for image processing, AVS for image processing software and an animation processing system (video, frame scan converter).

This report summarizes the information to use AVS and to develop and maintain computer and network environment for image processing.

Keywords : AVS (Application Visualization System), Animation, Image Processing, Visualizaition.

* RIST (Research Organization for Information Science Technology)

目　　次

はじめに …… 1
1．AVSの操作手順 …… 2
1.1　AVSを利用するための環境 …… 2
1.2　AVSの起動 …… 5
1.3　ネットワークエディタ …… 6
1.4　ネットワークの読み込み，保存 …… 9
1.5　ネットワークの作成 …… 15
1.6　ネットワークの実行 …… 22
1.7　ネットワークエディタメニュー …… 30
1.8　ネットワークの作成例 …… 36
2．AVSで扱われるデータタイプ …… 51
2.1　フィールドデータ …… 53
2.2　ジオメトリデータ …… 67
3．サブシステム …… 72
3.1　サブシステムの起動，終了 …… 72
3.2　ジオメトリビューワーサブシステム …… 73
3.3　ジオメトリビューワーサブシステムの操作手順 …… 78
4．モジュール作成 …… 92
4.1　骨格部分の作成 …… 93
4.2　処理ルーチンの組み込み …… 97
5．動画の録画 …… 110
5.1　動画の録画に必要な環境 …… 110
5.2　スクリプトファイルによる動画の作成 …… 111
5.3　AVSビデオキットによる動画の作成 …… 121
おわりに …… 131
謝　　辞 …… 131
参考文献 …… 131
付録A　GWSの性能 …… 132
付録B　索引 …… 139

Contents

Introduction ····· 1
1. Operational Procedure of AVS ····· 2
1.1 Environment for Using AVS ····· 2
1.2 Starting of AVS ····· 5
1.3 Network Editor ····· 6
1.4 Reading and Writing of Network ····· 9
1.5 Making of Network ····· 15
1.6 Action of Network ····· 22
1.7 Network Editor Menu ····· 30
1.8 Examples of Networks ····· 36
2. Data Type ····· 51
2.1 Field Data ····· 53
2.2 Geometry Data ····· 67
3. Subsystem ····· 72
3.1 Starting and Closing of Subsystem ····· 72
3.2 Geometry Viewer Subsystem ····· 73
3.3 Operational Procedure of Geometry Viewer Subsystem ····· 78
4. Making of Module ····· 92
4.1 Making of Framework ····· 93
4.2 Making of Operation Routine ····· 97
5. Recording of Animation ····· 110
5.1 Environment of Animation ····· 110
5.2 Recording of Animation by Scriptfile ····· 111
5.3 Recording of Animation by AVS Video Kit ····· 121
Concluding Remarks ····· 131
Acknowledgements ····· 131
References ····· 131
Appendix A Efficiency of Graphic Work Station ····· 132
Appendix B Index ····· 139

はじめに

近年、ネットワーク技術の高度化とワークステーションやパーソナルコンピュータの高性能化により、シミュレーションや実験の結果をワークステーション上において可視化ツールやユーティリティを用いて３次元として画像を容易に表示できるようになってきた。情報システム管理課では、従来大型計算機で行なっていた画像処理やファイル編集などを、ワークステーションなどの小型の計算機へ移行させ、作業の効率化を図る分散処理環境の整備を進めており、シミュレーション結果の可視化のため画像処理サーバ、カラープリンタサーバなどの整備を行なっている。この画像処理環境の整備作業の一環として、汎用可視化ツールＡＶＳが画像処理サーバに導入されている。ＡＶＳは、シミュレーションや実験などの結果（数値データ）を、直接プログラミングすることなしにＧＵＩ操作で可視化することができるソフトウェアである。

プログラミングすることなしにＡＶＳは可視化するソフトウェアであるが、ＡＶＳを利用したことがない者には、動作環境の整備のみによって、即座にＡＶＳを縦横無尽に使いこなせず、たちどころに問題にぶつかり、時間の浪費を招くことになる。

本報告書では、分散処理環境の整備作業の一環として整備した画像処理環境のうち、可視化ソフトウェアＡＶＳの利用方法を中心に解説する。また、ＡＶＳを使った動画の作成方法、本システムのような画像処理環境を整備するための情報について述べる。

以下に本報告書の概要を記述する。ＡＶＳの基本の操作手順（ネットワークエディタ）とネットワークの作成方法について第１章で、ＡＶＳで扱うデータ（主にフィールドデータ）のフォーマットとデータをＡＶＳに取り込む方法について第２章で、サブシステムの操作手順について第３章で述べる。第４章ではモジュールの作成方法について、第５章では２通りの動画の作成方法について述べる。

1．AVSの操作手順

1.1　AVS を利用するための環境

ＡＶＳを利用するためには、次のような環境が必要です。*

1. ハードウェアに関連する環境

- ＡＶＳは可視化したイメージをＸウィンドウ上に表示します。そのため、Ｘウィンドウシステムを利用できるワークステーション（以下、EWS という）、X端末またはX-VisionWare(X-Vision) 等が利用できるパーソナルコンピュータ（以下、PC という）が必要です。X端末及び PC に関しては Ethernet 等に接続され、かつ、telnet が使える必要があります。
- （X端末、PC を利用している方及び自分の EWS にＡＶＳがインストールされていないためＡＶＳが利用できない方は、自分で整備される前に情報システム管理課が整備した画像処理サーバを利用してみて下さい。

  画像処理サーバ

  ホスト名：furies

  IP アドレス：

  133.53. 5.21 （FDDI 接続）

  133.53.10.29 （Ethernet 接続）

  ID 名：avsuser

  パスワード：画像サーバを利用したい方は、情報システム管理課に尋ねて下さい。）
- 尚、以下で利用するＸウィンドウシステムは X11R5 （X Window System Version 11 Release 5）とします。

2. ソフトウェアに関連する環境（.avsrc の設定）

ＡＶＳは、起動時にスタートアップファイル.avsrc を読み込みます。まず、以下の 1) のシステムディレクトリにあるスタートアップファイルを読み込みます。続いて 2)、3) の順にスタートアップファイルを検索します。次に、2)、3) のスタートアップファイルがある場合は、1) の内容に上書き、追加を行ないます。ＡＶＳは、この読み込んだスタートアップファイルの内容に従って起動します。

1) システムディレクトリ（/usr/avs/runtime/avsrc）
2) カレントディレクトリのスタートアップファイル（./.avsrc）
3) ホームディレクトリのスタートアップファイル（$HOME/.avsrc）

*本章から、文体として通常のソフトウェアの使用手引書に用いられている「です」「ます」調を用いた。

スタートアップファイルは、ＡＶＳに必要なディレクトリの場所、ヘルプファイルの場所、表示するウィンドウの大きさを記述したファイルです。このファイルは、ＡＶＳを効率的に利用するためには欠かせないものです。以下にその作成方法を示します。

スタートアップファイルフォーマット

スタートアップファイルを上の 2) カレントディレクトリ、即ち、自分のディレクトリに作成する例を示します。スタートアップファイルは、 vi や nemacs 等のエディタを使って作成します。作成したスタートアップファイルは、ファイル名.avsrc に保存します。

ここでは、 nemacs を使ってスタートアップファイル（.avsrc）を作成する例を示します。まず、次のようにして nemacs エディタを開いて下さい。（但し、カレントディレクトリは、 /home/guest/avsuser とします）

```
% cd    /home/guest/avsuser ←% はシェルのプロンプトを表しています。
% nemacs    .avsrc
```

nemacs を起動したら、以下のように記述します。

```
HelpPath /usr/avs_jhelp
NetworkDirectory /home/guest/avsuser/network
DataDirectory /home/guest/avsuser/data
DisplayPixmapWindow 512x512+270+20
WindowMgr twm
PrintNetwork lp -drps
```

nemacs を使う場合、ファイルの保存は、以下のように行ないます。

Ctrl （コントロールキー）と x を同時に入力し、その後に Ctrl （コントロールキー）と c を同時に入力すると

Save file /home/guest/avsuser/.avsrc? (y or n)

と聞いてくるので'y' を入力すると記述した内容をファイルに保存し nemacs エディタを終了します。

上で設定した項目の内容を以下に説明します。以下に示した以外の項目については AVS USER'S GUIDE を参照して下さい。

1) HelpPath

ヘルプファイルのあるディレクトリを指定します。例ではディレクトリとして、日本語のヘルプ /usr/avs_jhelp を指定します。

2) NetworkDirectory

ネットワークファイルの読み込み／書き込みを行なうとき等に使用するファイルブラウザが最初に検索するディレクトリを指定します。ファイルブラウザは、読み込み、書き込みに用いるファイルを選択するウィンドウです。

例では /home/guest/avsuser/network を指定します。

3) DataDirectory
read data モジュール (read field, read image, read ucd 等) で使用するファイルブラウザが最初に検索するディレクトリを指定します。
例では /home/guest/avsuser/data を指定します。

4) WindowMgr
ユーザの使用しているXウィンドウシステムのウィンドウマネージャーを指定します。使用できるのは、 awm,mwm,twm,uwm,olwm,dxwm です。例では twm を指定します。

5) DisplayPixmapWindow
可視化したイメージを表示するビューウィンドウ（ジオメトリビューワー）の大きさと表示する位置を指定します。例では、ビューウィンドウを縦 512 ピクセル、横 512 ピクセルの大きさで、ディスプレイの左上から右方向に 270 ピクセル, 下方向に 20 ピクセルの位置に表示します。

6) ModuleLibraries
モジュールパレットにロードするモジュールライブラリを指定します。
指定しないとデフォルト（/usr/avs/runtime/avsrc）のとうりに設定されます。

7) PrintNetwork
ネットワークエディタで作成したネットワークのレイアウトを出力するプリンタ名を指定します。
例では、画像処理サーバである HP9000/730 CRX-24Z のローカルプリンタを指定しています。この場合の使用しているプリンタ名は /etc/printcap に以下のように記述してあります。

```
#
# #RICOH-Script on Furies
# Print queue name            :rps
# Remote machine name         :furies
# Remote printer queue name   :rps
# Spool directory             :/usr/spool/rlp/furies
#
lp|furiesps|remote RICOH-SCRIPT:\
        :lp=:\
        :rm=furies:\
        :rp=rps:\
        :lf=/usr/spool/rlp/furies/log:\
        :sd=/usr/spool/rlp/furies:\
        :
```

1.2 ＡＶＳの起動

ＡＶＳの起動方法は、以下のとうりです。

1) Ｘウィンドウシステムを以下のコマンドで起動します。だだし、自動的にＸウィンドウシステムがセッション開始時に立ち上がる場合は、省略して下さい。

　　% xinit

2) ＡＶＳによる可視化したイメージを自分のディスプレイに表示するために環境変数 DISPLAY を現在使用しているディスプレイ番号に設定します。現在使用している端末の IP アドレスを確認してください。ディスプレイのディスプレイ番号は、

　　IP アドレス:0.0

となります。ディスプレイ番号の設定は、以下のコマンドを使用します。

　　%　setenv DISPLAY （使用しているディスプレイ番号）

例えば、IP アドレスが 133.53.10.105 であれば、以下のようにディスプレイ番号を設定します。

　　%　setenv DISPLAY 133.53.10.105:0.0

なお、ＡＶＳをネットワークを介さずに自分のＥＷＳでスタンドアロンで利用する場合は、この 2) の操作を省略して下さい。

3) avs と小文字で入力し、リターンキーを押せば指定したディスプレイ上に先ほど設定したスタートアップファイルに従ってＡＶＳを起動します。

　　% avs

ＡＶＳを起動すると、ディスプレイの左側にメインメニュー（図 1.1）を表示します。

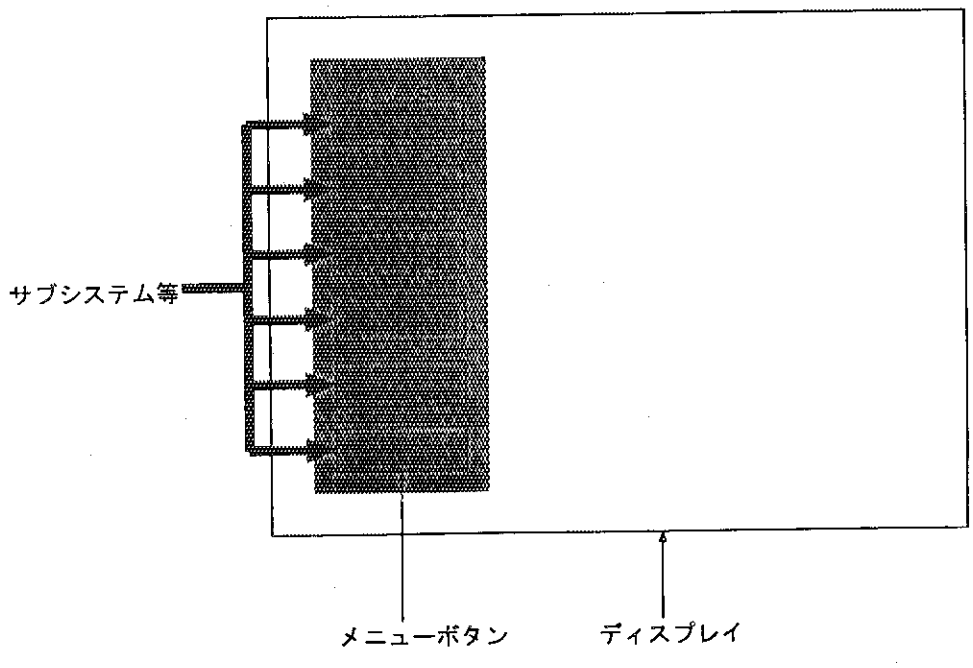

図 1.1　AVS の起動画面

メインメニューには、サブシステム等のメニューボタンがあります。これらのボタンをマウスの任意のボタンでクリックすると、以下のようなサブシステム等の機能が利用できます。

- イメージビューワーサブシステム (Image Viewer)
- グラフビューワーサブシステム (Graph Viewer)
- ジオメトリビューワーサブシステム (Geometry Viewer)
- ネットワークエディタ (Network Editor)
- AVS Application
- Exit AVS

1.3 ネットワークエディタ

1.3.1 モジュールとネットワーク

可視化において、ユーザの作成したデータを読み込んで実際に可視化したイメージを表示するまでの過程を1)～4)の行程に分けて考えます。

1) データを読み込む。(Data Input)

2) データを別のデータに変換する。情報を加えたり、減らしたりする。(Filter)

3) 表示用のデータに変換する。(Mapper)

4) データを表示する。(Data Output)

ＡＶＳでは、上にあげた可視化までの１つ１つの行程をモジュールとよばれる部品を組み合わせることによって表現します。モジュールは、上にあげた４つの行程のいずれかの処理を行ないます。

ＡＶＳでは、この可視化の一連の作業をモジュールの組み合わせで表現したものをネットワークといいます。モジュールを組み合わせてネットワークを作成するための道具としてネットワークエディタを用います。

1.3.2 ネットワークエディタの機能

a) ネットワークの保存、読み込み

あらかじめ作成しておいたネットワークをディスクからファイルとして読み込むことができます。また、現在作成しているネットワークをファイルとして保存することができます。

b) ネットワークの作成

ネットワークエディタのワークスペース (図1.2) と呼ばれる部分を使って、モジュールを組み合わせてネットワークを作成することができます。

c) ネットワークの実行
ネットワークにデータを読み込ませて実行することができます。

d) モジュールの作成
ＡＶＳのモジュールの中に処理させたい適当な機能をもつモジュールがない場合に、ユーザは、独自にモジュールを作成することができます。Module Generator モジュールが、モジュール作成の手助けをしてくれます。ただし、ＣまたはFORTRAN のプログラミングの知識が必要です。

1.3.3 ネットワークエディタの起動

メインメニュー（図1.1）の中にある Network Editor ボタンにカーソルを移動してクリックするとネットワークエディタが起動します。ネットワークエディタを起動すると、図1.2に示すように、
画面左側に ’ネットワークコントロールパネル’
画面右側に ’ネットワークコンストラクションウィンドウ’
を表示し、モジュールパレットにモジュールを表示します。ネットワークは、ワークスペース上に作成します。

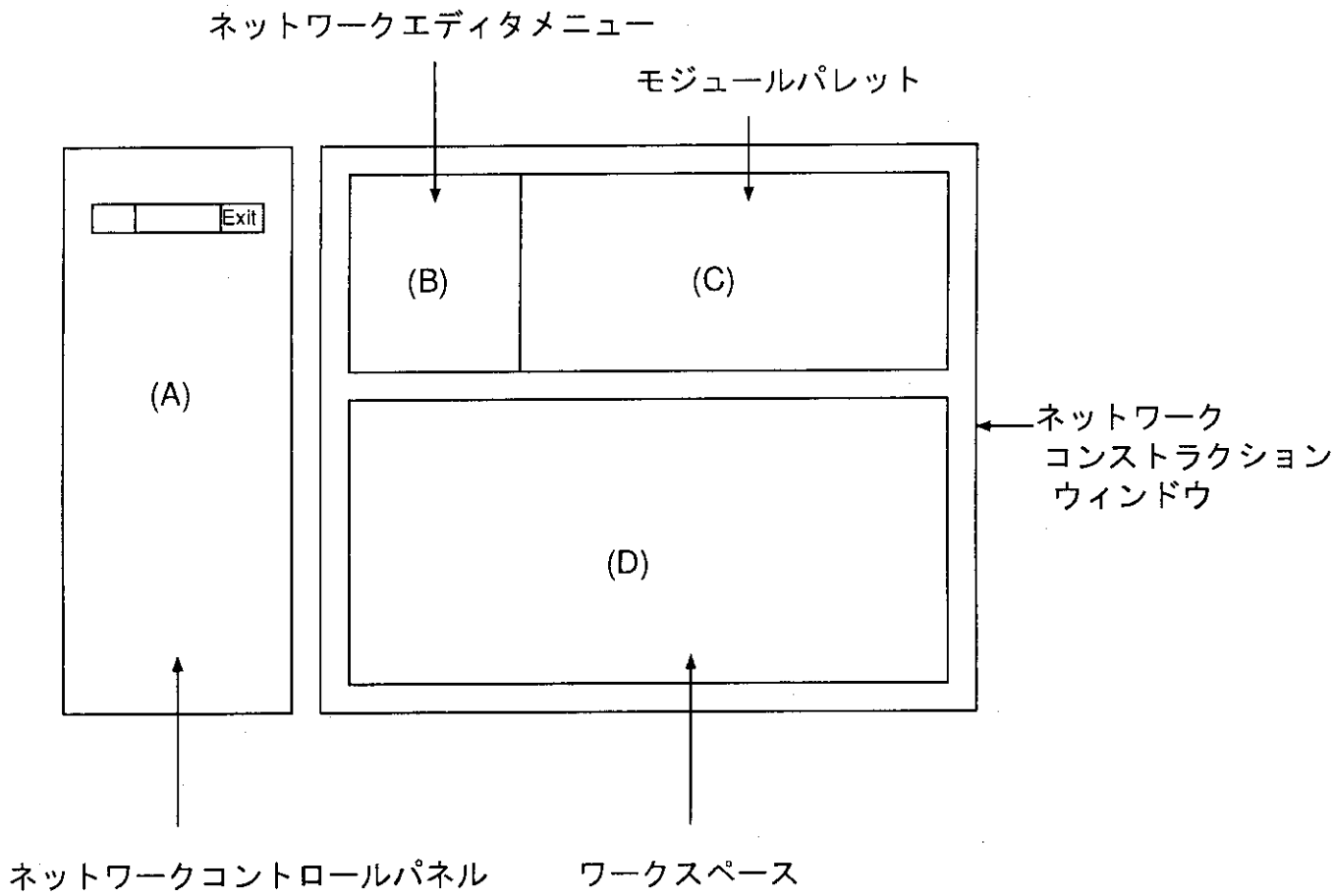

図 1.2 ネットワークエディタ

1.3.4 ネットワークエディタの終了

1. ワークスペースにネットワークがない場合の終了方法

   1) ネットワークコントロールパネルの一番上にある Exit ボタン（図 1.2）をクリックすると、メインメニュー（図 1.1）に戻ります。

   2) メインメニューの Exit AVS ボタンをクリックすると、図 1.3に示すようなウィンドウを表示します。

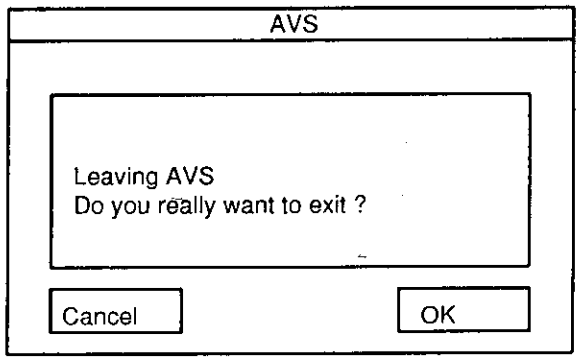

図 1.3 ＡＶＳの終了

   3) OK ボタン（図 1.3）をクリックすると、ＡＶＳを終了し、ＡＶＳを起動したウィンドウに AVS Exited とメッセージを出力します。

2. ワークスペース上にネットワークがある場合の終了方法

   1) ネットワークコントロールパネルの一番上にある Exit ボタン（図 1.2）をクリックします。

   2) ワークスペース上にモジュールまたはネットワークがあるので、終了していいかを図 1.4のように尋ねてきます。

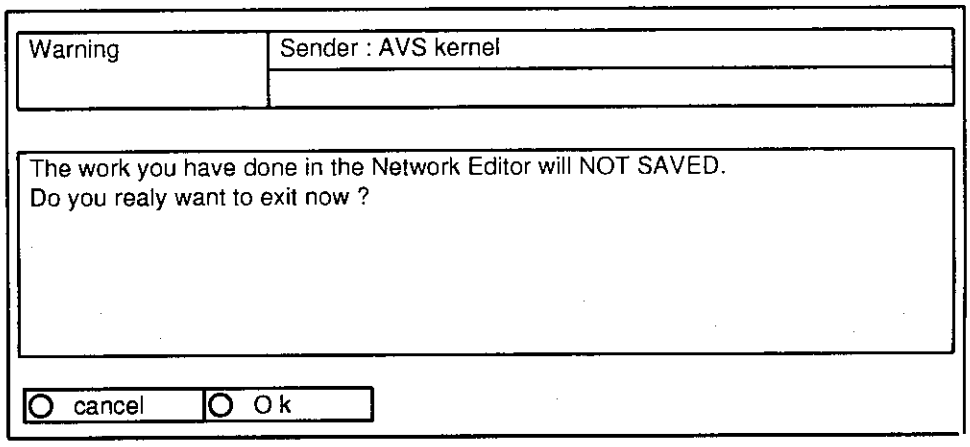

図 1.4 ネットワークエディタの終了

3) 終了しないときは、cancel ボタンをクリックします。この操作により、ネットワークエディタに戻ることができます。

4) 終了するとき（このとき、ＡＶＳは、ネットワークを保存しません）は、Ok をクリックして下さい。ネットワークエディタを抜けてメインメニューに戻ります。「１．ワークスペースにネットワークがない場合の終了方法」の 2) 以降の手順を実行すれば、ＡＶＳは終了します。

ネットワークコンストラクションウィンドウの閉じ方及び開き方

ネットワークエディタメニュー（図 1.2の (B) の部分）の一番上にある Close ボタンをクリックすると現在作業しているものはそのままの状態で、ネットワークコンストラクションウィンドウ（図 1.2の (B)、(C)、及び (D) の部分）を閉じます。ネットワークコンストラクションウィンドウを再び開くためには、ネットワークコントロールパネル（図 1.2の (A) の部分）の一番上にある Display Network Editor ボタンをクリックします。この機能によりウィンドウを重ねることなしに可視化したイメージを表示できます。

1.4　ネットワークの読み込み、保存

ネットワークの読み込みや保存は、ネットワークエディタメニューの 1 つ Network Tools メニューを使います。Network Tools ボタン（図 1.5）をクリックすると、以下に示すような Network Tools メニューを表示します。

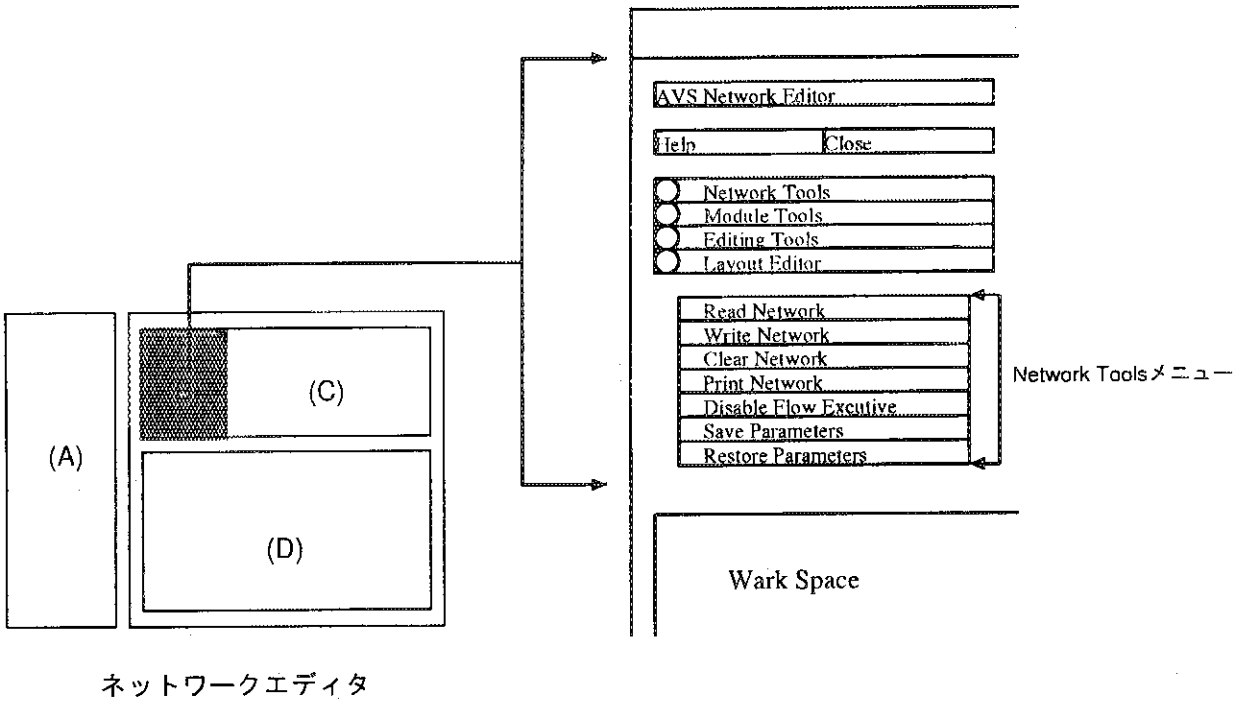

図 1.5　ネットワークエディタメニュー Network Tools

- Read Network
- Write Network
- Clear Network
- Print Network
- Disable Flow Excutive
- Save Parameters
- Restore Parameters

(1) ネットワークの読み込み ;Read Network

1. ワークスペースにネットワークがない場合
   以下のようにしてネットワークを読み込みます。

   1) 図1.5の(B)に表示されている Read Network ボタンをクリックするとＡＶＳは、図1.6に示すファイルブラウザを表示します。このファイルブラウザを使いディスク上のネットワークを保存しているファイルを選択します。
   2) ファイルブラウザでは、ディレクトリ名は赤文字、ファイル名は黒文字で表示します。
   3) ファイルブラウザの右側にあるスクロールバーにカーソルを移動します。ファイルブラウザはマウスの右ボタンをクリックすると下へスクロールし、左ボタンをクリックすると上にスクロールします。またディレクトリを移動したいときにはファイルブラウザに黒文字で表示されているディレクトリ名へカーソルを移動しクリックします。

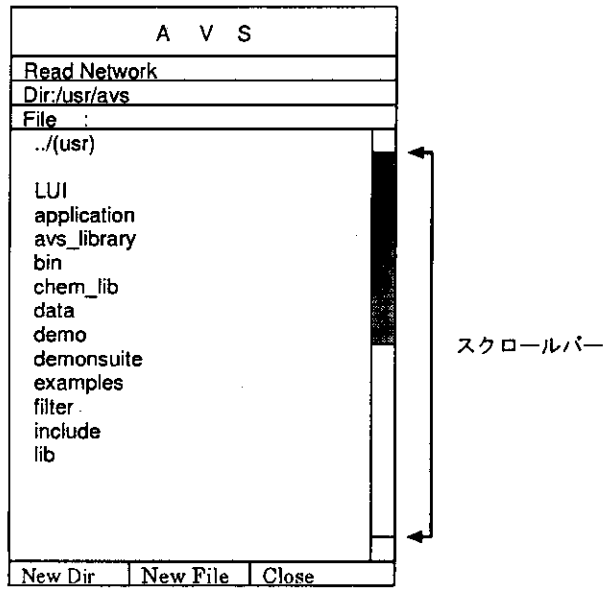

図 1.6 ファイルブラウザ

   4) 読み込むネットワークを保存したファイル（ネットワークファイルという）を見つけたら、カーソルをそのファイル名に移動しクリックします。ここでは、/home/guest/avsuser, をクリックします。

2. ワークスペースにネットワークがある場合

Read Network ボタンをクリックすると、既にワークスペース上にネットワークがある場合は、以下のようなメッセージを表示します。

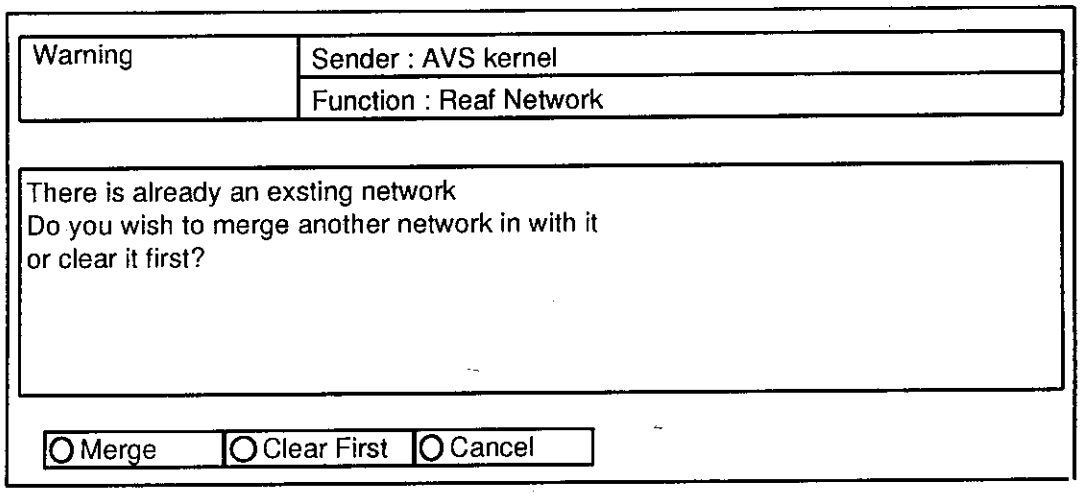

図 1.7 Read Network メッセージ

- Merge ボタン

  ワークスペース上のネットワークを残したままでネットワークを読み込む場合にクリックします。

- Clear First ボタン

  ワークスペース上のネットワークを消去してから新たにネットワークを読み込む場合にクリックします。

- cancel ボタン

  ネットワークの読み込みを中止したい場合にクリックします。

ネットワークファイルを読み込むとＡＶＳは、ワークスペース上にネットワーク（図1.8）を表示します。引き続き、ビューウィンドウ（ジオメトリビューワー）に可視化したイメージ（図1.9）を表示します。

ネットワーク上の長方形で表示しているアイコンをモジュールといいます。

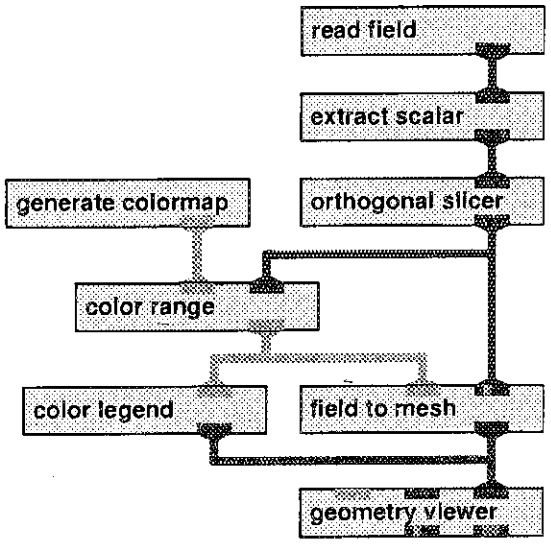

図 1.8　ネットワーク read_wata_mesh.net

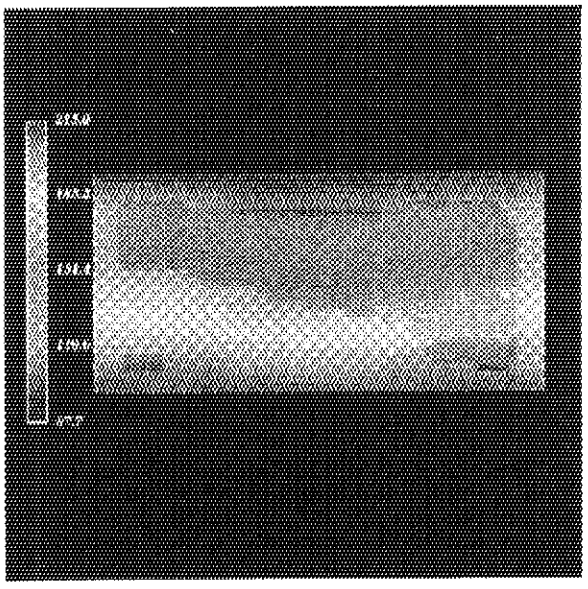

図 1.9　実行結果

(2) ネットワークの保存 ;Write Network

1. 新しいファイルとして保存する方法

   1) ネットワークエディタメニュー（図1.5）にあるWrite Networkボタンをクリックすると、ＡＶＳは、ファイルブラウザを表示します。

   2) ネットワークの読み込みのときと同じように、マウスを使って、保存したいディレクトリを見つけて下さい。

   3) ファイルブラウザの下にあるNew Fileをクリックすると以下のようなウィンドウを表示します。このウィンドウ内にカーソルを移動して新しいファイル名を入力します。OKをクリックするとＡＶＳは、ネットワークをファイルに保存します。またネットワークの保存を中止したい場合はcancelをクリックします。

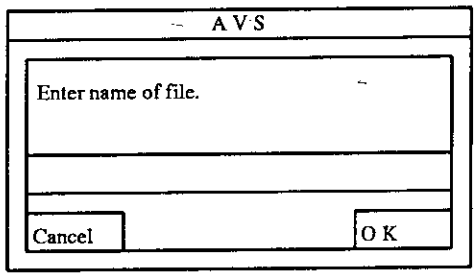

図 1.10　ファイル名の入力

2. 読み込んだファイルを更新して保存

   1) ネットワークエディタメニュー（図1.5）にあるWrite Networkをクリックします。

   2) ＡＶＳは、ファイルに更新した内容を上書きしていいかを図1.11のようにたずねてきますので、そのまま上書きしていいときは、Okをクリックします。別のファイルに保存したい時はファイルブラウザのNew Fileをクリックし、「1. 新しいファイルとして保存する方法」の3)の手順を実行します。

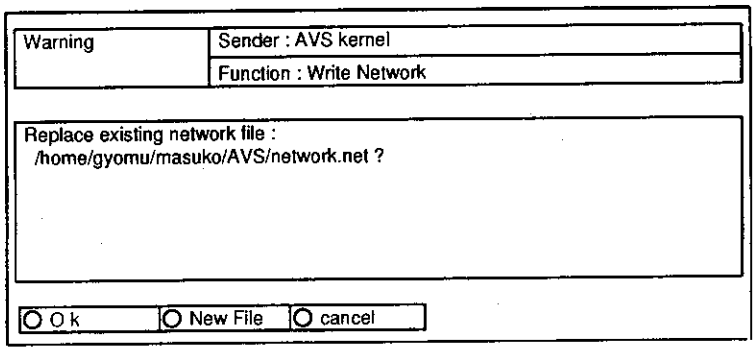

図 1.11　ネットワークの保存

Write Network の保存情報

Write Network では、以下の情報を保存します。

- ワークスペース上にある全てのモジュールとモジュール間の接続情報
- 初期設定値より変更されたモジュールのパラメータの最新値
- 現在の画面上のレイアウトの状況（ウィンドウの位置など）

(3) ワークスペース上の全てのモジュールの消去 ;Clear Network

Network Tools メニューの Clear Network ボタン（図 1.5）をマウスでクリックするとワークスペース上の全てのモジュールを消去します。

(4) ネットワークのレイアウトのプリントアウト ;Print Network

ワークスペース上にあるネットワークのレイアウトをプリントアウトする方法を以下に示します。

1) Print Network ボタンをマウスでクリックします。

2) ＡＶＳは Print Network ボタンをクリックした時点でネットワークのレイアウトをポストスクリプトファイル /tmp/AVSnet＊＊＊.ps に保存します。なお、＊＊＊は、自動的にＡＶＳがキャラクタ文字を設定します。

3) また、ＡＶＳはプリンタに出力するかどうかを以下のように尋ねてきます。OK をマウスでクリックするとスタートアップファイルに記述しているプリンタに出力します。Cancel をクリックした場合は、ファイルには保存しますがプリンタには出力されません。

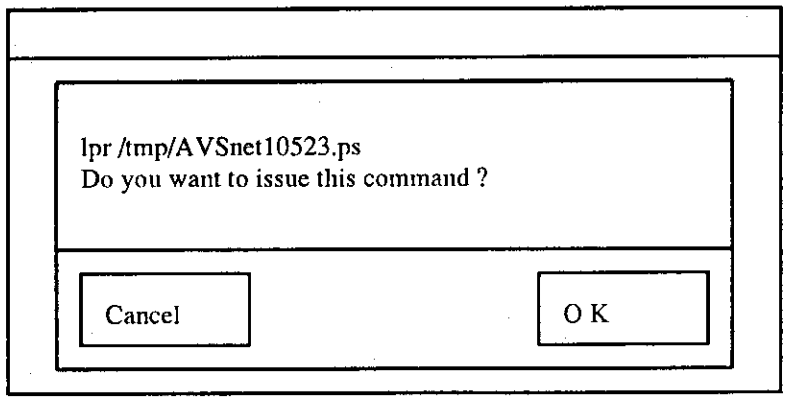

図 1.12 ネットワークのプリントアウト

(5) ネットワークの実行の停止 ;Disable Flow Excutive

Network Tools メニューの Disable Flow Excutive ボタン（図1.5）をクリックすると、ネットワークの実行が停止します。

停止したネットワークを再度実行させるには、Disable Flow Excutive ボタンをもう一度クリックします。

(6) カレントパラメータ値の保存 ;Save Parameters

ワークスペース上にあるネットワークを構成しているモジュール（カレントモジュール）のパラメータをパラメータファイル /tmp/AVSsnp.＊に保存します。

(7) ネットワークパラメータのリセット ;Restore Parameters

ワークスペース上にあるネットワークのパラメータを最後に保存したパラメータにリセットします。

1.5 ネットワークの作成

1.5.1 モジュールアイコン

各モジュール（図1.13参照）を表現したアイコンには、入力ポート、出力ポート、モジュール名が表示されています。

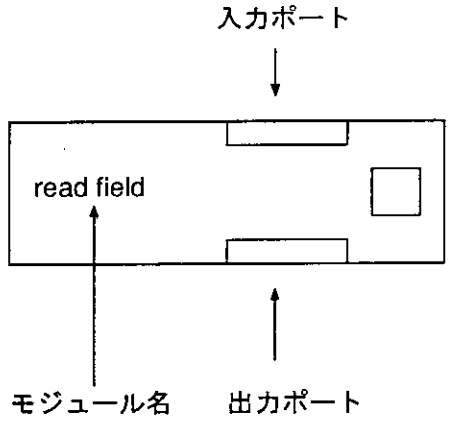

図 1.13 モジュール

モジュールは、入力ポートからデータを受けとりモジュールの中で何らかの処理をし、出力ポートから下流のモジュールにデータを渡します。

入力ポートはモジュールが受け入れ可能なデータのタイプ、出力ポートはモジュールが出力するデータのタイプを色で区別しています。モジュールが扱うことのできるデータはＡＶＳで定めたタイプでなければなりません。

ＡＶＳで扱えるデータには、以下のようなものがあります。（データについては、第２章で説明しています。）

- フィールドデータ
- ＵＣＤデータ
- ジオメトリデータ
- 整数
- 浮動小数点
- 文字列

ポートの色とデータタイプ

ポートの色とデータタイプは、以下のように対応します。

表 1.1　ポートの色とデータタイプの対応

ポートの色	データタイプ
複数のカラー	フィールドデータ
赤	ジオメトリデータ
オレンジ	ＵＣＤデータ
黄	カラーマップデータ
明紫色	整数
暗紫色	浮動小数点
青緑	文字列

ラインの色とデータタイプ

また、モジュールのポート間をつないでいるラインの色は、モジュール間で受渡すデータのタイプを示します。ラインの色とデータタイプは以下のように対応します。

表 1.2　ラインの色とデータタイプの対応

ラインの色	データタイプ
青	フィールドデータ
黄	カラーマップ
オレンジ	ＵＣＤデータ
赤	ジオメトリデータ
明紫色	整数
暗紫色	浮動小数点
青緑	文字列

モジュールの中には、入力ポートをもたないものや出力ポートをもたないものがあります。このようなモジュールは、データをモジュールの中で作成するか、ファイル等からデータを読み込んだりファイル等へ保存したりします。

また、モジュールには、取り込んだデータをどのように処理するのかを決めるパラメータがあります。パラメータは、コントロールウィジェットを使ってモジュールの中で与えたり、パラメータポート（パラメータの値を受けとる入力ポート）を通して違うモジュールから与えたりします。パラメータを調整することにより、回転の角度を変えたり、イメージを拡大したり、色をつける方法を変えたり、読み込むファイルを変更したりすることができます。

1.5.2 モジュールの扱い

(1) モジュールライブラリとモジュールパレット

スクリーン右側のネットワークコンストラクションウィンドウ上（図1.2）にある、AVS Module Library（図1.14）には、ＡＶＳが読み込んだモジュールライブラリ名を表示します。モジュールライブラリは、モジュール名とモジュールのカテゴリーを記述したファイルです。

また、AVS Module Library の下のモジュールパレット（図1.14）には、左から順にカテゴリー別にモジュールが並びます。以下にカテゴリの内容を示します。

- Data Input（データ入力モジュール）
  データをＡＶＳに取り込むためのモジュール
- Filter（フィルターモジュール）
  データを別のデータに変換するモジュール
- Mappers（マッパーモジュール）
  データを表示用の幾何データ（ジオメトリデータ）に変換するモジュール
- Data Output（データ出力モジュール）
  表示や出力を行なうモジュール

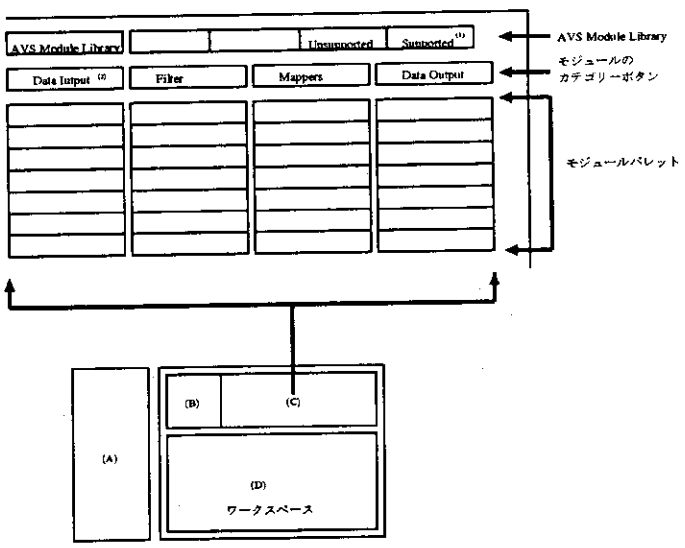

図 1.14 モジュールパレット

(2) ネットワークの作成手順

ＡＶＳでは、ワークスペース（図1.14の(D)）でモジュールを組み合わせてネットワークを作成します。以下にネットワーク作成手順の概要を示します。

1) ネットワーク作成に必要なモジュールをモジュールパレットで検索する。

2) モジュールをモジュールパレットからワークスペースに移動します。

3) モジュールを接続する。

(3) モジュールの検索

以下にネットワークの作成に必要なモジュール（例として read field モジュール）をモジュールパレットで検索する方法を説明します。read field モジュールは、Data Input のカテゴリにあります。

1) モジュールのカテゴリーボタンの中の Data Input ボタン（図1.14の(2)）にマウスのカーソルを移動します。

2) モジュールの検索を以下のようにして行ないます。モジュールの検索の方法は a), b) の2通りあります。

   a) Data Input ボタン（図1.14）の上にカーソルを移動します。マウスの右ボタンをクリックするとモジュールパレットが下へスクロールし、左ボタンをクリックするとモジュールパレットが上へスクロールします。この操作により、read field モジュールを検索します。

   b) 検索したいモジュールがわかっている、または、そのモジュールの名前の一部分がわかっているときには、Data Input ボタンの上にカーソルを移動し、その文字をキーボードから入力します。ここでは、read と入力して下さい。ＡＶＳは、Data Input ボタンの位置に入力した文字を表示し、モジュールの検索を開始します。検索を終了するとモジュールパレットの先頭に検索したモジュールを表示します。また、リターンキーを入力する毎に次候補を検索していきます。

(3) モジュールの移動

検索したモジュールをモジュールパレットからワークスペースに移動する方法を説明します。

1) モジュールパレットにあるモジュールにカーソルを移動します。

2) 任意のボタンをクリックしたままカーソルをワークスペースに移動します。

3) ワークスペースでマウスのボタンを離すと、マウスを離した場所にモジュールを表示します。また、ネットワークコントロールパネルにモジュールのコントロールウィジェット（図1.15）を表示します。

   複数のモジュールがワークスペースにある場合は、ネットワークコントロールパネルのモジュール名の書いてあるボタンをクリックするとネットワークコントロールパネルにモジュールのコントロールウィジェットを表示します。

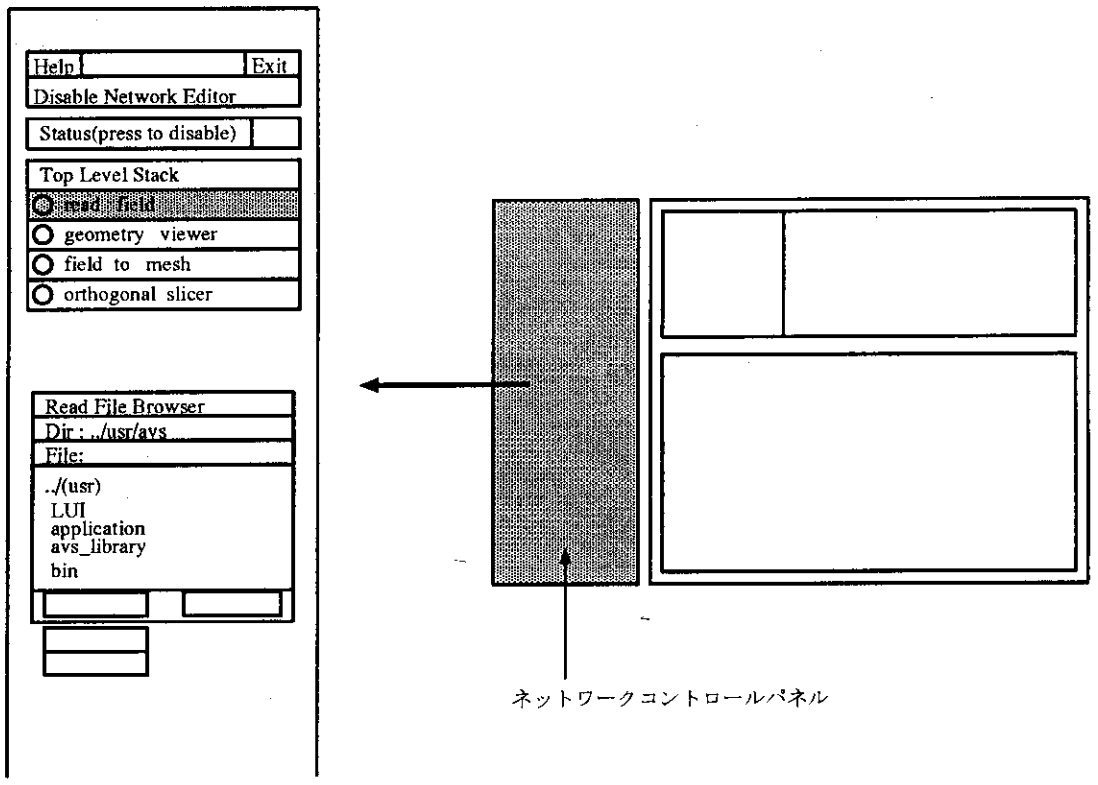

図 1.15 ネットワークコントロールパネル

以下に、例題として使用するモジュールを示します。モジュール名の右側のカッコ内は、そのモジュールのあるカテゴリを示します。以下のモジュールを read field モジュールをワークスペースに移動した方法でワークスペースに移動してください。

- read field モジュール (Data Input)
- geometry viewer モジュール (Data Output)
- orthogonal slicer モジュール (Mapper)
- field to mesh モジュール (Mapper)
- generate colormap モジュール (Data Input)
- color range モジュール (Data Input)
- color legend モジュール (Mapper)

(4) モジュールの消去

ワークスペースで必要でないモジュールは、以下の手順でワークスペース上から消去することができます。

1) ワークスペースの消去したいモジュールにカーソルを移動します。
2) マウスの左ボタンをクリックしたままワークスペースの右下にあるハンマー（図1.16）までカーソルを移動します。
3) ハンマーの上で、マウスの左ボタンを離すと、ＡＶＳはモジュールを消去します。

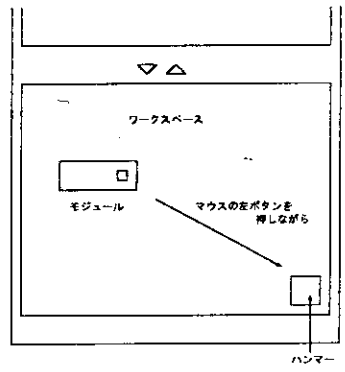

図 1.16　モジュールの消去

(5) モジュールの移動（ワークスペース）

ワークスペース上でモジュールを移動する方法を示します。モジュールの上にカーソルを移動します。マウスの左ボタンをクリックしたままカーソルを移動してワークスペースの任意の場所でマウスのボタンを離します。モジュールはマウスのボタンを離した場所に移動します。

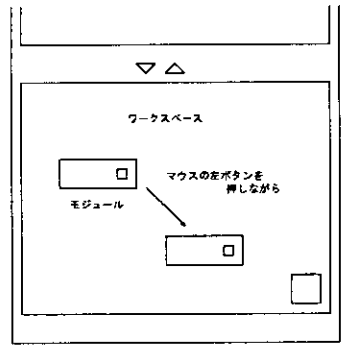

図 1.17　モジュールの移動（ワークスペース）

(6) モジュール間の接続、切断

モジュールをワークスペース上に移動したら、モジュール間の接続を行ないます。モジュールとモジュールを接続するためには、出力側のモジュールの出力ポートと入力側のモジュールの入力ポートの色が同じでなければいけません。

1. モジュール間の接続

   ワークスペース上に取り出したモジュール間の接続の方法を read field モジュールと orthogonal slicer モジュールを例に説明します。

   1) ワークスペースの read field モジュールの出力ポートの上にカーソルを移動し、マウスの中ボタンをクリックしたままにします。
   2) ＡＶＳは read field モジュールがどのモジュールと接続が可能かを判断し、それを示すラインを表示します。
   3) さらにクリックしたまま orthogonal slicer モジュールの方にカーソルを動かすと、ラインが白くなります。
   4) 接続したいラインが白くなっている状態でマウスのボタンを離すと接続が行なわれます。また、マウスを離さずに read field モジュールまでカーソルを戻し、白いラインが消えた状態でボタンを離すと、モジュールの接続を回避することができます。

2. モジュール間の切断

   モジュール間の切断を read field モジュールと orthogonal slicer モジュールを例に説明します。

   1) read field モジュールの出力ポートの上にカーソルを移動してマウスの右ボタンをクリックしたままにします。
   2) クリックしたまま切断したいモジュールの方へカーソルを動かすとラインが白くなります。
   3) ラインが白くなった状態でボタンを離すとＡＶＳは、モジュール間を切断します。

以上の手順によってモジュールを組み合わせて以下のようなネットワークを作成してください。

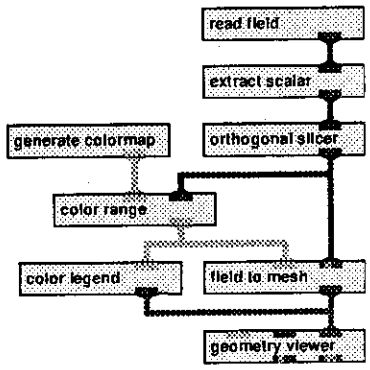

図 1.18 ネットワーク read_wata_mesh.net

1.6 ネットワークの実行

read field モジュール及び orthogonal slicer モジュールを使ってのネットワークの実行例

ネットワーク read_wata_mesh.net は以下のようにして実行します。

1) ネットワークコントロールパネルの read field ボタン（図 1.15）をクリックするとファイルブラウザ（図 1.19）を表示します。

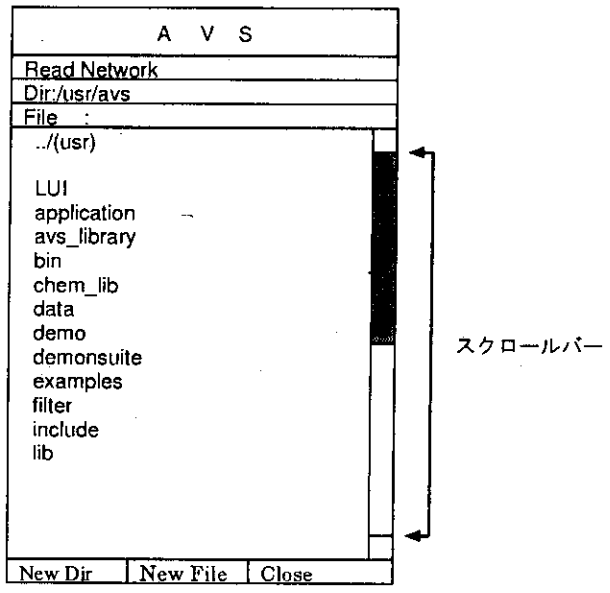

図 1.19 ファイルブラウザ

2) ファイルブラウザを使って、データを保存しているファイルを選択します。この例では、
/home/guest/avsuser/read_wata.fld
を選択します。

3) read field モジュールは、2) でクリックしたファイルを読み込みます。

4) ネットワークは順次データを下流のモジュールへ送ります。最後に可視化イメージを表示します。

ただし、パラメータを正しく設定しないと可視化したイメージをうまく表示できません。また、モジュール間のデータの受渡しの際に、受け入れ不可能なデータを受渡ししようとした場合には、以下のようなエラーメッセージを表示します。

例 Recieved imcompatible data type in module 3D bar chart;.user.0-Field Input

1.6.1 パラメータ

パラメータは、モジュールが取り込んだデータをどのように処理するのかを決定します。パラメータを設定する方法は、以下の2通りあります。

- コントロールウィジェットを使ってパラメータを設定する方法
- 他のモジュールからパラメータポートを通して設定する方法（通常、入力側のパラメータポートは表示していません）

1.6.2 コントロールウィジェットによるパラメータの設定方法

まず、コントロールウィジェットを使ってモジュールのパラメータを設定する方法を説明します。モジュールは、複数のパラメータをもつことができます。パラメータは、コントロールウィジェットを使って決定します。

orthogonal slicer モジュールのコントロールウィジェット

orthogonal slicer モジュールは、座標軸に垂直な平面で3次元のデータから2次元のデータを抽出するモジュールです。この orthogonal slicer モジュールのパラメータを設定するコントロールウィジェットについて説明します。

orthogonal slicer モジュールには、パラメータ slice plane とパラメータ axis があります。パラメータ slice plane はダイヤルウィジェットと呼ばれるコントロールウィジェットで設定し、パラメータ axis はラジオボタンウィジェットと呼ばれるコントロールウィジェットで設定します。パラメータを設定するには、まず、ネットワークコントロールパネルにコントロールウィジェットを表示させます。

コントロールウィジェットを表示させるには、ネットワークコントロールパネルに表示しているモジュール名 (orthogonal slicer) の書いてあるボタン（図 1.15）をクリックします。

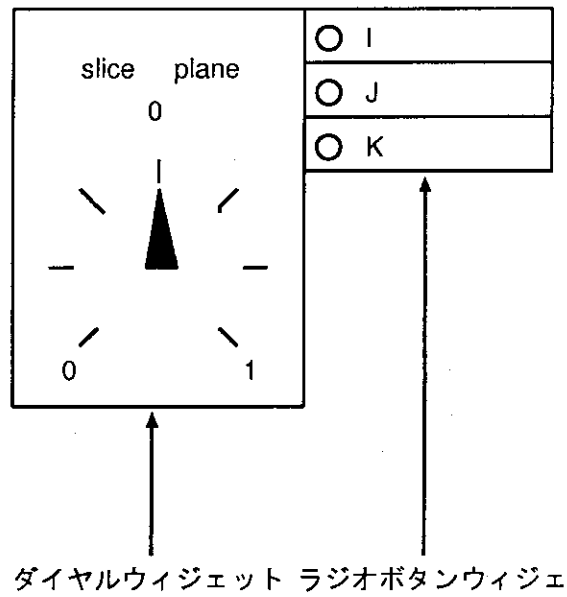

図 1.20 ダイヤルウィジェット、ラジオボタンウィジェット

(1) ダイヤルウィジェット

パラメータ slice plane は、3次元のデータからどの位置の断面（XY 平面などの2次元のデータ）を抽出するのか設定します。このパラメータの設定には、ダイヤルウィジェットを使います。ダイヤルウィジェットでパラメータを設定するには以下の2通りの方法があります。

1) ダイヤルの外周付近（メモリの打ってある場所）の任意の場所にカーソルを移動して、マウスの任意のボタンをクリックします。針はクリックした場所へ移動しその値にパラメータを設定します。orthogonal slicer モジュールは、変更したパラメータに合わせて断面を抽出します。

2) ダイヤルの針にマウスのカーソルを移動します。マウスのボタンをクリックしたままでカーソルを移動すると針が移動します。任意の場所でボタンを離します。このとき針の指している値がパラメータとなります。

(2) ラジオボタンウィジェット

パラメータ axis は、x,y,z のどの軸に垂直な断面（yz, zx, xy 平面）のデータを抽出するのかを設定します。このパラメータの設定には、ラジオボタンウィジェットを使います。ラジオボタンウィジェットは、1つのスイッチをオンにするとその他のスイッチはオフになるという排他的な選択のできるコントロールウィジェットです。以下にラジオボタンウィジェットでパラメータを設定する方法を示します。

1) カーソルを選択したいボタンへ移動します。

2) マウスをクリックします。

3) クリックしたボタンのみが白くなり、選択した軸に垂直な断面を抽出します。

コントロールウィジェットいろいろ

モジュールのパラメータには様々なタイプがあります。様々なパラメータを設定するために、ＡＶＳでは多様なコントロールウィジェットが用意されています。

- タイプインウィジェット

文字や数値をキー入力して、パラメータを設定します。タイプインウィジェットに入力する方法を以下に示します。

1) マウスのカーソルを入力したいタイプインエリアまで移動します。

2) タイプインエリア（図 1.21）に文字または数値を入力します。

3) 入力し終ったら、改行キーを押すか、マウスのカーソルをタイプインエリアの外へ動かすことでパラメータが決定します。

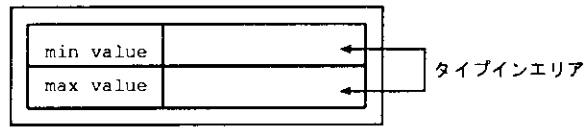

図 1.21 タイプインウィジェット

- スライダーウィジェット
  ダイヤルウィジェットと同じように、任意の場所でクリックするか、ボタンをクリックしながら針を移動するかの方法でパラメータを設定します。

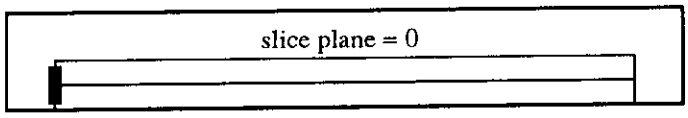

図 1.22 スライダーウィジェット

- トグルウィジェット
  マウスでクリックする度にオンとオフを繰り返します。オンになるとボタンが白くなります。

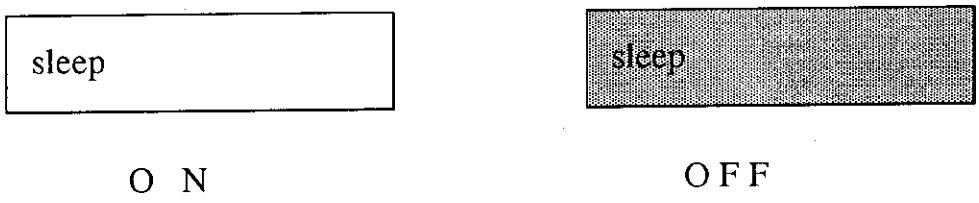

図 1.23 トグルウィジェット

- ファイルブラウザウィジェット
  Read File Browser のようなファイルを選択するウィジェットで、黒はファイル名、赤はディレクトリ名を示しています。使用方法については、「1.4 ネットワークの読み込み、保存」を参照して下さい。

- テキストブラウザウィジェット
  テキストブラウザは、テキストファイルを表示するためのファイルブラウザウィジェットです。Show Module Documentation ウィンドウ（図 1.27）がこの一例です。

パラメータ変更に用いるコントロールウィジェットの選択

パラメータによっては、変更に用いるコントロールウィジェットのタイプを選択することができます。この方法を以下に示します。

1) パラメータを変更したいモジュール内の右側にある正方形のボタンをマウスの右ボタンまたは中ボタンでクリックするとモジュールエディタウィンドウ（図 1.24）を表示します。ここでは read field モジュール内の右側にある正方形のボタンをクリックします。この操作はパラメータの変更のために用いるコントロールウィジェットのタイプを選択するために用いるパラメータエディタウィンドウを開くために行ないます。

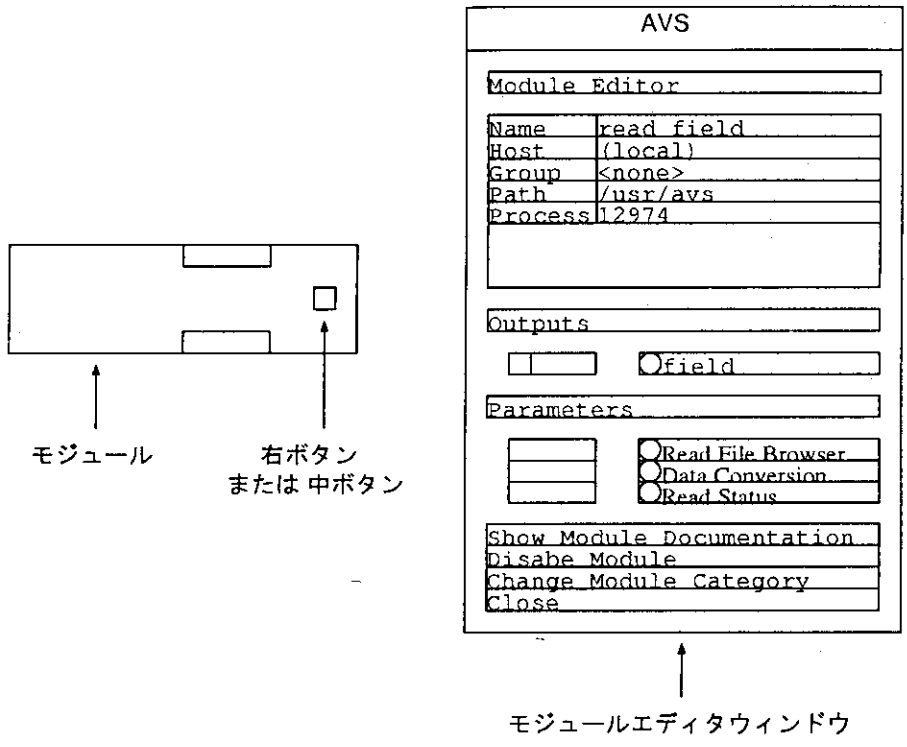

図 1.24 モジュールとモジュールエディタウィンドウ

2) モジュールエディタウィンドウ（図1.24）の変更したいパラメータ名の表示されているボタンをクリックするとパラメータエディタウィンドウ（図1.25）を表示します。ここではRead File Browser ボタンをクリックします。

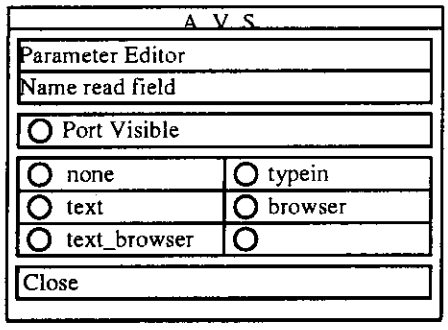

図 1.25 パラメータエディタウィンドウ

3) コントロールウィジェット名の表示されている任意のボタンをクリックするとネットワークコントロールパネルのコントロールウィジェットが変更されます。read field モジュールの場合は、typein ボタン, text ボタン, browser ボタン, text_browser ボタンの中から選択します。

4) モジュールエディタウィンドウの Close ボタンをクリックするとパラメータエディタウィンドウ、モジュールエディタウィンドウを閉じます。

1.6.3 他のモジュールからパラメータの設定方法

ここでは、他のモジュールからパラメータを設定する方法を説明します。

1) ワークスペースに read field モジュールと label モジュールを移動します。それぞれのモジュールに対して 2) から 5) の操作を行ないます。

2) モジュールの右側の正方形のボタンをマウスの右または中ボタンでクリックすると、モジュールエディタウィンドウ（図 1.24）を表示します。

3) モジュールエディタウィンドウの read field モジュールの場合は Read File Browser ボタンを、label モジュールの場合は Title String ボタンをクリックするとパラメータエディタウィンドウを表示します。

4) パラメータエディタウィンドウ（図 1.25）の Port Visible ボタンをクリックすると、Port Visible ボタンが白くなり、さらにワークスペースのモジュールに新しく入力ポートを表示します。この入力ポートを特にパラメータポートといいます。read field モジュールは、緑のパラメータポートを表示します。

5) モジュールエディタウィンドウの一番下の Close ボタンをクリックするとモジュールエディタウィンドウとパラメータエディタウィンドウを閉じます。

6 file browser モジュールをワークスペースに移動します。file browser モジュールはファイル名を出力するモジュールです。

7) 以下にパラメータポートを使ったネットワーク例を示します。例では、file browser モジュールから read field モジュールと label モジュールにファイル名を同時に出力することができます。

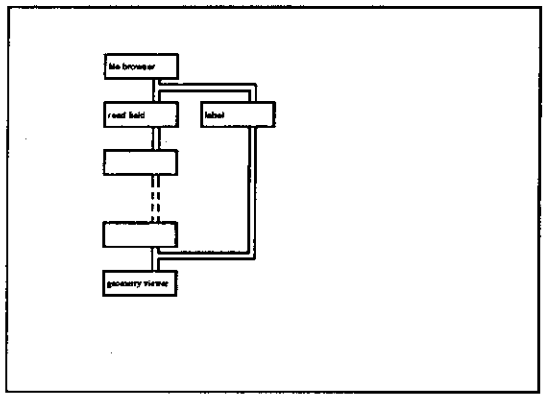

図 1.26 パラメータポート

なお、モジュールエディタウィンドウには、以下の機能をもつボタンがあります。

- Show Module Documentation

  モジュールに対するヘルプ（解説）を表示します。日本語ヘルプをスタートアップファイルで指定している場合は、モジュールエディタウィンドウの Show Module Documentation ボタンをクリックすると以下のような Show Module Documentation ウィンドウを（図 1.27）表示します。このウィンドウには、モジュールに対する以下のような説明を記述しています。

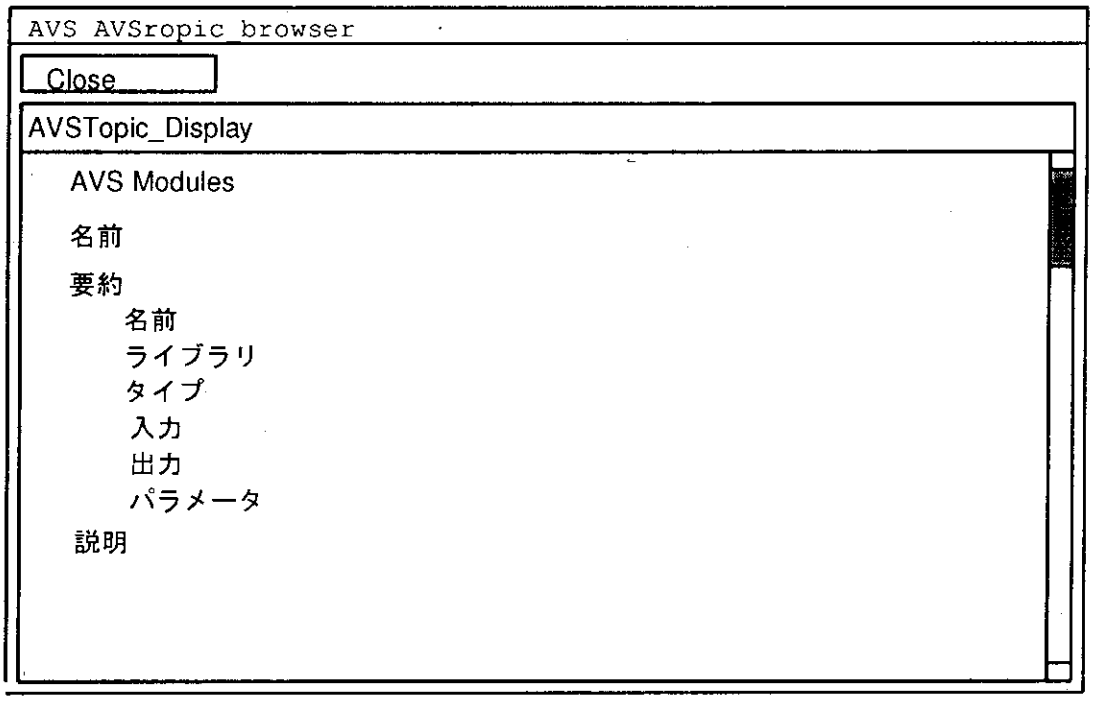

図 1.27 Show Module Documentation ウィンドウ

図 1.27に示される項目は以下の通りです。

名前	モジュール名
ライブラリ	モジュールのあるモジュールライブラリ
タイプ	モジュールのカテゴリ
入力	入力可能なデータタイプ
出力	出力可能なデータタイプ
パラメータ	パラメータ名、コントロールウィジェットのタイプ、初期値・最小値・最大値
説明	モジュールの機能の説明、パラメータの説明、ネットワーク例

- Disable Module

  モジュールエディタウィンドウの Disable Module ボタン（図 1.24）をクリックすると、モジュールの実行を禁止してデータの入出力を停止します。停止しているモジュールのアイコンは赤い表示となります。もう一度クリックすると使用可能になります。

1.6.4 ネットワークの実行をコントロールする Disable Flow Executive

ＡＶＳは、１つの入力データまたはパラメータが変更されるたびにネットワークを実行するので対話的に可視化を行います。

しかし、入力データやパラメータを変えるたびにネットワークを実行するのではなく、入力データやパラメータを全て変更してからネットワークを実行したい場合は、Network Tools メニューの Disable Flow Excutive ボタン（図 1.28）をクリックすると Disable Flow Excutive ボタンが白くなり、再び Disable Flow Excutive ボタンをクリックするまでネットワークの実行を行いません。

1.6.5 モジュールの異常終了と再実行

ネットワークの実行中にモジュールが異常終了すると、モジュールは反転して異常を知らせます。そして、メッセージパネルに原因を表示します。

メッセージパネルの accept ボタンをクリックすると該当するモジュールは、反転したまま（異常終了したまま）になります。ここで異常終了したモジュールをハンマー（図 1.16）まで移動し消去します。再度、モジュールパレットからモジュールを移動してきてネットワークに接続して下さい。

メッセージパネルの restart ボタンをクリックし、次に異常終了の原因となったパラメータを修正するとネットワークが再実行されます。

1.7 ネットワークエディタメニュー

以下に示すネットワークエディタメニューは、対応するサブメニューをそれぞれもっています。

- Netowork Tools メニュー
  「1.4 ネットワークの読み込み、保存を参照して下さい。」
- Module Tools メニュー
- Editing Tools メニュー
- Layout Editor メニュー

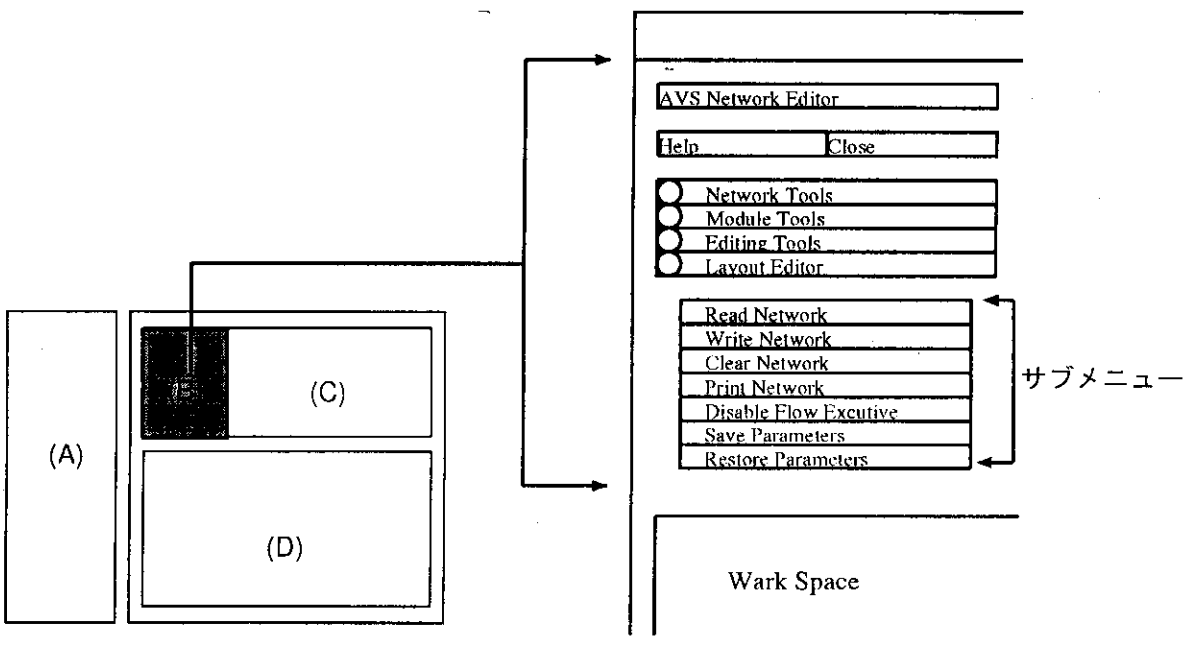

図 1.28 ネットワークエディタメニュー

1.7.1 Module Tools メニュー

ネットワークエディタの Module Tools ボタン（図 1.28）をクリックすると以下のようにサブメニューを表示します。

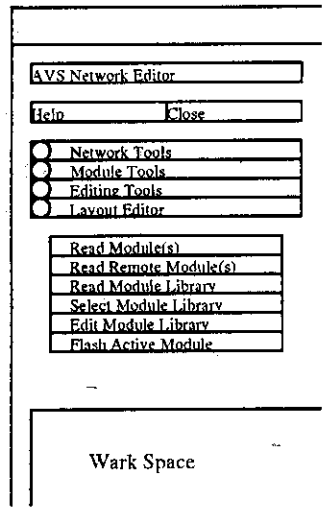

図 1.29 Module Tools メニュー

- Read Module(s)
  現在ネットワークエディタに表示されているモジュールパレットにモジュールを追加します。その方法は、Read Module(s) ボタンをクリックしファイルブラウザを表示させます。次に追加したいモジュールの実行ファイルをクリックするだけです。

- Read Remote Module(s)
  ネットワークにつながる別のシステムからモジュールパレットにモジュールを読み込みます。

- Read Module Library
  新たなモジュールライブラリ（モジュール名とモジュールのカテゴリを記述したファイル）をＡＶＳに読み込むことができます。その方法は、まず、Read Module Library ボタンをクリックするとファイルブラウザを表示します。次に追加したいモジュールライブラリをクリックするだけです。また、ＡＶＳに読み込んだモジュールライブラリの名前は AVS Module Library（図 1.14）に表示します。

- Select Module Library
  モジュールパレットに表示するモジュールライブラリ（AVS Module Library（図 1.14）に表示されている）を選択します。ＡＶＳ起動時には、モジュールライブラリ Supported がモジュールパレットに読み込まれます。

- Edit Module Library
  モジュールライブラリの作成、編集を行います。

- Flash Active Module
  Flash Active Module ボタンをクリックする（オンにする）と、ネットワークの実行時に実行中のモジュールが白く光ります。

1.7.2 Editing Tools メニュー

Editing Tools メニューは、コピー、カット、ペーストなどのネットワーク編集操作とマクロモジュール編集操作を行なうことができます。

(1) コピー＆ペースト、カット＆ペースト

ネットワーク編集操作としては、同じモジュールを２か所以上あるネットワークを作成する場合に便利なコピー＆ペースト、カット＆ペーストがあります。まず、コピー又はカットする部分の選択（モジュールの囲み）を行ないます。

モジュールの囲み

1) カーソルをワークスペースに移動します。次に、囲むモジュールの右上（左上）にカーソルを移動します。
2) マウスの左ボタンを押したまま左下（右下）カーソルを移動すると長方形を表示します。
3) 2) の方法でコピー又は、カットしたい場所を囲みます。
4) ここでコピー又はカットしたい部分を決定したらマウスのボタンを離します。長方形で囲んだ部分が白く光ります。
5) キャンセルする場合は、白く光っている長方形のエリア外でマウスをクリックします。

以下にしてネットワーク編集操作をとしてのコピー＆ペースト、カット＆ペーストの操作方法を示します。

コピー＆ペースト

1) コピーしたい部分をモジュールの囲みの操作で長方形で囲みます。
2) サブメニューの Copy Modules ボタンをクリックすると囲んだ部分をモジュールバッファに保存します。このとき、モジュールを囲んでいる長方形を消去します。モジュールはそのままワークスペースに残ります。また、モジュールの囲みの操作を行なわずに Copy Modules ボタンをクリックした場合は、ＡＶＳはモジュールバッファをクリアします。
3) サブメニューの Paste Modules ボタンをクリックするとモジュールバッファに保存した部分をワークスペースに長方形で囲んだ状態でペーストします。
4) カーソルをペーストした部分に移動しクリックしながらカーソルを移動するとペーストした部分を移動できます。長方形のエリア外でマウスをクリックするとモジュールを囲んでいる長方形は消去されます。

カット＆ペースト

1) コピーしたい部分をモジュールの囲みの操作で囲みます。
2) サブメニューの Cut Modules ボタンをクリックすると囲んだ部分をモジュールバッファに保存します。さらに、ワークスペースの長方形で囲んだモジュールを消去します。

3) サブメニューのPaste Modulesボタンをクリックするとモジュールバッファに保存した部分をワークスペースに長方形で囲んだ状態でペーストします。

4) ペーストした部分は、コピー＆ペーストの4)の操作で移動することができます。長方形のエリア外でマウスをクリックするとモジュールを囲んでいる長方形は消去されます。

(2) マクロモジュールの編集操作

Editing Toolsのもう一つの機能は、マクロモジュール編集操作です。マクロモジュールとは、複数のモジュールとその接続を1つのモジュールで表現するものです。そのためマクロモジュールの取り扱いは、ＡＶＳがサポートするモジュールと同じように扱えます。また、このマクロモジュールは他のマクロモジュールを含むことができます。

マクロモジュールの作成手順

1) ワークスペースに既に存在するネットワークのモジュールを使ってマクロモジュールを作成する場合は、ワークスペースのマクロモジュールとして使うモジュールをモジュールの囲みの方法を使って長方形で囲みます。そうでない場合は、2)から始めて下さい。

2) Editing ToolsメニューのCreate Macro Moduleボタンをクリックするとマクロモジュール名を入力するためのウィンドウを表示します。モジュール名を入力します。

3) 2)で表示されるウィンドウのOkをクリックするとワークスペースに以下(図1.30)のようにマクロモジュールの入力ポート（IN →モジュール）と出力ポート（OUT →モジュール）を表示します。また、1)の操作を行なった場合は長方形で囲んだモジュールも表示します。このとき、ワークスペースにあったネットワークは表示しませんがそのままの状態で保存してあります。

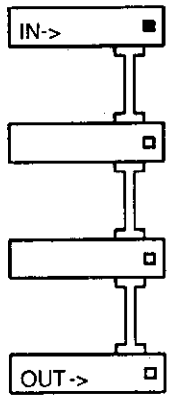

図 1.30 マクロモジュール

4) ワークスペースのネットワークの入力ポートと出力ポートの間にマクロモジュールとして扱うモジュールを接続します。また、モジュールパレットから新たにモジュールを移動してきて接続することもできます。

5) マクロモジュールを保存する場合は、Network Tools メニューの Write Network ボタンをクリックすると以下のウィンドウ（図 1.31）を表示します。

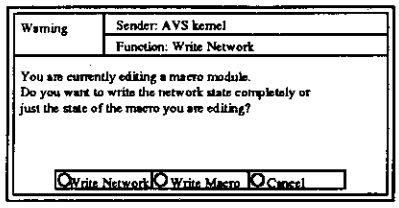

図 1.31 マクロモジュールの保存

- Write Network
  Write Network ボタンをクリックするとマクロモジュールを含んだネットワークをネットワークファイルに保存します。
- Write Macro
  Write Macro ボタンをクリックするとマクロモジュールのみをファイルに保存します。保存したマクロモジュールは、Module Tools のサブメニュー Read Modules を使ってＡＶＳでサポートしているモジュールと同様にモジュールパレットに読み込むことができます。
- Cancel
  Cancel ボタンをクリックするとマクロモジュールの保存を中止します。

6) 1) から開始した場合、Editing Tools メニューの Done Editing Level ボタンをクリックすると 1) の操作の前に存在していたネットワークにマクロモジュールを組み込んだネットワークに変更して表示します。

7) 2) から開始した場合、Editing Tools メニューの Done Editing Level ボタンをクリックするとマクロモジュールを表示します。

マクロモジュールのカテゴリの変更

1) モジュールの囲みの方法を使い、マクロモジュールを長方形で囲みます。

2) Editing Tools メニューのサブメニューの Edit Macro Module ボタンをクリックします。

3) IN →モジュールの内の右側にあるボタンをクリックしてモジュールエディタパネルを起動します。

4) カテゴリの変更は、モジュールエディタパネルの Change Module Category ボタンをクリックしたままにするとポップアップメニューを表示します。マウスをクリックしたままカーソルを設定したいカテゴリ名に移動してボタンを離します。

マクロモジュールの名前の変更

1) モジュール名を変更するには、モジュールエディタパネルの Name の右側にカーソルを移動すると白く光るので、その部分に名前を入力します。リターンキーを入力するとモジュール名を変更します。

1.7.3 Layout Editor メニュー

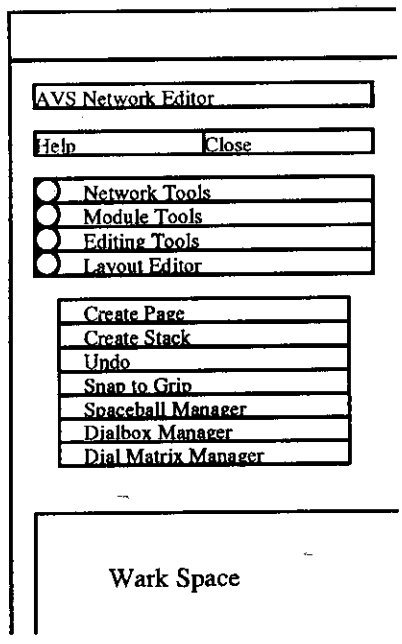

図 1.32 Layout Editor メニュー

Layout Editor メニュー（図 1.32）は、コントロールパネルに表示されているコントロールウィジェットの移動及びウィジェットタイプの変更に用います。Layout Editor メニューは、Layout Editor ボタン（図 1.32）をクリックすると表示します。

ここでは、マウスを使った方法を示します。ただし、この場合、マウスボタンでパラメータを変更できません。

- 移動

  コントロールウィジェットにカーソルを移動しマウスの左ボタンをクリックします。クリックしたままカーソル移動し、任意の場所で左ボタンを離すと、その場所にコントロールウィジェットを移動します。コントロールパネルの外に移動した場合は、ウィンドウとして表示します。

- ウィジェットタイプの変更

  コントロールウィジェットにカーソルを移動し右ボタンをクリックするとポップアップメニューが表示されます。右ボタンをクリックしたまま変更したいタイプの上にカーソルを移動して右ボタンを離すと、そのタイプにコントロールウィジェットを変更することができます

- Create Page

  Create Page ボタン（図 1.32）をクリックすると、コントロールパネルに空のページ（ページの生成）を表示します。

- Undo

  Undo ボタン（図 1.32）をクリックすると、最後に行なった変更を無効にします。続けて 2 回クリックすると 5 回前までの変更を無効にします。ただし、ページ生成のように戻すことのできないものもあります。

1.8 ネットワークの作成例

ここでは、ＡＶＳを使った可視化の表現方法を紹介します。また、ＡＶＳの可視化で使用するモジュールについての説明をします。モジュールの詳しい内容は、ＡＶＳ附属の AVS MODULE REFERENCE を参照して下さい。

なお、ネットワークに取り込むデータのデータタイプの詳しい内容については、次の章で説明します。

1. スカラーデータの表現

   スカラーデータとは、配列の各ノードにデータを１つだけもつようなデータのことをいいます。このスカラーデータを使うと、主に以下のような図を作成することができます。

   (a) コンター図

   (b) 鳥瞰図

   (c) ３次元グラフ

   (d) 等値面

2. ベクトルデータの表現

   スカラーデータに対して配列の各ノードにデータを２つ以上もつようなデータをベクトルデータといいます。このベクトルデータを使うと、ベクトル図を作成することができます。

コンター図

コンター図は、各ノードのスカラーデータの値に対して対応するカラーデータ（カラーデータは generate colormap モジュールで生成します）を与えて、色の変化によってデータを表示します。

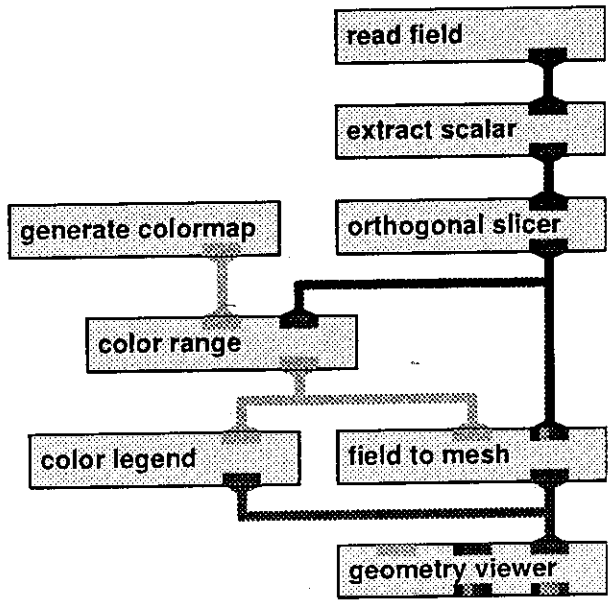

図 1.33 ネットワーク read_wata_mesh.net

ネットワークに read_wata_mesh.net （図 1.33）を、データに read_wata.fld を使い、コンター図を作成します。ネットワークの概要は、以下のとおりです。

1) read field モジュールのボタンを左ボタンでクリックしてネットワークコントロールパネルにファイルブラウザを表示し、ファイル
/home/guest/avsuser/data/read_wata.fld
を指定してデータを読み込みます。データ read_wata.fld は 40x20x3 の大きさを持つ 3 次元の配列です。（2章 ＡＶＳで扱われるデータタイプ参照）

2) ベクトルデータ（各ノードに複数のデータをもつ）なので、extract scalar モジュールによってベクトルデータから任意のスカラーデータを取り出します。

3) orthogonal slicer モジュールによって断面のデータ（3 次元のデータから 2 次元のデータの抽出）を取り出します。

4) 取り出した断面のデータを field to mesh モジュールによってカラーデータに変換し表示用の幾何データ（ジオメトリデータ）を出力します。カラーデータは、generate colormap モジュールにより生成します。

5) 表示用の幾何データ（ジオメトリデータ）をもとに可視化したイメージを geometory viewer モジュールによりディスプレイ上（ビューウィンドウ）に表示します。

以下にネットワーク read_wata_mesh.net で用いた各モジュールの詳細を示します。モジュール名の右側のカッコには、そのモジュールのカテゴリを示します。

- orthogonal slicer モジュール (mapper)

  入力データ：フィールドデータ 2D/3D n-vector any-data any-coordinates

  出力データ：フィールドデータ 1D/2D n-vector same-data same-coordinates

  orthogonal slicer モジュールは、3 次元から 2 次元のデータ（xy,yz,zx 平面に平行な断面のデータ）を、または、2 次元から 1 次元のデータ（x,y 軸に平行な直線上のデータ）を取り出します。ネットワーク read_wata_mesh.net では、3 次元から 2 次元のデータを取り出します。2 つのコントロールウィジェットを使い、どの平面上（パラメータ axis）のどの位置の断面（パラメータ slice plane）のデータを取り出すかを設定します。

  データ read_wata.fld は、図のように 40x20x3 の 3 次元のフィールドデータです。read_wata.fld は、z = 0 の xy 平面上のみにデータをもっており、z = 1, z = 2 の xy 平面上のデータ値は、全て 0 になっています。ここでは、xy 平面に平行な z = 0 の断面のデータを取り出すようにパラメータを設定します。orthogonal slicer モジュールのコントロールウィジェットを表示させ、パラメータ axis を K に、slice plane を 0 に設定します。コントロールウィジェットの使用方法は図 1.20を参照して下さい。

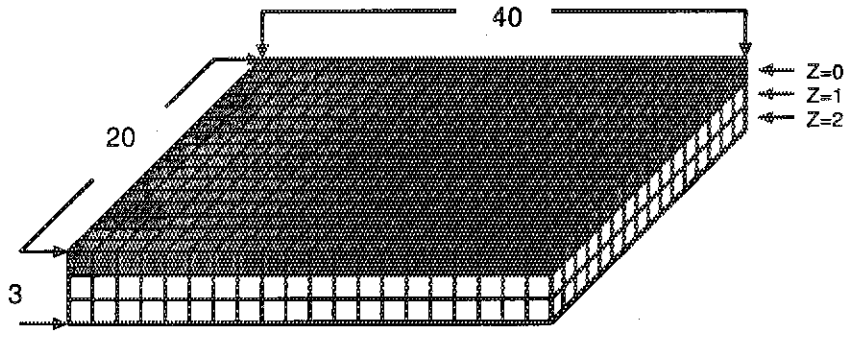

図 1.34 データ read_wata.fld

- field to mesh モジュール

  入力データ：フィールドデータ 2D scalar any-data any-coordinates

  カラーデータ

  出力データ：ジオメトリデータ

  カラーデータ（generate colormap モジュールで生成する）を入力とし、カラーデータに従ってコンター図（ジオメトリデータ）を生成します。

- extract scalar モジュール

  入力データ：フィールドデータ any-D n-vector any-data any-coordinates

  出力データ：フィールドデータ same-D scalar any-data any-coordinates

  ベクトルデータから任意のスカラーデータを取り出します。読み込んだデータは、各ノードに 5 つのデータをもっています。コントロールウィジェット channel 1 ～ channel n の任意のボタンをクリックしスカラーデータを選択します。（ここでは、channel 0 ～ channel 4 のうちの channel 3 をクリックします。）

- geometry viewer モジュール

  入力データ：ジオメトリデータ

  出力データ：フィールドデータ 2D 4-vector byte

  geometory viewer モジュールは表示用の幾何データ（ジオメトリデータという）に従って可視化したイメージを表示します。ここでは、field to mesh モジュールによって生成したコンター図と color legend モジュールによって生成された対応表（カラーデータとデータの対応表）を表示します。

  geometry viewer モジュールのコントロールウィジェットを表示するには、geometry viewer モジュールの右側にあるボタンをマウスの左ボタンでクリックして下さい。ネットワークコントロールパネルウィンドウの上に、geometry viewer モジュールのコントロールパネルを表示し、コントロールパネル上に geometry viewer モジュールのパラメータを設定するためのコントロールウィジェットを表示します。

  geometry viewer モジュールは、ビューウィンドウに可視化したイメージを、ビューウィンドウの中心を (x,y,z)=(0,0,0) として、データがもつ座標位置に表示します。このとき、可視化したイメージがビューウィンドウに表示しきれずにはみ出してしまったり、場合によってはビューウィンドウの外にあって全く表示されない場合もあります。このようなときに、geometry viewer モジュールの Normalize ボタン（図 3.2）をクリックすると、ウィンドウ内に可視化したイメージ全体が見えるようにイメージを移動、または、拡大 / 縮小して表示することができます。

  geometry viewer モジュールの詳しい内容については「3.2 サブシステム」を参照して下さい。

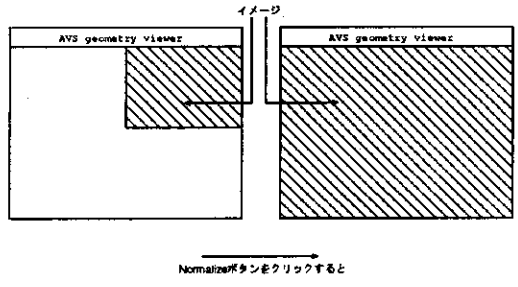

図 1.35 Normalize ボタンの使用前と使用後

- generate colormap モジュール（color range モジュールの説明も参照して下さい）

  入力データ：なし

  出力データ：カラーデータ

  generate colormap モジュールはカラーデータを生成します。生成したカラーデータは、以下のようなモジュールの入力ポートから取り込みます。

  - colorizer モジュール
  - arbitrary slicer モジュール
  - bubbleviz モジュール
  - field to mesh モジュール
  - isosurface モジュール

ネットワークコントロールパネルの generate colormap をクリックするとネットワークコントロールパネルに generate colormap モジュールのコントロールウィジェット（図 1.36）を表示します。カラーデータは、グラフ領域に表示するマップに従って生成されます。

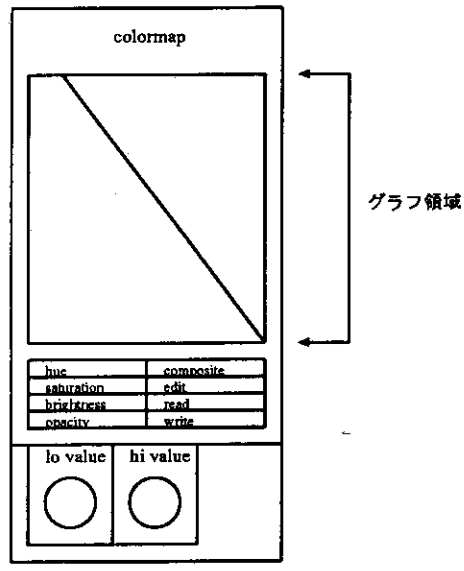

図 1.36 generate colormap モジュールのコントロールウィジェット

グラフ領域は、色を混ぜ合わせるのに使うパレットのようなものです。色は、色相 (hue)、明度 (saturation)、彩度 (brightness) 及び、不透明度 (opacity) によって決定されます。色相 (hue)、明度 (saturation)、彩度 (brightness) 及び、不透明度 (opacity) の設定は、それぞれコントロールウィジェットの hue ボタン, saturation ボタン, brightness ボタン, opacity ボタン（図 1.36）をクリックすることによってグラフ領域にそれぞれのマップを表示させて変更します。グラフ領域は、細かく分割された水平バー（２５６列に分割されている）から成るマップを表示します。例えば hue ボタンをクリックするとグラフ領域に色相のマップを表示します。初期設定では、色相のマップは、水平バーの０～２５５列が青から赤へ線形に変化するように表示されます。

以下に、色相 (hue) のマップを例にグラフ領域に表示されるマップの変更方法を示します。

1) hue ボタン（図 1.36）をクリックしてグラフ領域に色相のマップを表示させます。

2) edit ボタン（図 1.36）をクリックするとカラーマップエディタ（図 1.37）を表示します。水平バーの０～２５５列が青から赤へ線形に変化するように表示されます。

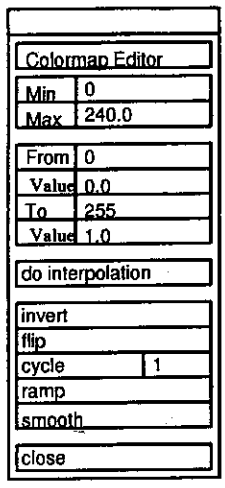

図 1.37　カラーマップエディタ

3) カラーマップの色相の範囲を入力します。ＡＶＳでは、色相を円で表現しています。Min,Max に 0 ° ～ 360 ° の範囲で数値を入力します。初期設定では、円の 0 °（赤）～ 240 °（青）の部分を設定しています。

4) マップの指定した水平バーに、任意のカラーを設定することができます。
例として、水平バーの 0 列～ 100 列を赤（0.0）に変更したい場合を示します。

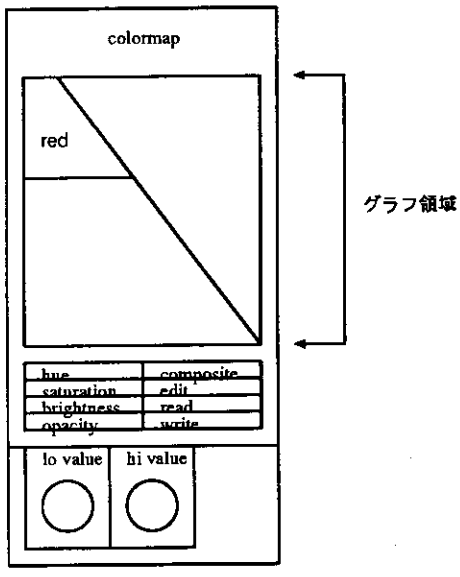

図 1.38　カラーマップの変更

図1.37に表示されているの項目の右側に以下のように数値を入力します。

From : 0
Value : 0.0
To : 100
Value : 0.0

do interpolation ボタン（図1.37）をクリックすると、図1.38のように色相のマップの0列～100列の範囲が赤に変更されます。

5) Write ボタンをクリックするとファイルブラウザを表示するので、これを使って編集したマップをファイルに保存することができます。また、Read ボタンを使うと保存したマップを読み込むことができます。

同様な方法で明度、彩度、不透明度のマップも変更することができます。また、composite ボタンをクリックすると、色相、明度、彩度、不透明度を合成した色をグラフ領域に表示します。

- color range モジュール

入力データ：フィールドデータ any-D n-scalar any-data any-coordinates
カラーデータ
出力データ：カラーデータ

スカラーデータとカラーデータを入力とします。スカラーデータの最大値と最小値をカラーマップの最大値と最小値に合うように正規化し出力します。

例えば、以下のように0～2.55までの範囲にあるデータをカラーデータに変換する場合に color range モジュールを使うことで、データに対応するカラーデータを正規化することができます。

データ	色相	水平バー (列)
0	青	0
0.01		1
0.02		2
.....		
2.53		253
2.54		254
2.55	赤	255

- color legend モジュール

入力データ：カラーデータ
出力データ：ジオメトリデータ

データとカラーデータの対応表（ジオメトリデータ）を生成します。

鳥瞰図

コンター図ではデータを色の変化によって表現しました。鳥瞰図ではこれに加えてデータを起伏の大きさで表現します。データを起伏の大きさで表現することで変化の大小を知ることができます。鳥瞰図は地形の形状などを表現するのに有効です。

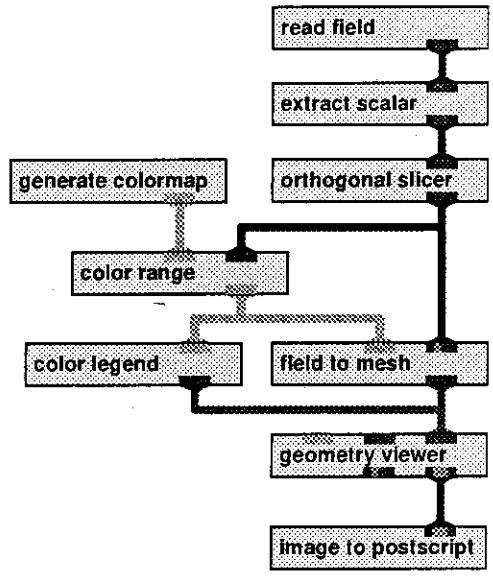

図 1.39 ネットワーク read_wata_chokan.net

ネットワークに read_wata_chokan.net （図 1.39）を、データに read_wata.fld を使い、鳥瞰図を作成します。鳥瞰図を作成するネットワークの概要は以下のとおりです。1) ～ 3) の処理はコンター図と同じです。

1) read field モジュールでファイル
/home/guest/avsuser/data/read_wata.fld
を指定してデータを読み込みます。データは 40x20x3 の大きさを持つ 3 次元の配列です。

2) ベクトルデータなので extract scalar モジュールでベクトルデータから任意のスカラーデータを取り出します。

3) orthogonal slicer モジュールによって断面のデータ（3 次元のデータから 2 次元のデータの抽出）を取り出します。

4) 取り出した断面のデータを field to mesh モジュールによってカラーデータに変換し、さらに、起伏をつけた表示用の幾何データ（ジオメトリデータ）を出力します。

5) 表示用の幾何データ（ジオメトリデータ）をもとに可視化したイメージを geometory viewer モジュールによりディスプレイ上（ビューウィンドウ）に表示します。

以下にネットワーク read_wata_chokan.net で使用したモジュールの詳細を示します。

- field to mesh モジュール

  入力データ：フィールドデータ 2D scalar any-data any-coordinates
  カラーマップデータ

  出力データ：ジオメトリデータ

  field to mesh モジュールは、データ値をカラー値に変換し、さらに断面のデータを各ノードのもつデータの値の大きさに合わせて凹凸のある面に変換します。高さのスケールは、パラメータ Z scale で決定します。凹凸の高さは、データ値に比例します。

- image to postscript モジュール

  入力データ：フィールドデータ 2D 4-vector byte

  ビューウィンドウに表示している可視化したイメージを Postscript ファイルに保存します。

  image to postscript モジュールのコントロールウィジェット（タイプインウィジェット）を表示させて、filename の右側にファイル名を入力して下さい。カーソルをファイル名を入力した場所から離すと、可視化したイメージを入力した Postscript ファイルに保存します。

  保存終了後にパラメータの変更などで再実行を行った場合、image to postscript モジュールが入力したファイルに再書き込みを行なうのを防ぐために実行しません。再び保存したい時は、タイプインウィジェット filename にファイル名を入力し直して下さい。

  保存したファイルは Postscript 対応のプリンタで出力することができます。（情報システム管理課では、画像処理サーバである furies のローカルプリンタか、ネットワークプリンタのいずれかで出力することができます。）出力するプリンタの指定は、スタートアップファイル.avsrc で行ないます。

3次元グラフ

3次元グラフでは、スカラーデータを棒グラフで表現します。3次元グラフでは、図1.40のように2次元のスカラーデータの大きさに従って決定される値を高さとする棒グラフを表示するものです。

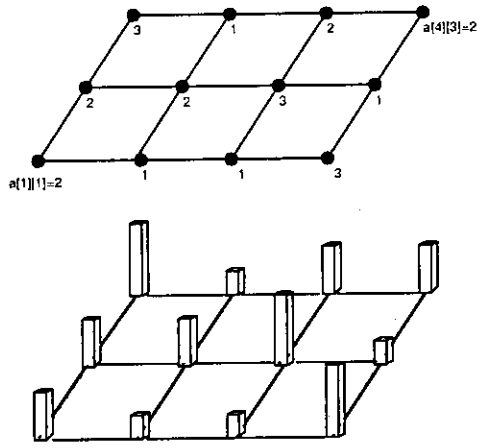

図 1.40　3次元グラフ

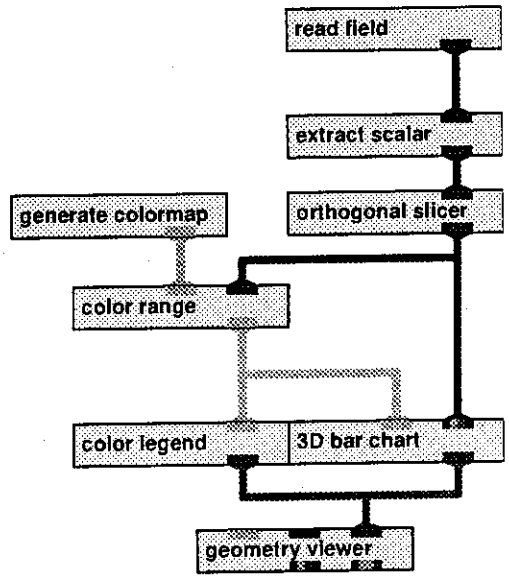

図 1.41 ネットワーク read_wata_3d.net

ネットワークに read_wara_3d.net （図 1.41）を、データに read_wata_uniform.fld を使い、3 次元の棒グラフを作成します。3 次元グラフを作成するためのネットワークの概要は以下のとおりです。

1) read field モジュールでファイル
/home/guest/avsuser/data/read_wata_uniform.fld
を指定してデータを読み込みます。データは 40x20x3 の大きさをもつ 3 次元の配列です。

2) ベクトルデータなので extract scalar モジュールでベクトルデータから任意のスカラーデータを取り出します。

3) orthogonal slicer モジュールによって断面のデータ（3 次元のデータから 2 次元のデータの抽出）を取り出します。

4) 取り出した断面のデータを 3D bar chart モジュールによって棒グラフを表示するための幾何データに変換します。

5) 表示用の幾何データ（ジオメトリデータ）をもとに可視化したイメージを geometory viewer モジュールによりディスプレイ上（ビューウィンドウ）に表示します。

以下にネットワークで用いたモジュールの詳細を示します。

- 3D bar chart モジュール
  入力データ：フィールドデータ 2D scalar float uniform
  カラーマップ
  出力データ：ジオメトリデータ
  データの大きさを棒グラフの高さとして表示するための幾何データに変換します。パラメータ Z scale によって高さのスケールを設定します。また、カラーデータを取り込んで棒グラフに色を与えます。

等値面

等値面は、２次元上で同じ値のスカラーデータを線で結んで等値線を作成するように、３次元上において同じ値のスカラーデータを面で結ぶことによって作成します。

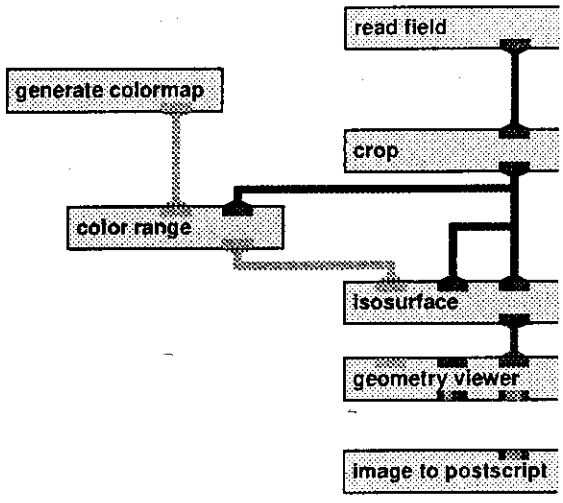

図 1.42 ネットワーク isosurface.net

ネットワークに isosurface.net （図 1.42）を、データに read_con.fld を使い等値面を作成します。等値面を生成するためのネットワークの概要は以下の通りです。

1) read field モジュールでファイル
/home/guest/avsuser/data/read_wata.fld
を指定してデータを読み込みます。データは 40x20x3 の大きさを持つ３次元の配列です。

2) isosurface モジュールで等値面を生成し、表示用の幾何データ（ジオメトリデータ）として出力します。また、generate colormap モジュールによって等値面にカラーを与えます。

3) 表示用の幾何データ（ジオメトリデータ）をもとに可視化したイメージを geometory viewer モジュールによりディスプレイ上（ビューウィンドウ）に表示します。

以下にネットワーク isosurface.net で使用したモジュールの詳細を示します。

- crop モジュール
  入力データ：フィールドデータ（任意のタイプ）
  出力データ：フィールドデータ（任意のタイプ）
  入力ポートから取り込んだデータの中から指定した範囲のデータを取り出して出力します。取り出す範囲は、配列のインデックスで指定します。配列のインデックスは、コントロールウィジェットを表示させ、パラメータ min x, max x, min y, max y, min z, max z を設定します。

- isosurface モジュール

  入力データ：フィールドデータ 3D scalar any-data any-coordinates
  　　カラーデータ

  出力データ：ジオメトリデータ

  3 次元上の同じ値を持つスカラーデータを面で結んで等値面を生成します。生成する等値面の値はコントロールウィジェットを表示させ、パラメータ isosurface level を設定します。また、カラーマップからカラーデータを取り込んで等値面をカラーで表示します。

ベクトル図

ベクトル図は、ベクトルデータを矢印で表現します。例えば、風や磁場の向きと強さを同時に表現したい場合に有効です。ベクトルデータは、x、y 及び z の 3 成分をもっていなければなりません。

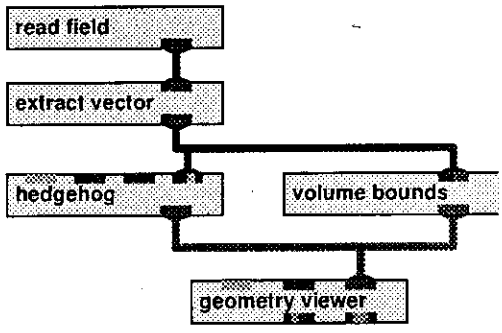

図 1.43. ネットワーク read_wata_vector.net

ネットワークに read_wata_vector.net を、データに read_wata.fld を使ってベクトル図を作成します。ベクトル図を作成するためのネットワークの概要は以下の通りです。

1) read field モジュールでファイル
/home/guest/avsuser/data/read_wata.fld
を指定してデータを読み込みます。データは 40x20x3 の大きさを持つ 3 次元の配列です。

2) データは、ベクトルデータです。各ノードにもつ 5 つのデータのうち 3 つのデータを extract vector モジュールで取り出します。

3) hedgehog モジュールで各ノードの座標値と extract vector モジュールで取り出した 3 成分のデータから矢印を表す幾何データを出力します。

4) 表示用の幾何データ（ジオメトリデータ）をもとに可視化したイメージを geometory viewer モジュールによりディスプレイ上（ビューウィンドウ）に表示します。

- hedgehog モジュール

  入力ポート：フィールドデータ 3D 3-vector float

  出力ポート： ジオメトリデータ

  hedgehog モジュールは、3つのデータをもつベクトルデータ（実数値）を入力とします。矢印の表示する座標を X, Y, Z とし、3つのベクトルデータの値を x, y, z とすると、（X, Y, Z）と（X+x, Y+y, Z+z）を結ぶ矢印を表す幾何データを出力します。hedgehog モジュールでは、以下のようなパラメータを設定します。

  - Vector Scale

    表示する矢印の長さのスケールを決めます。表示される矢印の長さは Vector Scale の値に比例します。

  - Sumple

    3次元上に全ての矢印を表示すると、数が多すぎて見づらくなってしまうので、どの領域の矢印を生成するのかを指定します。hedgehog モジュールは、指定した領域の矢印を生成します。選択できる領域は以下の5通りです。

    * 点上
    * 直線上
    * 円上
    * 断面上
    * 全体

    また、矢印を表示する領域の位置は、geometry viewer モジュールで指定します。

  - N Segment

    表示する矢印の数を整数で入力します。

  - arrow heads

    arrow heads ボタンをオンにするとベクトルデータを矢印で表示し、オフにすると線分で表示します。

  - Show Bounds

    Show Bounds ボタンをオンにすると矢印を表示している領域を囲む線を生成します。

- volume bounds モジュール

  入力データ：フィールドデータ 3D n-vector any-data any-coordinates

  出力データ：ジオメトリデータ

  データの存在する範囲を囲む3次元3成分の枠を表す幾何データ生成します。

- geometry viewer モジュール

  入力データ：ジオメトリデータ

  geometry viewer モジュールは、表示用の幾何データ（ジオメトリデータ）を可視化したイメージとしてディスプレイ上に表示します。また、hedgehog モジュールの矢印を表示する領域の位置を指定するために geometry viewer モジュールを使用します。以下に使用手順を示します。

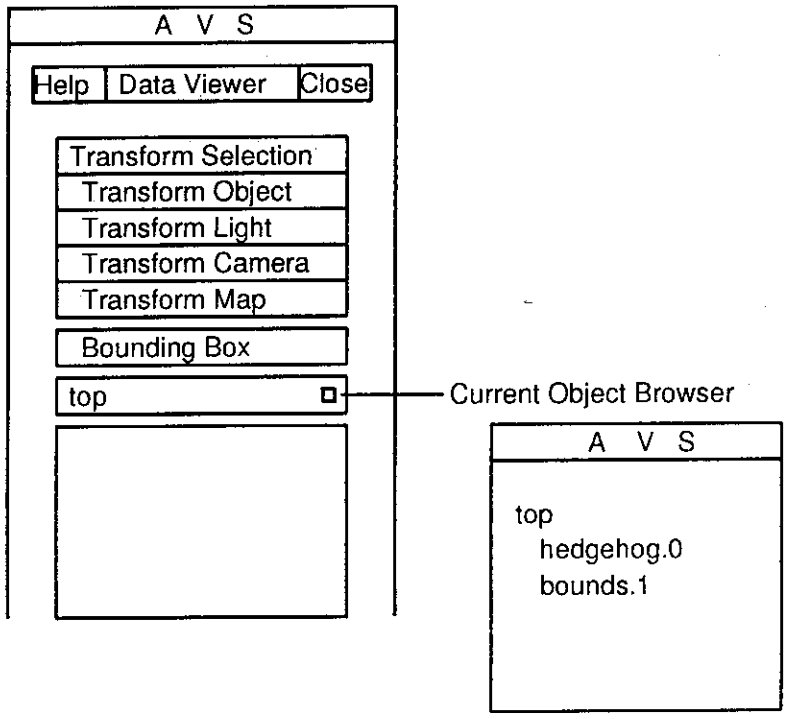

図 1.44. Current Object Browser

表示する矢印の領域の位置は、以下のようにして指定します。

1) ワークスペース上の geometry viewer モジュール内の右側にある正方形のボタンをクリックすると、geometry viewer モジュールのコントロールパネルを表示します。
2) コントロールパネルを（図 1.44）に表示される正方形のボタンをクリックすると Current Object Browser を表示します。
3) Current Object Browser の hedgehog.0 の上にカーソルを移動しクリックします。このようにして選択したオブジェクトをカレントオブジェクトといいます。
4) ビューウィンドウ内にマウスを移動します。
5) マウスの右ボタン（平行移動）と中ボタン（回転）で領域の位置を移動することができます。カレントオブジェクト hedgehog.0 を移動した位置のデータを表示します。

表現の組み合わせ

ＡＶＳは、これまで紹介してきた表現方法のいくつかを組み合わせて可視化することができます。その例として以下に、コンター図とベクトル図の組み合わせを示します。

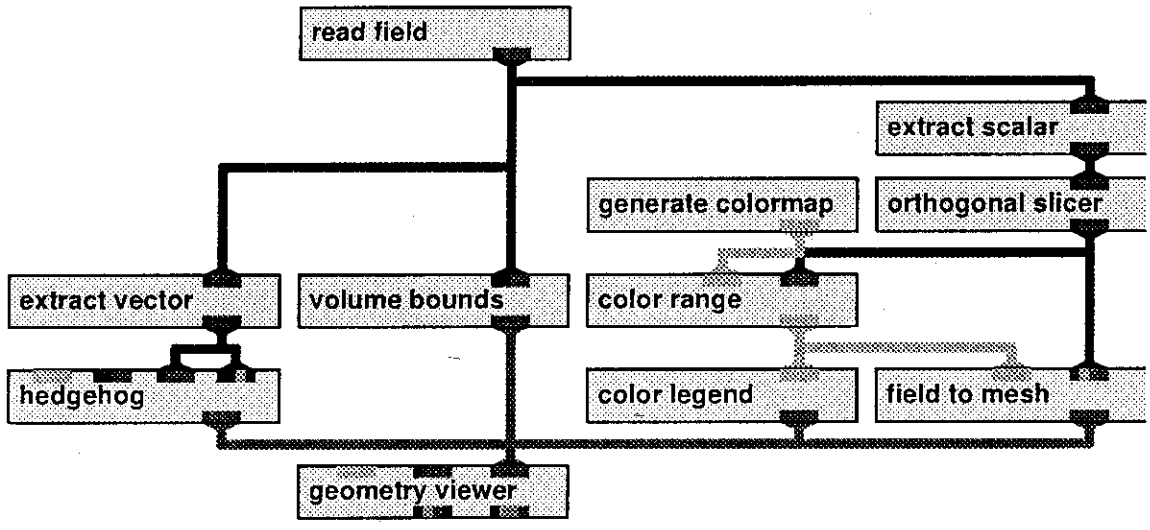

図 1.45　ネットワーク read_wata_plus.net

ネットワークに read_wata_plus.net を、データに read_wata.fld を使ってコンター図とベクトル図をの組み合わせを作成します。コンター図とベクトル図をの組み合わせを作成するためのネットワーク（図 1.45）の概要は以下の通りです。

ネットワーク read_wata_plus.net は、１つのファイルからデータを読み込み、コンター図を生成する処理とベクトル図を生成する処理を別々に行ない、最後に１つのビューウィンドウに処理を移しコンター図とベクトル図とを組み合わせた可視化したイメージを重ねて表示します。

2．AVSで扱われるデータタイプ

ＡＶＳにデータを取り込むには、以下の２つのいずれかの方法でデータを作成しなければなりません。

(1) データをＡＶＳが定めるデータフォーマットに従って作成する方法

(2) ユーザ独自のデータ入力モジュールを作成し、ＡＶＳのデータとして読み込む方法

ＡＶＳで扱われるデータタイプには、次のようなものがあり、ここではそれぞれのデータタイプの概要を述べ、詳しい説明を2.1以降に述べます。

1) フィールド（構造格子）データ
フィールドデータは、ＡＶＳで最も頻繁に使われるデータタイプで、Ｉ×Ｊ×Ｋ×……の大きさをもつ配列にデータをセットして使用します。フィールドデータでは、配列のインデックスに従ってノード（配列の１つ１つ）が並びます。このために、ノードの並びはメッシュ上になります。フィールドデータは、数値データとノードの座標を示す座標データで構成されます。

2) ＵＣＤ（非構造格子）データ
ＵＣＤデータは、１次元の配列にデータをセットして使用します。ＵＣＤデータでは、ノードの並びのデータをもっており、それに従ってノードが並びます。ＵＣＤデータは、数値データ、座標データ及びノードの並びのデータから構成されます。

3) ジオメトリデータ
ジオメトリデータは、表示用の幾何データです。フィールドデータやＵＣＤデータを可視化する場合に、ジオメトリデータに変換します。ＡＶＳでは、ジオメトリデータに従ってディスプレイ上に可視化したイメージを表示します。ジオメトリデータを作成する方法は、以下の３通りのです。

- フィールドデータやＵＣＤデータをMapperモジュールによってジオメトリデータに変換する方法
- ＡＶＳで定めているデータフォーマットでデータを作成し、フィルターによってジオメトリデータを作成する方法
- ＡＶＳジオメトリ関数を使用してジオメトリデータを生成するモジュールを作成する方法

4) カラーデータ
カラーデータは、データの最大値～最小値に色を割り当てるために使用するデータタイプです。カラーデータはgenerate colormapモジュールで生成します。

5) プリミティブデータ
以下に示すプリミティブデータは、モジュール間で受け渡すパラメータ（例：slice planeの値、Z scaleの値等）に使用します。

- バイト（byte）
- 整数
- 単精度浮動小数点
- 倍精度浮動小数点
- 文字列

2.1 フィールドデータ

(1) フィールドデータとは、

フィールドデータは、配列の１つ１つ、即ち、ノードにデータをセットして使用するデータタイプであり、ノードはきれいなメッシュ上に並びます。通常、１つのノードにデータを１個づつしかもつことができませんが、フィールドデータは、１つのノードに複数のデータをもつことができます。ここでは、次のようなデータを例にしてフィールドデータを説明します。

表 2.1 データ例 example

場所	温度	高さ
A 1	2 0	5 0
A 2	2 2	2 0
B 1	2 5	7 0
B 2	2 0	6 0
C 1	2 2	1 0
C 2	3 0	4 0

データ例 example は、１つのノードに２つのデータ（温度と高さ）をもっています。このデータをフィールドデータとしてＡＶＳに取り込む場合、データを以下のような２次元の配列にセットします。

Data(i,j)
i=1,2 j=1,3
Data(1,1)=(20,50)
Data(2,1)=(22,20)
Data(1,2)=(25,70)
Data(2,2)=(20,60)
Data(1,3)=(22,10)
Data(2,3)=(30,40)

配列 Data は２×３の大きさをもつ２次元の配列です。フィールドデータは、任意の次元の任意の大きさの配列にデータをセットすることができます。

また、フィールドデータでは、配列の１つのノードに複数のデータをセットすることができます。しかし、セットする複数のデータは、同じデータタイプ（バイト、整数、単精度浮動小数点、倍精度浮動小数点のうちのいずれか）でなくてはいけません。配列 Data は、温度と高さの２つのデータをもち、そのデータタイプは、整数です。

(2) フィールド

フィールドデータを可視化するためには、データを以下の物理空間の座標上のいずれかにマッピングするのかを決めなければなりません。この物理空間のことをフィールドといいます。uniform（ユニフォームフィールド）、rectilinear（レクティリニアフィールド）、irregular（イレギュラーフィールド）の3種類のフィールドの中からデータをマッピングするフィールドを選択します。

1) uniform（ユニフォームフィールド）

ユニフォームフィールドを選択した場合は、データを配列のインデックスに従って等間隔にフィールド上にマッピングします。（図2.1参照）

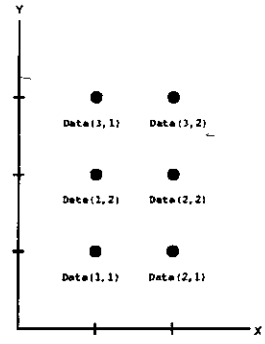

図 2.1 ユニフォームフィールド

2) rectilinear（レクティリニアフィールド）

レクティリニアフィールドを選択した場合は、各軸のインデックスに対して決めた座標値（浮動小数点）に従ってデータをマッピングします。（図2.2参照）

以下のように各軸のインデックスに対して座標値を定めた場合は、ノードは図2.3のように並びます。

x=50.0,80.0

y=10.0,40.0,90.0

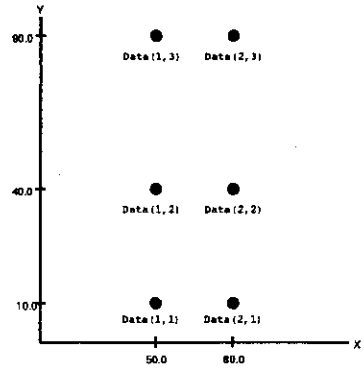

図 2.2 レクティリニアフィールド

3) irregular（イレギュラーフィールド）

イレギュラーフィールドを選択した場合は、各ノードに対して決めた座標値（浮動小数点）に従ってデータをマッピングします。つまり、各ノード（Data(i,j)）に対して座標値（x, y）を決め、その座標値に従ってデータをマッピングします。（図 2.3参照）

以下のように各ノードに対して座標値を定めた場合に、ノードは図 2.3のように並びます。

Data(1,1) の x,y 座標 5.0,5.0
Data(2,1) の x,y 座標 10.0,20.0
Data(1,2) の x,y 座標 10.0,30.0
Data(2,2) の x,y 座標 30.0,30.0
Data(1,3) の x,y 座標 20.0,50.0
Data(2,3) の x,y 座標 50.0,5.0

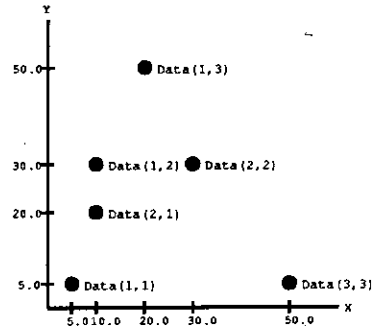

図 2.3 イレギュラーフィールド

フィールドデータのデータタイプの表現方法を以下に説明します。

表 2.3：データ例 example の場合は、

field 2D 2-space 2-vector integer uniform

になります。上記の表記は、

- 計算空間（配列）が 2 次元
- 物理空間（フィールド）が 2 次元
- 配列の各ノードのデータ数（要素数）が 2
- フィールドデータのデータタイプが整数
- フィールドのタイプ uniform

であることを表しています。すなわち、この表現方法は表記法に基づいて表されます。

?D	計算空間（配列）の次元数
?-space	物理空間（フィールド）の次元数
?-vector	配列の各ノードのデータ数（要素数）
integer	フィールドデータのデータタイプ
uniform	フィールドのタイプ

(3) 入出力ポート

第1章で入出力ポートについて説明しましたが、フィールドデータの扱う入出力ポートは、さらに4つに区分して表示します。

フィールドデータポートの色とデータタイプについて説明します。

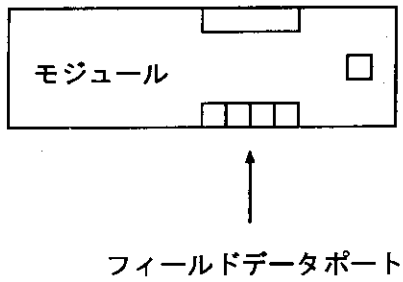

図 2.4 フィールドデータポート

フィールドデータポートの左端の色は、全て青になっておりフィールドデータであることを表します。フィールドデータポートの左から二番目の色は、フィールドの次元数を表します。次元数とフィールドデータポートの色は、以下のように対応します。

表 2.2 ポートの色と次元数

フィールドデータポートの色	データタイプ
赤	1次元
緑	2次元
青	3次元
灰	任意の次元

フィールドデータポートの三番目の色は、各ノードのもつ要素数を表します。

表 2.3 ポートの色と要素数

色	要素数
赤	1
緑	2以上
灰	任意の要素数

フィールドデータポートの四番目の色は、各ノードのもつデータのデータタイプを表します。

表 2.4　ポートの色とデータタイプ

色	要素数
赤	バイト
緑	整数
青	単精度浮動小数点
黄	倍精度浮動小数点
灰	上記のいずれか （任意）

モジュールによってはフィールドデータポートの色が灰色のものがありますが、そのフィールドデータポートをもつモジュールは、サポートされている選択肢のいずれでも取り扱えることを示しています。

例えば、モジュールを接続するときに出力側のモジュールのポートの４番目の色が赤（データタイプがバイト）である場合、入力側のモジュールの４番目のポートの色が灰色（データタイプが任意）であればこの２つのモジュールは接続できます。

2.1.1　フィールドデータのファイルフォーマット

フィールドデータは、read field モジュールによってＡＶＳに取り込むことができます。そのため、ユーザは、作成したデータをフィールドデータのフォーマットに変換してＡＶＳに取り込まなければなりません。

ここでは、まず、フィールドデータのフォーマットを示します。次に、ユーザの作成したデータをフィールドデータのフォーマットに変換してＡＶＳに取り込む方法について示します。

フィールドデータのフォーマットは、次のような２つの部分で構成されます。

1) ＡＳＣＩＩヘッダー

2) バイナリ領域

　a) データ領域

　b) 座標領域

ＡＳＣＩＩヘッダーとバイナリ領域は、セパレーター文字で区切ります。セパレータ文字には、

＊ コントロール Ｌ

＊ ８進数の１４

＊ １０進数の１２

＊ １６進数の０Ｃ

のいずれかの文字を２回続けて書きます。（^L^L）

以下に ASCII ヘッダー、バイナリ領域のそれぞれの内容について説明します。

(1) ＡＳＣＩＩヘッダーのフォーマット

以下にデータ例 example（表 2.1）をフィールドデータとしてＡＶＳに取り込んだ場合のＡＳＣＩＩヘッダーのフォーマットを説明します。

（例）

```
# AVS
# field file
ndim=2
dim1=2
dim2=3
nspace=2
veclen=2
data=integer
field=uniform
```

AVS

ＡＳＣＩＩヘッダーは、最初は必ず＃　AVS で始まります。ＡＶＳは、フィールドデータであることを確かめるために最初の 5 文字が＃　AVS であることを確認します。

field file

#以下は、コメント行として扱われます。

ndim=2

ndim は、フィールドデータをセットする配列の次元数を記述します。データ例 example では 2 次元の配列を使っているので ndim=2 となります。

dim1=2、dim2=3

dim1,dim2 は、配列の各次元の大きさを記述します。データ例 example をセットした配列 Data (i,j) の場合、dim1 にインデックス i の大きさ 2、dim2 に j の大きさ 3 を記述します。

nspace=2

nspace は、データをマッピングする物理空間（フィールド）の次元数を記述します。

veclen=2

veclen は、配列の各ノードにセットするデータ数（要素数）を記述します。データ例 example では、データを 2 つ（温度と高さ）もつので veclen=2 となります。

data=integer

data は、配列の各ノードにセットするデータのデータタイプ (byte,integer,float,double) を記述します。データ例 example では、整数のデータを扱っているので data=integer となります。

field=uniform

field は、マッピングする物理空間（フィールド）のタイプ (uniform,rectilinear,irreular) を記述します。

(2) バイナリ領域のフォーマット

a) データ領域のフォーマット

データ領域には、データが配列のインデックスの順に並びます。ただし、実際には以下に示すデータは、バイナリで表現されています。左端の文字（2桁）がデータを示し、コロン（：）以降はそのデータの説明であり、実際には、データ領域にはありません。

```
20      : Data(1,1) の1つ目（温度）のデータ
50      : Data(1,1) の2つ目（高さ）のデータ
22      : Data(2,1) の1つ目（温度）のデータ
20      : Data(2,1) の2つ目（高さ）のデータ
25      : Data(1,2) の1つ目（温度）のデータ
70      : Data(1,2) の2つ目（高さ）のデータ
20      : Data(2,2) の1つ目（温度）のデータ
60      : Data(2,2) の2つ目（高さ）のデータ
22      : Data(1,3) の1つ目（温度）のデータ
10      : Data(1,3) の2つ目（高さ）のデータ
30      : Data(2,3) の1つ目（温度）のデータ
40      : Data(2,3) の2つ目（高さ）のデータ
```

b) 座標領域のフォーマット

rectilinear, irregular の場合には、データ領域と座標データを記述した座標領域が必要です。 uniform の場合には、データ領域のみで座標領域は必要ありません。

rectilinear, irregular の場合は、データを座標データに従ってフィールドへマッピングします。ただし、座標データは、データ領域と同様にバイナリで表現します。ただし、座標領域のデータは、実数でなければなりません。

1) rectilinear の場合

rectilinear の場合、各ノードのデータを各座標軸のインデックスに対して決めた座標値に従ってフィールドにマッピングします。データ例 example（表 2.1）を図 2.5に示すように座標値を決めた2次元のフィールド（フィールドの各軸をそれぞれ x 軸、y 軸とする）にマッピングする場合、座標領域に各ノードのデータは以下のように並びます。

座標情報はX軸、次にY軸のインデックスの若い順に並べます。以下に示す座標情報では、コロンの左側にある数字（実数）が座標値で、コロン（：）以降は、そのデータの説明であり、実際には、座標領域にありません。

50.0　　：フィールドのｘ軸の１番目のインデックス
80.0　　：フィールドのｘ軸の２番目のインデックス
30.0　　：フィールドのｙ軸の１番目のインデックス
60.0　　：フィールドのｙ軸の２番目のインデックス
90.0　　：フィールドのｙ軸の３番目のインデックス

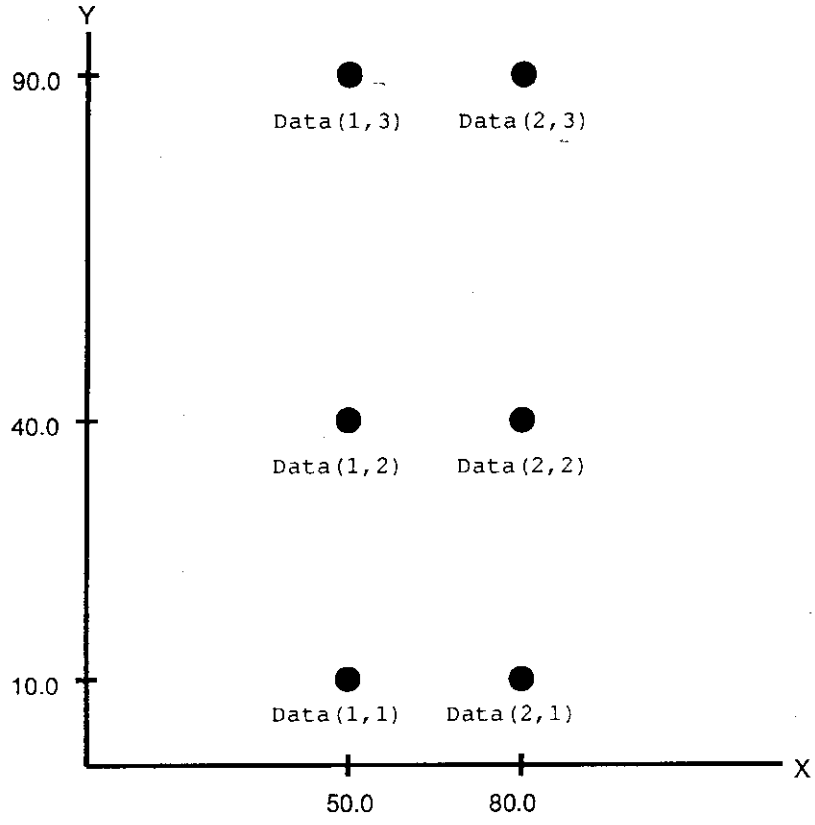

図 2.5　レクティリニアフィールドへのマッピング

2) irregular の場合

irregular の場合、各ノードのデータは、各ノードに対して決めた座標値に従ってフィールドにマッピングされます。

データ例 example を図 2.6に示すように座標値を決めた２次元のフィールド（フィールドの各軸をそれぞれｘ軸、ｙ軸とする）にマッピングする場合、各ノードのデータは以下に示すように並びます。

コロンの左側にある数字（実数）が座標値を示し、コロン（：）以降、そのデータの説明であり、実際にはありません。

```
5.0     : Data(1,1) のｘ座標
10.0    : Data(2,1) のｘ座標
10.0    : Data(1,2) のｘ座標
30.0    : Data(2,2) のｘ座標
20.0    : Data(1,3) のｘ座標
50.0    : Data(2,3) のｘ座標
5.0     : Data(1,1) のｙ座標
20.0    : Data(2,1) のｙ座標
30.0    : Data(1,2) のｙ座標
30.0    : Data(2,2) のｙ座標
50.0    : Data(1,3) のｙ座標
5.0     : Data(2,3) のｙ座標
```

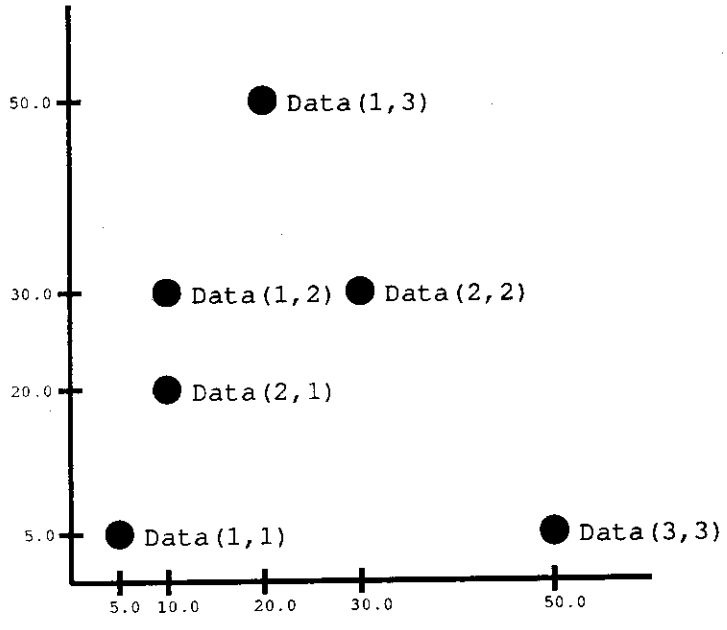

図 2.6 イレギュラーフィールドへのマッピング

(3) print field モジュール

print field モジュールは、フィールドデータを出力する read field モジュール等と接続し、モジュールが出力したフィールドデータの内容を表示します。print field モジュールを使うことで、モジュール間でデータの受渡しを正しく行なっているのかを調べることができます。

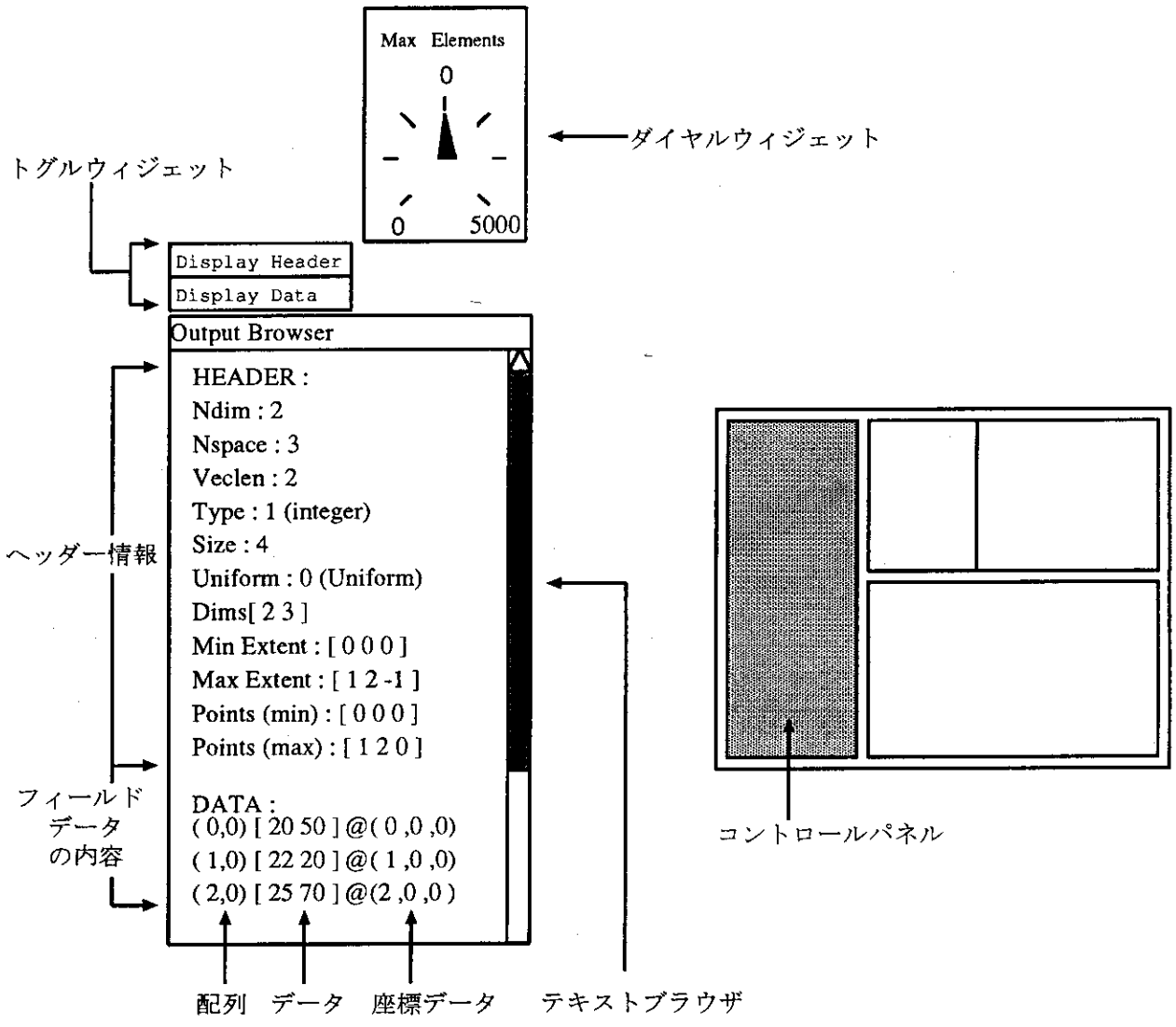

図 2.7 print field モジュールのコントロールウィジェット

以下に、print field モジュールの使用方法を、read field モジュールで読み込んだフィールドデータの表示を例に説明します。

1) モジュールパレットからワークスペースに read field モジュールを移動して、ファイルブラウザでファイルを選択しデータを読み込みます。

2) 同様にワークスペースに print field モジュールを移動します。

3) read field モジュールの出力ポートと print field モジュールの入力ポートを接続します。

4) コントロールパネル上の print field ボタンをクリックすると、コントロールウィジェット（テキストブラウザ、ダイヤルウィジェット、トグルウィジェット）を図2.7のように表示します。

5) テキストブラウザには read field モジュールが出力したフィールドデータの内容を表示します。フィールドデータの内容には、ヘッダー情報とデータ領域、座標領域の内容を表示します。データ領域、座標領域の内容には左から、配列、データ、座標データの順に表示されます。ヘッダー情報には以下のような内容を表示します。

Ndim	(次元数)
Dims	(各次元のサイズ)
Nspace	(フィールドの次元数)
Veclen	(ベクトルの長さ)
Type	(データタイプ)
Uniform	(フィールドのタイプ)
Data	(データ領域と座標領域の内容)

6) ダイヤルウィジェットでテキストブラウザに表示するデータの数（Max 5000）を設定します。

7) トグルウィジェットの Display Header ボタンをクリックしてオンにすると、図2.7のようにヘッダー情報を表示します。

8) トグルウィジェットの Display Data ボタンをクリックしてオンにすると、図2.7のようにフィールドの内容を表示します。

2.1.2 フィールドデータの取り込み

(1) ASCII記述ファイル

ここでは、ユーザの作成したデータをフィールドデータのフォーマットで取り込む方法について説明します。read field モジュールはフィールドデータのフォーマットで書かれたデータを取り込む以外に、ASCIIで書かれたデータ、FORTRANの書式なし、または、全てバイナリのデータをAVSフィールドデータとしてＡＶＳに取り込むことができます。read field モジュールでこれらのデータ（ASCIIで書かれたデータ、FORTRANの書式なしで書かれたデータ、全てバイナリのデータ）を取り込むには、ユーザの作成したデータを保存したファイルとは別にASCII記述ファイル（拡張子.fld）を用意します。

以下にASCII記述ファイルの内容を説明します。

ASCII記述ファイルの前半部分は以下のようになります。内容については、フィールドデータのフォーマットのASCIIヘッダーの部分を参照して下さい。

```
# AVS
# field file
ndim=2
dim1=2
dim2=3
nspace=2
veclen=4
data=integer
field=uniform
```

以上の記述の後にデータを参照する方法を記述します。

1) データに関する情報の読み込みのための記述
   データ部分の参照の方法は、以下のように記述します。
   variable l file=filename filetype=type skip=n offset=m stride=p

l: データを区別するための番号

file: データを保存しているファイルの絶対パス名

filetype: データを保存しているファイルのタイプ
ASCIIファイルのとき file=ascii、バイナリファイルのとき file=binary

skip: 読み飛ばす行数またはバイト数（記述しないと skip=0）

file=ascii のとき: データ読み込みの際に最初に読み飛ばす行数

file=binary のとき: データ読み込みの際に最初に読み飛ばすバイト数

offset: 最初のデータを読み込むまでに読み飛ばすデータ数（記述しないと offset=1） ascii のときのみ使用します。最初のデータを読みはじめる前に、読み飛ばすデータ数（座標データを含む）を記述します。ASCIIファイルでは、データ（座標データを含む）は、１つ以上のスペースで区切られていなければなりません。

stride: 記述しないと stride=1 になります。

file=ascii のとき: 次のデータを読み込むまでに読み飛ばすデータ数

file=binary のとき: 次のデータを読み込むまでに読み飛ばすバイト数

2) 座標情報に関する情報の読み込みのための記述

座標情報に関する情報の読み込みのための記述は、上のデータに関する情報の読み込みのための記述の variable を coord に変更したものです。coord 以降の記述の仕方は、データに関する情報の読み込みのための記述と同じです。

coord l file=filename filetype=type skip=n offset=m stride=p

(2) ASCII 記述ファイルの例

以下のデータをもとに実際に ASCII 記述ファイルの作成を作成します。なお、データは以下のような状態でファイル (data_field.dat) に保存されているとします。

```
  X     Y   A    B
0.0   0.0  11    1
0.8   0.1  11    1
 .     .   .    .
 .     .   .    .
8.0   7.0  11    1
9.0   7.0  11    1
```

フィールドデータとして data_field.dat を読み込むために、ASCII 記述ファイル (data_field.fld) を以下のように作成します。ネットワーク中の read field モジュールでの ASCII 記述ファイル (data_field.fld) を読み込むと、read field モジュールは、続いて ASCII 記述ファイルの内容に従って data_field.dat を読み込みます。

```
# AVS field file
#
#
ndim=2
dim1=10
dim2=8
nspace=2
veclen=2
data=integer
field=irregular
variable 1 file=/home/guest/avsuser/data/data_field.dat filetype=ascii
 skip=1 offset=2 stride=4
variable 2 file=/home/guest/avsuser/data/data_field.dat filetype=ascii
 skip=1 offset=3 stride=4
coord 1 file=/home/guest/avsuser/data/data_field.dat filetype=ascii
 skip=1 offset=0 stride=4
coord 2 file=/home/guest/avsuser/data/data_field.dat filetype=ascii
 skip=1 offset=1 stride=4 (return) ←リターンキーを入れるのを忘れない
```

(3) ASCII記述ファイルを用いたネットワーク例

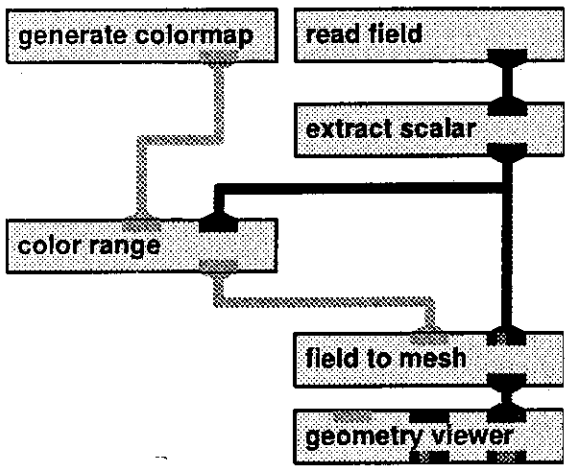

図 2.8 ネットワーク data_field_mesh.net

(4) write field モジュール

write field モジュールは、ASCII記述ファイルとデータや座標データが保存されているファイルを1つのファイルにして保存します。write field モジュールによって保存されたファイルはフィールドデータのフォーマットとなります。そのため、read field モジュールで、ASCII記述ファイルを用いた時よりも高速に読み込みます。以下に write field モジュールを使い、ASCII記述ファイルとデータと座標データが保存されているファイルをフィールドデータのフォーマットにした1つのファイルに保存する方法を説明します。

1) write field モジュールを使い、以下のようなネットワークを作成します。

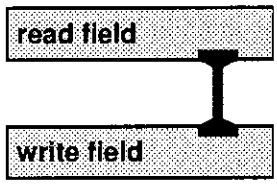

図 2.9 write field モジュール

2) ネットワークコントロールパネルの read field ボタンをクリックするとファイルブラウザを表示するので、作成したASCII記述ファイルをクリックします。

3) 同様にネットワークコントロールパネルの write field ボタンをクリックするとファイルブラウザを表示します。次にファイルブラウザの New File ボタンをクリックすると図2.10のようなウィンドウを表示します。

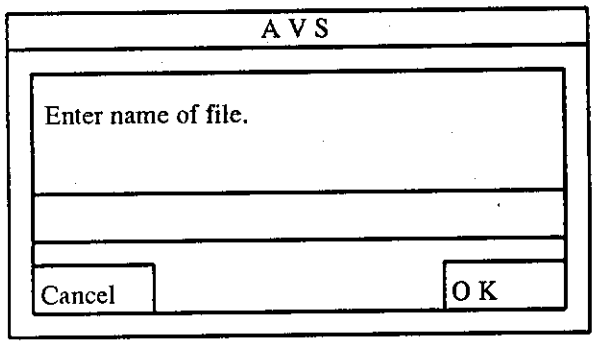

図 2.10 ファイル名の入力

4) ファイル名を入力し、ok をクリックするとファイルに保存します。

2.2 ジオメトリデータ

ジオメトリデータは3次元の形状を記述した幾何データです。フィールドデータや UCD データのような生データをディスプレイ上に表示するためには、既存のモジュールによってジオメトリデータに変換します。モジュール自身でジオメトリデータを生成するものもあります。さらに、データをＡＶＳ規定のフォーマットによって記述し、フィルターと呼ばれる変換用のプログラムによってジオメトリデータを生成する方法があります。ジオメトリデータは、read geom モジュール又は、ジオメトリビューワーサブシステムで取り込むことができます。

ここでは、フィルターによるジオメトリデータの作成方法を示します。

2.2.1 フィルターによるジオメトリデータの作成

ＡＶＳ規定のフォーマットで作成したデータをジオメトリデータに変換するフィルターには、以下のようなものがあります。なお、フィルターの実行ファイルは /usr/avs/bin にあります。またフィルターのソースファイルは /usr/avs/filter にあります。

ファイルフォーマット	実行ファイル	ソースファイル
Polyhedron	polyh_to_geom	polyh.c
Polygon	polyg_to_geom	polygon.c polygon.f
Mesh	mesh_to_geom	mesh.c mesh.f
Sphere	sphere_to_geom	sphere.c

以下のようなＡＶＳ規定のフォーマットで書かれたデータ（ファイルexample_polyhに保存されている）をフィルターpolyh_to_geomを使いジオメトリデータに変換する方法を示します。フィルターpolyh_to_geomは以下に示すデータから図2.11のようなポリゴンを組み合わせた幾何形状を記述したジオメトリデータに変換します。データは、各ポリゴンの頂点の座標と並びが必要です。コロンの左側の文字及び数字がデータ等を表し、コロン（：）以降はそのデータ等の説明であり実際にはデータ領域にはありません。

```
facet                 : タイプ(facet, smoothのどちらかを指定)
8                     : オブジェクトの頂点数
-2.0  1.0  1.0        : 頂点1の座標(float)
-2.0 -1.0  1.0        : 頂点2の座標(float)
 1.0 -1.0  1.0        : 頂点3の座標(float)
 1.0  1.0  1.0        : 頂点4の座標(float)
-2.0  1.0 -1.0        : 頂点5の座標(float)
-2.0 -1.0 -1.0        : 頂点6の座標(float)
 1.0 -1.0 -1.0        : 頂点7の座標(float)
 1.0  1.0 -1.0        : 頂点8の座標(float)
4                     : 面1の頂点数
1 2 3 4               : 面1の頂点の並び
4                     : 面2の頂点数
1 4 8 5               : 面2の頂点の並び
4                     : 面3の頂点数
1 5 6 2               : 面3の頂点の並び
4                     : 面4の頂点数
2 6 7 3               : 面4の頂点の並び
4                     : 面5の頂点数
3 7 8 4               : 面5の頂点の並び
4                     : 面6の頂点数
8 7 6 5               : 面6の頂点の並び
```

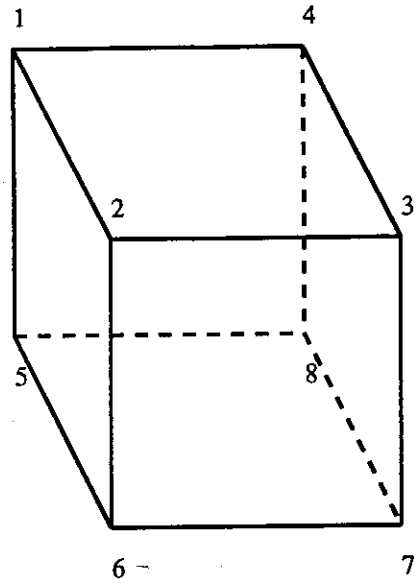

図 2.11 データ example_polyh

フィルターの使用方法

フィルター polyh_to_geom によってデータを以下のようにしてジオメトリデータに変換します。フィルターの使用方法はどのフィルターも同じです。

```
% polyh_to_geom          < examle_polyh   > examle_polyh.geom
```
(フィルタの実行ファイル名)　(入力ファイル名)　(出力ファイル名)

以下に polyh_to_geom 以外のフィルターとデータのフォーマットを示します。

1. mesh_to_geom
   2次元格子のメッシュを表示するためのジオメトリデータを生成します。

   (a) scalar
   各格子点にスカラー量 Z(i,j) が必要です。スカラー量は、浮動小数点 (float) です。データのフォーマットを以下に示します。

```
scalar 3 4    : タイプ X軸方向格子数 Y軸方向格子数
0.0 0.0 0.0   : Z(1,1) Z(1,2) Z(1,3)
0.0 1.0 0.0   : Z(2,1) Z(2,2) Z(2,3)
0.0 1.0 0.0   : Z(3,1) Z(3,2) Z(3,3)
0.0 0.0 0.0   : Z(4,1) Z(4,2) Z(4,3)
```

(b) vertex

各格子点に座標値 X(i,j), Y(i,j), Z(i,j) が必要です。座標値は、浮動小数点 (float) です。データのフォーマットを以下に示します。

```
vertex n m
X(1,1) Y(1,1) Z(1,1)
X(1,2) Y(1,2) Z(1,2)
X(1,3) Y(1,3) Z(1,3)
  ......................
X(m,n) Y(m,n) Z(m,n)
```

(c) color

各格子点に座標値とカラー値が必要です。座標値とカラー値は、浮動小数点 (float) です。データのフォーマットを以下に示します。

```
color n m
X(1,1) Y(1,1) Z(1,1) R(1,1) G(1,1) B(1,1)
X(1,2) Y(1,2) Z(1,2) R(1,2) G(1,2) B(1,2)
  ..............................................
X(m,n) Y(m,n) Z(m,n) R(m,n) G(m,n) B(m,n)
```

2. polyh_to_geom

ポリゴンを生成します。データのフォーマットについては、データファイル例 example_polyh を参照して下さい。

3. sphere_to_geom

球を生成します。球の中心座標値、半径、カラーが必要です。存在範囲には通常 0.0 が入ります。

```
0.0 0.0 0.0                          : 球の存在範囲
0.0 0.0 0.0                          : 球の存在範囲
2                                    : 球の個数
-1.0 0.0 0.0 1.0 0.5 0.5 0.5   : 中心座標 (X,Y,Z) 半径 カラー (R,G,B)
 1.0 0.0 0.0 1.0 1.0 0.0 1.0   : 中心座標 (X,Y,Z) 半径 カラー (R,G,B)
```

4. polyg_to_geom

ポリゴンを生成します。各ポリゴンの頂点数と座標値が必要です。データのフォーマットを以下に示します。以下のデータでは立方体を表示するジオメトリデータを生成します。

```
smooth              : タイプ (facet, smooth のどちらかを指定)
4                   : ポリゴンの頂点数
 0.0  0.0  0.0      : 各頂点の座標 (float)
 0.0 -5.0  0.0      : 各頂点の座標 (float)
 5.0 -5.0  0.0      : 各頂点の座標 (float)
 5.0  0.0  0.0      : 各頂点の座標 (float)
```

```
4                    : ポリゴンの頂点数
 5.0   0.0   0.0     : 各頂点の座標(float)
 5.0  -5.0   0.0     : 各頂点の座標(float)
 5.0  -5.0  -5.0     : 各頂点の座標(float)
 5.0   0.0  -5.0     : 各頂点の座標(float)

4                    : ポリゴンの頂点数
 5.0   0.0  -5.0     : 各頂点の座標(float)
 5.0  -5.0  -5.0     : 各頂点の座標(float)
 0.0  -5.0  -5.0     : 各頂点の座標(float)
 0.0   0.0  -5.0     : 各頂点の座標(float)

4                    : ポリゴンの頂点数
 0.0   0.0  -5.0     : 各頂点の座標(float)
 0.0  -5.0  -5.0     : 各頂点の座標(float)
 0.0  -5.0   0.0     : 各頂点の座標(float)
 0.0   0.0   0.0     : 各頂点の座標(float)

4                    : ポリゴンの頂点数
 0.0   0.0   0.0     : 各頂点の座標(float)
 5.0   0.0   0.0     : 各頂点の座標(float)
 5.0   0.0  -5.0     : 各頂点の座標(float)
 0.0   0.0  -5.0     : 各頂点の座標(float)

4                    : ポリゴンの頂点数
 0.0  -5.0   0.0     : 各頂点の座標(float)
 5.0  -5.0   0.0     : 各頂点の座標(float)
 5.0  -5.0  -5.0     : 各頂点の座標(float)
 0.0  -5.0  -5.0     : 各頂点の座標(float)
```

フィルターには、技術分野でよく使うデータのフォーマットをジオメトリデータに変換するものがあります。それらのフィルターを以下に示します。

ファイルフォーマット	実行ファイル	ソースファイル
Mathmatica ThreeScript	ts_to_geom	
Movie BYU	byu_to_geom	byu.c
Protein Data Bank	pdb_to_geom	pdb.c
UNC	ppoly_to_geom	ppoly.c
Wavefront	wfront_to_geom	wfront_to_geom

3．サブシステム

3.1　サブシステムの起動、終了

ＡＶＳには、ネットワークエディタ以外に３つのサブシステムがあります。

- ジオメトリビューワーサブシステム
  ３次元の幾何データ (＊＊＊.geom, ＊＊＊.scr) を表示させ、可視化したイメージの回転などの操作をするためのサブシステムです。

- イメージビューワーサブシステム
  イメージデータ (＊＊＊.x) を表示させ、可視化したイメージの拡大縮小の操作をするためのサブシステムです。

- グラフビューワーサブシステム
  グラフ（折れ線、棒グラフ）を表示するためのサブシステムです。

これら３つのサブシステムはこれまで説明してきたモジュール（geometry viewer モジュール、image viewer モジュール、graph viewer モジュール）としてネットワークと接続して使用すること以外に単独でサブシステムとしてその機能を使用できます。

3.1.1　サブシステムの起動方法

サブシステムの起動方法には、以下のような方法があります。

1. 直接シェルから起動する方法
   以下のコマンドで起動します。例えばジオメトリビューワーサブシステムを以下のコマンドで起動すると図 3.1を表示します。

```
% avs -image
% avs -graph
% avs -geometry
```

2. メインメニューから起動する方法
   ＡＶＳを起動し、メインメニュー（図 1.1）にある該当するボタンをクリックするとサブシステムが起動します。

3. 他のサブシステムから起動する方法
   各サブシステムのコントロールパネルの一番上にある Data Viewer ボタン（図 3.1）をクリックするとポップアップメニューにサブシステム名を表示します。マウスのボタンをクリックしたままで起動したいサブシステム名までカーソルを移動します。ボタンを離すとサブシステムを起動します。

4. ネットワークエディタの中から起動する方法
   ネットワーク内の image viewer モジュール, geometry viewer モジュール, graph viewer モジュールの右側にある正方形のボタンをマウスの左ボタンでクリックするとネットワークコントロールパネルの上にそれぞれのサブシステムのコントロールパネルを起動します。

3.1.2 サブシステムの終了方法

1. 直接シェルから起動した場合
   コントロールパネルの Exit ボタンをクリックするとシェルに戻ります。

2. メインメニューから起動した場合　コントロールパネルの Close ボタン（図 3.1）をクリックするとコントロールパネルをディスプレイ上から消去し、メインメニューに戻ります。

3. 他のサブシステムから起動した場合
   コントロールパネルの Close ボタン（図 3.1）をクリックするとコントロールパネルをディスプレイ上から消去します。

4. ネットワークエディタの中から起動する方法
   コントロールパネルの Close ボタンをクリックするとコントロールパネルをディスプレイ上から消去します。

3.2 ジオメトリビューワーサブシステム

ジオメトリビューワーサブシステムを操作するために必要な基礎（シーン、オブジェクトの階層、表示）について説明し、次に操作手順を説明します。

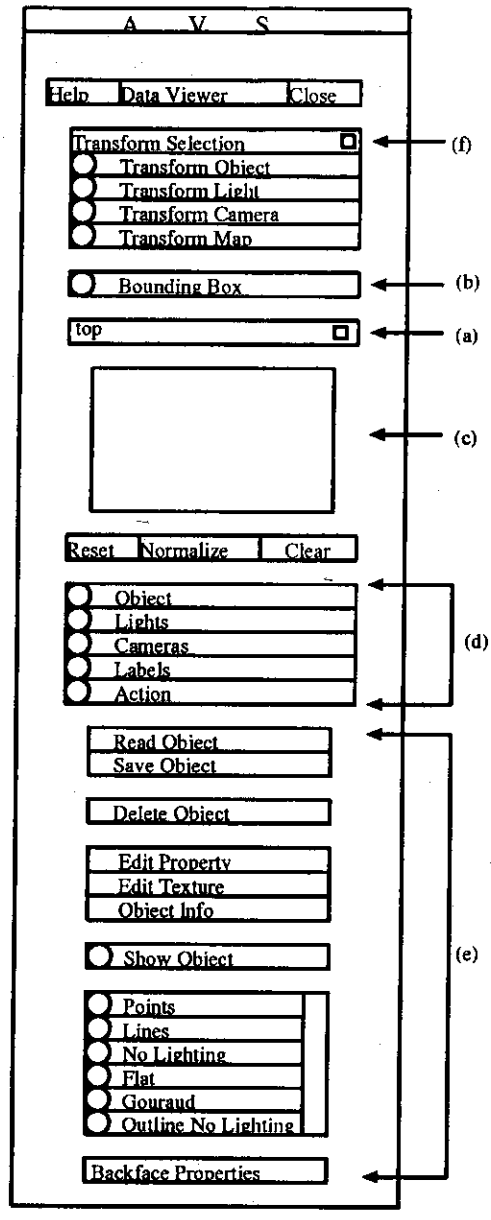

図 3.1 ジオメトリビューワーサブシステムのコントロールパネル

3.2.1 Scene(シーン)

一般に写真撮影をする場合、被写体をライトで照らしカメラで写真を撮ります。ＡＶＳの世界（ジオメトリビューワー）では、同じように被写体となるオブジェクト、光源であるライト、被写体を撮るカメラ、カメラのファインダーに相当するビューウィンドウ（ジオメトリビューワー）があります。オブジェクト、ライト、カメラ、ビューウィンドウをＡＶＳの世界ではシーンと呼びます。ワールド空間と呼ばれる座標空間の中にオブジェクトがあり、ビューウィンドウを介してのオブジェクトをみた場合、X、Y、Z軸の正方向がそれぞれの右方向、上方向、手前方向を指します。ワールド空間の中心である (x, y, z)=(0, 0, 0) がビューウィンドウの中心になります。

- オブジェクト
  オブジェクトはカメラで撮影される形状を指します。その形状は、カラー、透明、反射などの属性（サーフェス属性という）をもちます。

- ライト
  オブジェクトを照らす光のことです。複数のライトで１つのオブジェクトを照らすこともできます。また、各ライト毎に異カラーを設定できます。

- カメラ
  ワールド空間内のオブジェクトを撮るカメラです。カメラは撮った映像をビューウィンドウに表示します。複数のカメラで１つのオブジェクトを異なった角度から撮影し、それぞれ撮影した映像を複数のウィンドウに表示することができます。

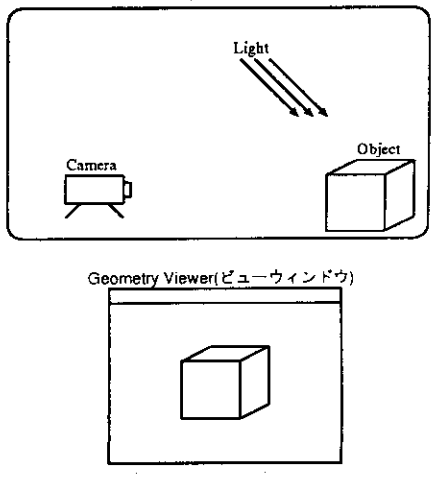

図 3.2 カメラ、ライト、オブジェクト、ビューウィンドウ

3.2.2 オブジェクトの階層

ビューウィンドウに表示した複数のオブジェクトは、２つの階層で管理されます。１つめの階層は、top という名前のオブジェクト名で表されるルートオブジェクトです。このルートオブジェクトの下の層に name.number という名前のオブジェクト名で表される個々のオブジェクトがあります。例えばジオメトリデータファイル sumple1.geom, sumple2.geom を読み込んだ場合のオブジェクト名は、sumple1.0、sumple2.1 となります。

階層構造によって同じシーンに複数のオブジェクトを読み込んだときにオブジェクトに対する

操作（回転や移動など）を、個々のオブジェクトに対して行なうのか、それとも、ワールド空間内にあるすべてのオブジェクトに対して行なうのかを選択できます

オブジェクトの階層を表示するには、コントロールパネルの図3.1の(a)のボタン（右側にある四角形のボタン）をクリックします。すると、Current Object Browser（図3.3）に階層構造を表示します。topをクリックすると全体のオブジェクトに対して操作をすることができ、個々のオブジェクト名をクリックするとそれぞれのオブジェクトに対して操作をすることができます。

Current Object Browserでクリックした操作の対象となるオブジェクトをカレントオブジェクトといいます。

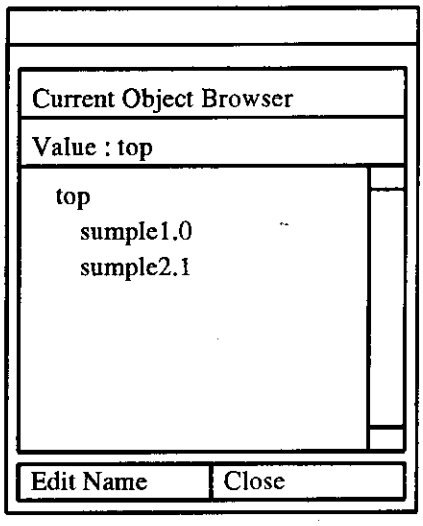

図 3.3 Current Object Browser

また、Current Object BrowserのEdit Nameボタン（図3.3）をクリックするとカレントオブジェクトの名前を変更することができます。

3.2.3 表示

オブジェクトの表示方法はgeometry viewerモジュールによる表示とジオメトリビューワーによる直接表示の2通りあります。

ビューウィンドウは、複数のジオメトリデータを読み込んで複数のオブジェクトを表示することができます。ジオメトリビューワーサブシステムとgeometry viewerモジュールは取り込んだ複数のオブジェクトを1つのビューウィンドウに表示したり、異なるビューウィンドウに表示することもできます。

(1) geometry viewerモジュールによる表示

1) geometry viewerモジュールでオブジェクトを表示するためには、ジオメトリデータを出力とするモジュールの下流モジュールとしてgeometry viewerモジュールを接続します。ジオメトリデータを出力とするモジュールの出力ポートは赤で示されています。

2) geometry viewerモジュールは、接続した上流モジュールからジオメトリデータを読み込んでビューウィンドウにオブジェクトを表示します。

(2) ジオメトリビューワーサブシステムによる直接表示

1) ジオメトリビューワーメニュー（図3.1の(d)）のObjectsボタンをクリックすると図3.1の(e)の部分にObjectsメニューを表示します。なお、初期設定ではジオメトリビューワーサブシステムを起動した時点でObjectsボタンが選択してあるのでObjectsボタンをクリックする必要はありません。

2) Read Objectボタンをクリックするとファイルブラウザを表示します。

3) マウスで読み込みたいジオメトリデータが保存されたファイルをクリックします。

4) ビューウィンドウが生成され、オブジェクトが表示されます。

ジオメトリビューワーサブシステムは、以下に示すようなジオメトリデータの読み込みと保存ができます。

種類	内容	拡張子
Geometryデータ	形状のみを記述したデータ	***.geom
Objectデータ	Geometryファイルの内容に任意の属性の記述を加えたデータ	***.scr
Sceneデータ	Objectファイルの内容にカメラの位置を加えたデータ	***.scr

- Geometryデータと Objectデータの読み込みと保存

  Objectsメニューの Read Objectボタン（図3.1）をクリックするとファイルブラウザが表示されます。そのファイルブラウザで Geometryデータ又は Objectデータを読み込みます。

  Objectsメニューの Save Objectボタン（図3.1）をクリックするとタイプインウィジェット（図1.21）を表示するのでファイル名を入力し、OKボタンをクリックします。この操作によりタイプインウィジェットで入力したファイルに保存されます。

- Sceneデータの読み込みと保存

  Camerasメニューの Read Sceneボタン（図3.1）をクリックするとファイルブラウザが表示されます。そのファイルブラウザで Sceneデータを読み込みます。なお Objectsメニューの Read Objectで読み込むこともできます。

  Camerasメニューの Save Sceneボタン（図3.1）ををクリックするとタイプインウィジェットを表示するのでファイル名を入力し、OKボタンをクリックします。この操作によりタイプインウィジェットで入力したファイルに保存されます。

ＡＶＳはビューウィンドウに Objectデータと Sceneデータを表示する場合に Geometryデータを参照して表示します。そのため、オブジェクトの形状を記述した Geometryデータを保存しているファイル(***.geom)を消去しないようにして下さい。

また、複数のオブジェクトの位置やカラーやカメラ操作などのビューウィンドウでの表示を正確に保存するには、Save Sceneを使用して下さい。

3.3 ジオメトリビューワーサブシステムの操作手順

3.3.1 ジオメトリビューワーサブシステムの操作概要

ジオメトリビューワーサブシステムでは、ジオメトリデータを読み込んでビューウィンドウにオブジェクトを表示する以外に表示したオブジェクトに対して以下の操作をすることができます。

- 幾何変換操作
  オブジェクト、ライト、カメラに対して移動、拡大 / 縮小、回転などの幾何変換
- オブジェクトに対する操作
  表面属性の設定、シェーディングタイプの変更、テクスチャマッピング
- ライトに対する操作
  複数のライトの設定、ライトの色の設定
- カメラに対する操作
  複数のカメラの設定

3.3.2 Bounding Box

通常、ジオメトリビューワーでは、オブジェクトを移動、拡大、縮小した時に途中経過のオブジェクトも順次ビューウィンドウに表示します。しかし、テクスチャマッピングなどを施して表示される複雑なオブジェクトの表示は非常に遅くなります。このような場合、Bounding Box ボタン（図3.1の (b)）をクリックすると移動、回転などの途中経過を表示せずに操作後のオブジェクトを表示します。Bounding Box ボタンは、特に描画速度の遅いシステムに便利です。

Bounding Box は、初期設定でオンに指定しておくことができます。初期設定を行なう場合はスタートアップファイルに以下の行を加えて下さい。

```
BoundingBox  1
```

3.3.3 幾何変換操作

ジオメトリビューワーサブシステムでは、オブジェクト、ライト、カメラ、テクスチャマッピングの4つを対象として移動、回転及び拡大/縮小の幾何変換を行なうことができます。幾何変換を行なうカレントオブジェクトの選択、幾何変換の対象を選択する方法を説明し、その後に幾何変換の操作手順を示します。

(1) カレントオブジェクトの選択

幾何変換の対象となるカレントオブジェクトを以下のようにして選択します。カレントオブジェクトの選択方法には、2通りあります。

1. 図3.1の(a)をクリックすると、Current Object Browser（図3.3）を表示します。Current Object Browserに表示されているオブジェクト名をクリックすると、そのオブジェクトがカレントオブジェクトになります。
2. ビューウィンドウに表示されている任意のオブジェクトの上にカーソルを移動してマウスの左ボタンでクリックするとそのオブジェクトがカレントオブジェクトとなります。再度そのオブジェクトをクリックするとtopがカレントオブジェクトになります。例ではsumple1.0、sumple2.1のいずれかのオブジェクトをクリックします。

カレントオブジェクトに選択したオブジェクト名が図3.1の(a)に表示されます。また、図3.1の(c)には、選択したオブジェクト自身を表示します。

(2) 幾何変換の対象の選択

幾何変換を行なうことのできるものには、以下のようなものがあります。

- オブジェクトの幾何変換
  Transform Selection（図3.1）のTransform Objectボタンをクリックしてから幾何変換を行ないます。初期設定では、Transform Objectが選択されています。
- ライトの幾何変換
  Transform Selection（図3.1）のTransform Lightボタンをクリックしてから幾何変換を行ないます。
- カメラの幾何変換
  Transform Selection（図3.1）のTransform Cameraボタンをクリックしてから幾何変換を行ないます。
- テクスチャーの幾何変換
  Transform Selection（図3.1）のTransform Textureボタンをクリックしてから幾何変換を行ないます。

(3) 幾何変換の操作手順

幾何変換を行なう方法は、2通りあります。マウスによる幾何変換とキーボードによる幾何変換です。

マウスによる幾何変換では、マウスの動きに応じて簡単に幾何変換（移動、回転、拡大/縮小）を行なうことができます。

キーボードによる幾何変換では、回転角度や移動の距離をキーボードで入力することで正確な幾何変換を行なうことができます。

1. マウスによる幾何変換

- 移動

マウスの右ボタンをクリックしたままでカーソルを平行移動するとビューウィンドウ内で以下に示すように平行移動（2次元）します。カレントライト、カレントカメラ、カレントテクスチャーの選択方法は、後のジオメトリビューワーメニューを参照して下さい。

＊ オブジェクト

カレントオブジェクトがビューウィンドウ内で平行移動します。

＊ ライト

光源であるライトのうち幾何変換するライト（カレントライト）がビューウィンドウ平面上で平行移動します。ライトが平行光線または、双方向からの平行光線である場合は、ライトを示すマークを移動するだけなのでオブジェクトの見え方は変わりません。ライトを示すマークを表示するには、ジオメトリビューワーメニューのLightsボタン（図3.6）をクリックしLightsメニューを表示させShow Lightsボタン（図3.6）をクリックします。（オンにする）

＊ カメラ

幾何変換するカメラ（カレントカメラ）をビューウィンドウ平面上で平行移動します。

＊ テクスチャー

カレントオブジェクトのテクスチャーの位置をウィンドウ平面上で平行移動します。

- 回転

マウスの中ボタンをクリックしたままでカーソルを移動するとビューウィンドウ内で以下に示すように回転します。

＊ オブジェクト

カレントオブジェクトが3次元空間で回転します。

＊ ライト

カレントライトの位置又は、方向が回転します。

＊ カメラ

カレントカメラの位置が回転します。実際にはオブジェクトが回転しているように見えます。

＊ テクスチャー

カレントテクスチャーの位置が回転します。

・ 拡大、縮小

キーボードのシフトキーとマウスの中ボタンをクリックしたままカーソルを移動すると以下に示すように拡大（カーソルを上方向に移動）、縮小（下方向に移動）を行ないます。

＊ オブジェクト

カレントオブジェクトが拡大 / 縮小します。

＊ ライト

カレントライトを示すマークが拡大 / 縮小します。

＊ カメラ

カレントカメラのズームが拡大 / 縮小します。

＊ テクスチャー

カレントテクスチャーが拡大 / 縮小します。

2. キーボードによる幾何変換

キーボードによる幾何変換を行なう場合は、まず、図 3.1の (f) の右側にある正方形のボタンをクリックします。クリックすると以下のような Transformation Option パネル（図 3.4）を表示します。

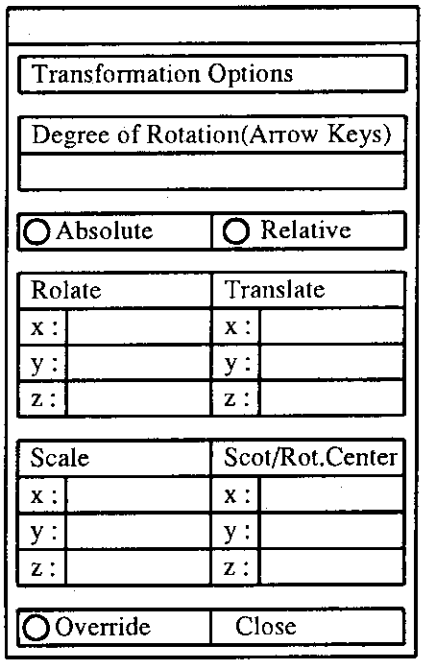

図 3.4 Transformation Option パネル

- 矢印キーによる回転

  まず、Transformation Option パネル（図 3.4）の Degree of Rotation の下に回転角（単位：度）を入力します。初期設定では 45.0 度です。次にカーソルをジオメトリビューワーに移動し、矢印キーで以下のような回転を行なうことができます。

  * 左右の矢印キー
    カレントのオブジェクト、ライト、カメラが Y 軸回りに回転します。
  * 上下の矢印キー
    カレントのオブジェクト、ライト、カメラが X 軸回りに回転します。
  * Shift キーと左右の矢印キー
    カレントのオブジェクト、ライト、カメラが Z 軸回りに回転します。

- 相対的と絶対的な幾何変換

  Relative と Absolute は、現在の位置に対して相対的 (Relative) な位置に幾何変換を行なうのか、それとも絶対的な位置 (Absolute) に幾何変換するのかを選択します。

  まず、幾何変換する対象（オブジェクト、ライト、カメラ）を選びます。Transform Selection の下のボタン（図 3.1）で幾何変換する対象（オブジェクト、ライト、カメラ）を選択します。

  次に Transform Selection の右側の正方形のボタンをクリックすると Transformation Option パネル（図 3.4）を表示します。Transformation Option パネルの Absolute ボタンをクリックするとカレントのオブジェクト、ライト、カメラの現在の状態を Transformation Option パネル上に表示します。Rotate を除く全ての項目に新しい数値を入力して幾何変換を行なうことができます。また、Relative ボタンをクリックすると現在の状態から入力した値を増分として幾何変換を行うことができます。

  以下に幾何変換の方法を示します。

  幾何変換は、キーボードで数値を入力することによって行います。Absolute ボタン、または、Relative ボタンをクリックします。カーソルを変更したい項目のスペースまで移動してキーボードで数値を入力します。入力したらリターンキーを押してマウスのカーソルを項目のスペースから外すと幾何変換を行ないます。

  * Rotate
    カレントのオブジェクト、ライト、カメラが Scale/Rot.Center で設定した座標を中心として回転します。
  * Translate
    カレントのオブジェクト、ライト、カメラが平行移動します。
  * Scale
    カレントのオブジェクト、ライト、カメラが Scale/Rot.Center で設定した座標を中心として拡大縮小します。
  * Scale/Rot.Center
    カレントのオブジェクト、ライト、カメラ自身の中心点を設定することができます。

- その他の幾何変換
  - * Reset
    カレントオブジェクト、カレントライト、カレントカメラを最初にビューウィンドウを起動したときの状態まで戻します。ただし、オブジェクトのカラー、サーフェス属性、中心点、レンダリングモードはそのままです。
  - * Normalize
    ビューウィンドウいっぱいにカレントオブジェクトが表示されるように大きさを拡大 / 縮小します。

3.3.4 ジオメトリビューワーメニュー

ジオメトリビューワーサブシステムは、図3.1の (d) に、ジオメトリビューワーメニューを表示します。ジオメトリビューワーメニューにある機能を使って可視化したイメージを作成します。ジオメトリビューワーメニューは、以下の5つのメニューがあります。ここではObjectsメニュー、Lightsメニュー、Camerasメニュー、Labelsメニューについて説明します。

1. Objectsメニュー
2. Lightsメニュー
3. Camerasメニュー
4. Labelsメニュー
5. Actionメニュー

(1) Objects メニュー

Object ボタンをクリックした場合は、図3.1の (e) の部分に以下のような Objects メニュー（図3.5）を表示します。Objects メニューでは、オブジェクトの表示、移動、属性の変更などを行ないます。

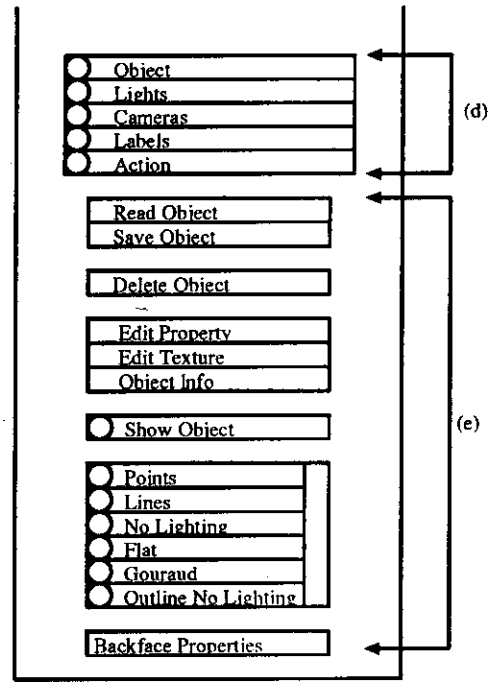

図 3.5 Objects メニュー

1) Read Object

Read Object ボタンをクリックするとファイルブラウザを表示します。任意のジオメトリデータをクリックするとデータを読み込みビューウィンドウにオブジェクトを表示します。

2) Save Objects

Save Objects は、カレントオブジェクトを Object データ（ファイル名***.scr）として保存します。このとき、形状の他にカラーや反射などの属性を保存します。Save Objects ではライトやカメラの位置などは保存できません。これらを全て保存したい場合は Camera メニューの Save Scene を使います。

3) Delete Object

Delete Object ボタンをクリックするとカレントオブジェクトをビューウィンドウから消去します。

4) Edit Property

Edit Property ボタン（図3.5）をクリックすると属性編集ウィンドウを表示します。属性編集ウィンドウではカレントオブジェクトのカラーや反射などの属性を変更します。反射の属性とは、オブジェクトに光を当てた場合にオブジェクトが光をどのように反射するのかを定義したものです。

各属性値は属性編集ウィンドウのスライダーウィジェット（図1.22）で設定します。設定できる属性には以下のようなものがあります。

- Red, Green, Blue
  それぞれ赤、緑、青の量を調節してオブジェクトのカラーを設定します。
- Hue, Saturation, Value
  色相、彩度、明度の量を調節してオブジェクトのカラーを設定します。
- Amb
  オブジェクトが反射するアンビエントライト（環境光）の量を設定します。アンビエントライトとは、オブジェクトの表面のどの部分にも同じ影響を与えるライトのことです。
- Diff
  オブジェクトが反射するアンビエントライト以外のライト（点光源、平行光線）の量を設定します。
- Spec, Gloss, Metal
  スペキュラーハイライトの明るさ、鋭さ、色を設定します。
- Trans
  オブジェクトの透明度を設定します。

5) Edit Texture

2次元のイメージをオブジェクトの表面にマッピングする場合に使用します。（テクスチャマッピング）

6) Object Info

Object Info ボタンをクリックするとウィンドウを表示し、カレントオブジェクトに関する情報を表示します。

7) Show Object

Show Object ボタンをクリックするとカレントオブジェクトを非表示したり、再表示したりします。

8) 以下は、カレントオブジェクトのレンダリング方法を指定します。

Points	オブジェクトを点の集合で描画
Lines	オブジェクトをワイヤーフレームで描画
No Lighting	オブジェクトをカラーだけで描画
Flat	オブジェクトをフラットシェーディングで描画
Gouraud	オブジェクトをグローシェーディングで描画
Outline No Lightning	オブジェクトを Lines + No Lighting で描画
Outline Flat	オブジェクトを Lines + Flat で描画
Outline Gouraud	オブジェクトを Lines + Gouraud で描画
inherit	オブジェクトを親オブジェクトのレンダリング方法で描画

(2) Lights メニュー

Lights ボタンをクリックした場合は、図3.1の (e) の部分に以下のような Lights メニューを表示します。Lights メニューでは、光源の数や種類、移動、色の変更をします。

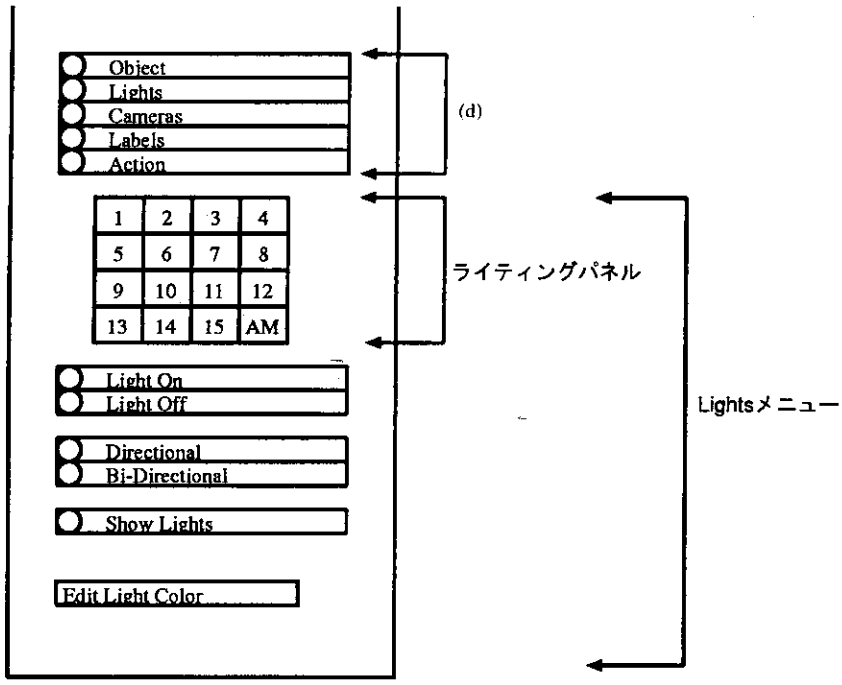

図 3.6 Lights メニュー

1) ライティングパネル

ライティングパネルの数字は、１６個の光源を表します。１つのシーンに対して１５個までの平行光線が設定できます。１６個目のＡＭではアンビエントライトを設定することができます。初期設定では、１番のライトとアンビエントライトがオンになっています。

2) Light On/Light Off

カレントライトのオンとオフを切替えます。ライティングパネルの数字の書いてある任意のボタン（アンビエントライとを含む）をクリックすることによって数字が白くなりカレントライトを選択できます。Light On をクリックするとカレントライトがオンとなり、Light Off をクリックするとカレントライトがオフとなります。

3) Directional, Bi-Directional

Directional ボタンをクリックするとカレントライトを平行光源として設定し、Bi-Directional ボタンをクリックするとカレントライトを双方向からのとして設定します。ただし、カレントライトを双方向からの平行光源にする場合は、１番～７番のライトを使います。これは、８番～１４番のライトを双方向からの平行光源のもう一方のライトとして使うためです。

4) Show Lights
Show Lights ボタンをクリックするとカレントライトの光源の方向を示す矢印を表示します。

5) Edit Light Color
カレントライトのカラーを設定します。カラーの設定は、Objects メニューの Edit Property を参照して下さい。

6) カレントライトの移動、回転
カレントライトの移動、回転は、以下のようにして行ないます。

6-1) マウスで Transform Light ボタン（図 3.1）をクリックします。

6-2) Show Lights ボタン（図 3.6）をオンするとカレントライトの方向を示す矢印を表示します。

6-3) マウスの右ボタンを使ってカレントライトを平行移動し、中ボタンを使ってライトの方向を回転させます。

(3) Cameras メニュー

Cameras ボタンをクリックした場合は、図 3.1の (e) の部分に以下のような Cameras メニューを表示します。

Cameras メニューでは、ワールド空間のオブジェクトを写すカメラの設定を行ないます。カメラで写した映像はビューウィンドウに表示します。また複数のカメラをセットすれば同じオブジェクトを別の角度から写せます。初期状態では、カメラは、(0, 0, 100) 付近にセットされて Z 軸に沿って原点の方向に向かってワールド空間の X が -5 ～ 5、Y が -5 ～ 5、Z が -100 ～ 100 の範囲を写しています。

また、カレントカメラで写した映像を表示しているビューウィンドウは、枠の色が赤となります。

カレントカメラは任意のウィンドウをクリックすることによって選択します。

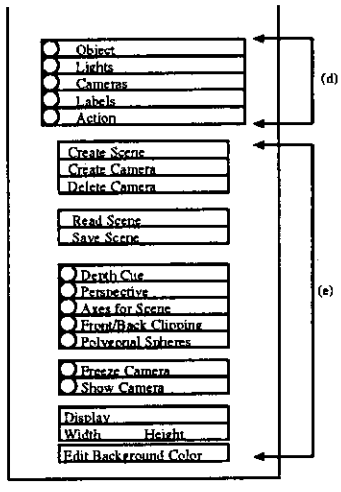

図 3.7 Cameras メニュー

1) Create Scene

Create Scene ボタン（図3.7 (e)）をクリックすると何も表示していない新しいビューウィンドウを表示します。Create Scene ボタンを使えば、別のオブジェクトを表示する新しいビューウィンドウを生成できます。

2) Create Camera

Create Camera ボタン（図3.7 (e)）をクリックすると新しいビューウィンドウを表示します。このビューウィンドウにはカレントウィンドウに表示しているオブジェクトと同じものを新たなカメラで写した場合のビューウィンドウを表示します。

そのために、片方のビューウィンドウで変更（オブジェクトの回転と移動、オブジェクトのサーフェス属性の変更、ライトのカラーと位置の変更など）したことは、同じオブジェクトを表示している全てのビューウィンドウで変更されます。

3) Delete Camera

Delete Camera ボタン（図3.7 (e)）をクリックするとカレントカメラを削除し、ビューウィンドウを消去します。

4) Read Scene/Save Scene

Scene データの読み込みや保存を行なうことができます。ファイル名は＊＊＊.scr です。

5) Depth Cue

Depth Cue ボタン（図3.7 (e)）をクリックすると視点から遠いものを暗く遠近表示します。

6) Perspective

Perspective ボタン（図3.7 (e)）をクリック（オンにする）すると透視投影法（視野角４５度）で表示します。再度クリックする（オフにする）と平行投影法で表示します。

7) Axes for Scene

Axes for Scene ボタン（図3.7 (e)）をクリックするとワールド座標系の座標軸を表示します。

8) Front/Back Clipping

Front/Back Clipping ボタン（図3.7 (e)）をクリックするとクリッピングを行ないます。クリッピングを行なうと視野にあまりにも近いオブジェクト、または、あまりにも遠いオブジェクトを表示しません。

9) Polygonal Spheres

Polygonal Spheres ボタン（図3.7 (e)）をクリックすると球をポリゴンで近似表示します。

10) Freeze Camera
同じオブジェクトを写している複数のビューウィンドウは、1つのビューウィンドウの変更が、全てのウィンドウにおいての変更となります。Freeze Camera ボタン（図3.7 (e)）をクリックしたビューウィンドウは、この変更の影響を受けずに映像を固定します。

11) Show Camera
Show Camera ボタン（図3.7 (e)）をクリックするとウィンドウを非表示にします。もう1度クリックすると再表示します。

12) Edit Background Color
カレントウィンドウの背景の色を変更します。カラーの設定は、Objects メニューの Edit Property を参照して下さい。

13) カメラの移動、回転
カレントカメラの移動や回転は、以下のようにして行ないます。オブジェクトを固定してカメラだけを移動、回転します。実際には、オブジェクトが移動、回転したように見えます。

1) マウスで Transform Camera ボタン（図3.1）をクリックします。
2) オブジェクトの場合と同じように、マウスの右ボタンを使ってカレントカメラを移動し中ボタンを使ってカレントライトを回転させます。

(4) Labels メニュー

Labels ボタンをクリックした場合は、図3.1の (e) の部分に以下のような Labels メニュー（図3.8）を表示します。Labels メニューでは、オブジェクトに1つ以上のラベルを付けることができます。

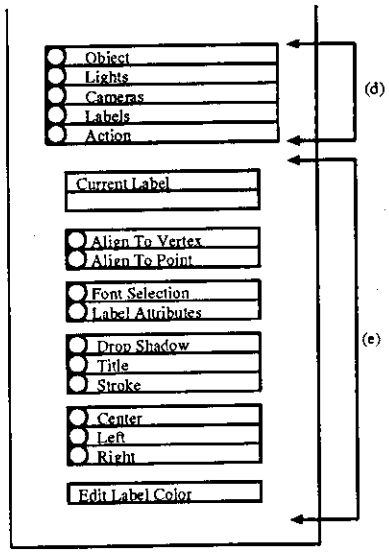

図 3.8 Labels メニュー

1) Current Labels

カレントオブジェクトにラベルを以下のようにして付けます。

1) ラベルを付けるオブジェクトを選択します。
2) Current Labels の下にカーソルを移動してラベル名を入力します。
3) リターンキーを押すとカレントオブジェクトの中心にラベルを張り付けます。
4) オブジェクトに複数のラベルを付ける時には、まず1つめのラベルをつけた後にマウスでカレントオブジェクトクリックすると Current Labels の下に入力した1つめのラベル名をクリアします。2)、3) の操作で2つめのラベルを付けます。

ラベルの移動について

ラベルは、以下のようにして移動することができます。

1) ビューウィンドウのラベルの上にカーソルを移動しクリックしたままにします。
2) カーソルを移動しボタンを離すとその場所にラベルが移動されます。

2) Align to Vertex/Align to Point

オブジェクトの頂点にラベルを付けます。

1) ラベルを付けるオブジェクトを選択します。
2) Align to Vertex ボタンをクリックします。
3) ビューウィンドウのラベルの上にカーソルを移動しマウスの左ボタンをクリックしたままにします。
4) カーソルをオブジェクトの1つの頂点の上に移動しマウスを離すとその場所にラベルが移動します。

3) Label Attribute

- Title

  Title ボタン（図3.8）をクリック（オンにする）してからラベルを Current Labels（図3.8）の下にカーソルを移動してラベル名を入力しリターンキーを入力するとビューウィンドウの左上に常にラベルを表示します。

- Center/Left/Right

  ラベルを右詰めにするのか左詰めにするのか真中にするのかを選択します。

- Edit Label Color

  Edit Label Color ボタン（図3.8）をクリックするとラベルの色を設定する編集パネルを表示します。編集パネルを使ってラベルの色を決定します。

4) Font Selection

Font Selection ボタン（図3.8）をクリックすると図3.9のようなメニューを表示します。メニュー（図3.9）の中から使用したいフォントをクリックすると選択したフォントでラベルが書き換えられます。

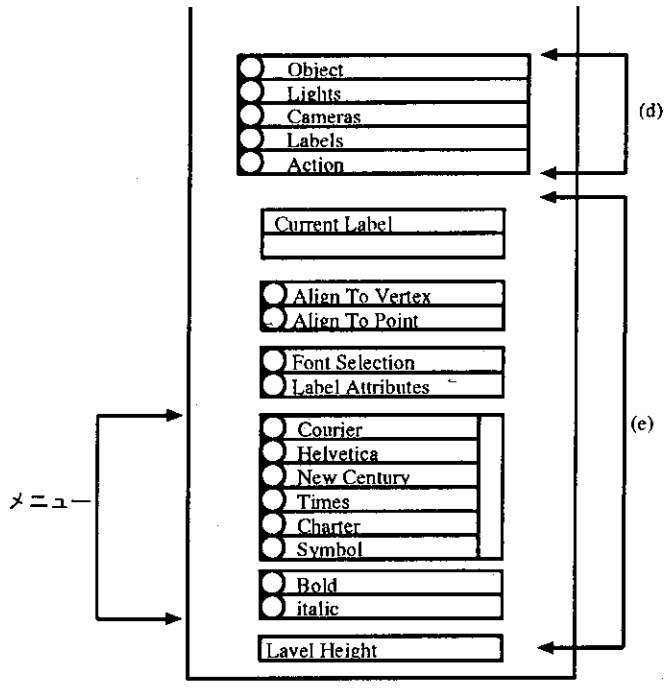

図 3.9 Font Selection

4．モジュール作成

C言語またはＦＯＲＴＲＡＮ言語を使って、独自の処理を行なうモジュールを作成できます。このモジュール作成を容易にするため、Module Generater モジュールを使用します。Module Generater モジュールは、プログラムの大まかな部分（骨格部分）を作成します。そのため、これを利用することによって、一からプログラミングすることなしに容易にモジュールを作成できます。

モジュールの作成手順の概要

モジュールの作成手順の概要を以下に示し、その具体的な方法を 4.1以降に示します。*

1) Module Generator モジュールを使用してプログラムの骨格部分を作成します。
2) 1) のプログラムにモジュールのメインとなる機能（処理ルーチン）を書き入れるとプログラムが完成します。
3) プログラムをコンパイルすると、モジュールが完成します。
4) モジュールをモジュールパレットに追加します。

プログラムの骨格部分で設定する項目

なお、プログラムの骨格部分を作成するに当たって、以下の項目の設定をしなければなりません。

1) モジュール名
2) カテゴリ（Input, Filter, Mapper, Output）
3) プログラミング言語（C言語、ＦＯＲＴＲＡＮ言語）
4) モジュールタイプ（サブルーチン、コルーチン）
5) 入力ポート
6) 出力ポート
7) パラメータ
8) ファイル名

---

*/usr/avs/examples にモジュールの作成例がありますので参考にして下さい。

サブルーチンモジュール、コルーチンモジュール

モジュールには、サブルーチン (Subroutine) モジュールとコルーチン (Coroutine) モジュールの２つのタイプがあります。

- サブルーチン (Subroutine) モジュール
  サブルーチンモジュールは、次に示すように受動的なモジュールです。サブルーチンモジュールをワークスペースに移動すると自分自身を初期化（パラメータを初期値に設定、データを読み込むためのメモリを確保、表示用のウィンドウを生成）します。その後は、実行要求がある（入力データまたは、パラメータが変更される）まで待ちます。
  サブルーチンは、以下の関数によって構成されます。
  1. initialization function
  2. description function
  3. computation function

- コルーチン (Coroutine) モジュール
  コルーチンモジュールは、次に示すように能動的なモジュールです。コルーチンモジュールは、サブルーチンモジュールと同じようにワークスペースに移動すると初期化します。その後は、実行要求の有無に関係なく常に実行します。
  コルーチンモジュールは以下の関数によって構成されます。
  1. description function
  2. main

4.1 骨格部分の作成

4.1.1 Module Generator モジュールの操作手順

Module Generator モジュールを使用してプログラムの骨格部分を作る方法を以下に説明します。作成するモジュールは、実数のフィールドデータを整数のフィールドデータに変換するものです。作成するモジュール (module_test) の骨格部分を作成するための以下のように項目を設定します。

モジュール名	module_test
モジュールのカテゴリ	Filter
プログラミング言語	FORTRAN
入力ポート名	input_a
入力ポートデータタイプ	field 3D 3-space 1-vector float uniform
出力ポート名	output_a
出力ポートデータタイプ	field 3D 3-space 1-vector integer uniform
パラメータ名	file name
パラメータタイプ	string
ウィジェットタイプ	browser
保存ファイル名	module_test_f.f

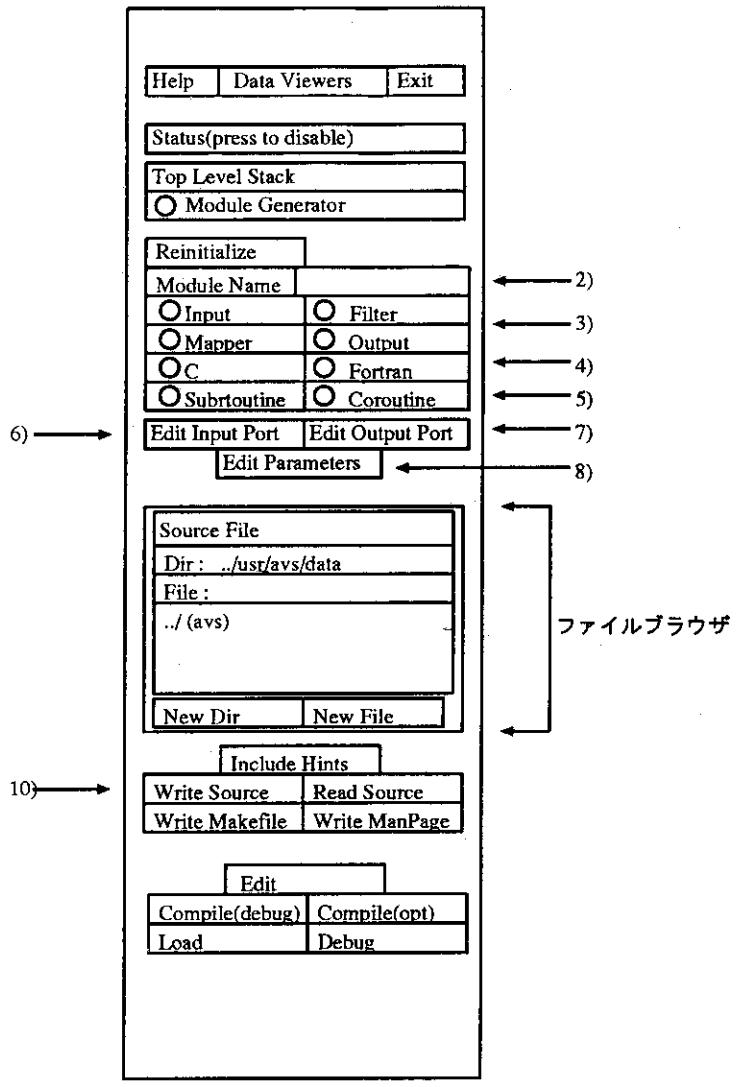

図 4.1 Module Generator モジュールのコントロールウィジェット

1) モジュールパレットより Module Generator モジュールをワークスペースに移動します。この移動により、ネットワークコントロールパネルに Module Generator モジュールのコントロールウィジェットとして図 4.1を表示します。

2) モジュール名を設定するため、Module Name の右側のスペースにモジュール名を入力します。ここでは、module_test と入力します。

3) モジュールのカテゴリ（Input Data, Filter, Mapper, Data Output（renderer））を選択し、マウスでクリックして下さい。ここでは、Filter ボタンをクリックします。

4) プログラミング言語（C, FORTRAN）を選択します。ここでは FORTRAN ボタンをクリックします。

5) モジュールタイプ（Subroutine, Coroutine）を選択します。ここでは Subroutine ボタンをクリックします。

6) 入力ポートを設定するため、Edit Input Port ボタンをクリックすると、Input Port Editor ウィンドウ（図4.2）を表示します。

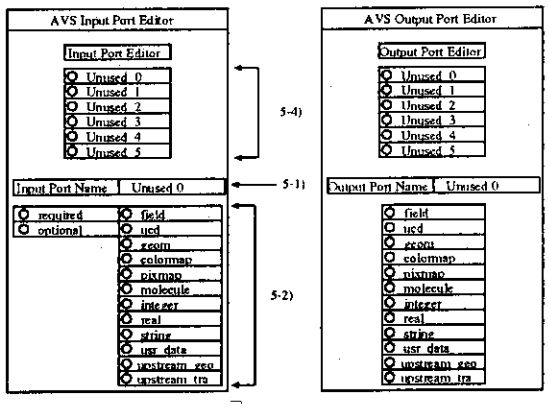

図 4.2 AVS Input(Output) Port Editor

6-1) Input Port Name の右側（最初は、Unused 0 になっている）に、入力ポートの名前（任意）を入力します。ここでは、input_a と入力します。

6-2) 入力ポートのデータタイプを選択します。ここでは、field ボタンをクリックします。field ボタンをクリックした場合は AVS Field Editor（図4.3）を表示します。

ここでは入力ポートのフィールドデータのデータタイプを以下のようにクリックします。

配列（計算空間）次元数	Dimensions	(3次元)
フィールド（物理空間）の次元	Physical Space	(3次元)
データタイプ	Data Type	(float)
マッピングのタイプ	Spacing	(uniform)
ノードのデータ数（要素数）	Vector Length	(1)

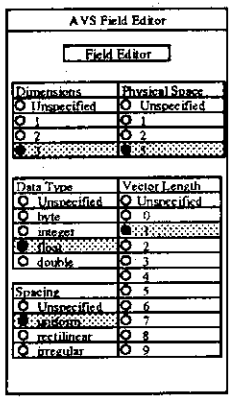

図 4.3 AVS Field Editor

6-3) Edit Input Port ボタン（図4.1）を再度クリックするとウィンドウ（図4.2、図4.3）を閉じます。

6-4) ２つ以上の入力ポートをもつモジュールを作成する場合、図4.2の Input Port Editor の下にある Unused 1 ～ Unused 5 をクリックしてから 6-1) ～6-3) の操作を行ないます。入力ポートは最大６つまで設定でき、２つ目の入力ポートの設定は Unused 1 をクリック、３つ目の力ポートの設定は Unused 2、……６つ目の入力ポートの設定は Unused 5 をクリックしてから行ないます。

7) 出力ポートの設定を行ないます。出力ポートは入力ポートと同じように最大６つまで設定できます。図4.1の 7 の Edit Output Port ボタンをクリックすると、AVS Output Port Editor（図4.2）を表示します。次に Output Port Name（図4.2）の右側（最初は Unused 0 になっている）に出力ポートの名前を入力します。ここでは output_a と入力し、データタイプは入力ポートのときと同じように設定します。

8) パラメータの設定を以下のようにして行ないます。パラメータは最大９９個まで設定できます。

8-1) Edit Parameters ボタン（図4.1）をクリックすると AVS Parameter Editor（図4.4）を表示します。Parameter Name の右側（最初は Unused 0 になっている）にパラメータの名前を入力します。ここでは、file name と入力します。

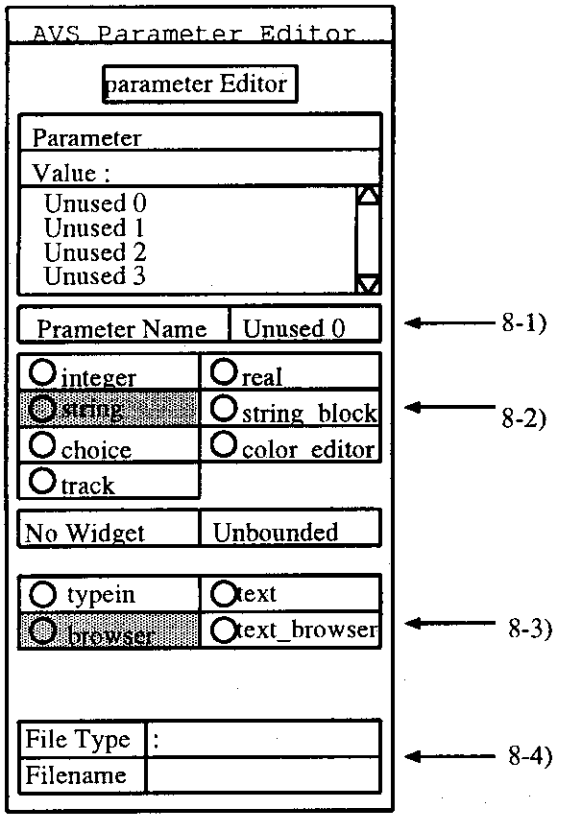

図 4.4 AVS Parameter Editor

8-2) パラメータのタイプを選択します。ここでは、パラメータタイプは文字列なので string をクリックします。

8-3) パラメータを設定するコントロールウィジェットのウィジェットタイプ（ダイヤルウィジェット (idial ボタン)、タイプインウィジェット (typein ボタン)、ファイルブラウザ (browser ボタン) など）を選んでそのボタンをクリックします。パラメータのタイプによって選択できるウィジェットタイプは異なります。ここでは browser ボタンをクリックします。

8-4) パラメータの初期値や最大値、最小値などを設定します。パラメータの種類によって設定する内容は異なります。ここでは特に設定していません。

9) Module Generator モジュールのコントロールウィジェットの中にあるファイルブラウザ（図 4.1）を使ってプログラムを保存するファイル名を選択、または、入力します。ここでは module_test_f.f に保存します。そのため、New File ボタンをクリックして表示したウィンドウに module_test_f.f と入力します。

10) Write Source ボタン（図 4.1）をクリックすると 9) で入力したファイルにモジュールの骨格部分のプログラムを保存します。

11) Write Makefile ボタン（図 4.1）をクリックするとソースファイルをコンパイルするための Makefile が自動的に作成されファイル名 Makefile. ソースファイル名として保存されます。

12 ファイルブラウザ（図 4.1）の下にある Include Hints ボタンをクリックしたあとで 10) の操作によって保存を実行すると骨格部分のプログラムにメモリーの解放や割り当てなどのヒントを加えたプログラムを 9) で入力したファイルに保存します。

4.2 処理ルーチンの組み込み

Module Generator モジュールを使用してプログラムの骨格部分を作成したら、プログラムにモジュールのメインとなる機能（処理ルーチン）を加えます。プログラムの編集は vi エディタや nemacs エディタを使用します。

vi エディタで組み込む方法を以下に示します。

1) ネットワークエディタを起動して Module Generator モジュールをワークスペースに移動します。

2) ファイルブラウザ（図 4.1）を使ってプログラムを保存したファイルをクリックします。

3) Read Source ボタンをクリックするとファイルを読み込みます。

4) edit ボタン（図 4.1）をクリックすると vi エディタを起動してファイルを読み込みます。

nemacs エディタで組み込む場合は、以下のコマンドを入力します。ファイル名にはプログラムを保存したファイル名を入力します。

```
% nemacs ファイル名
```

以下にここまで Module Generator モジュールで作成したプログラムに処理ルーチンを組み込む例を示します。

4.2.1 サブルーチンモジュールの作成

(1) サブルーチンモジュールの構成

サブルーチンモジュールは、以下の function （関数）で構成します。初期化関数と記述関数は、 Module Generator モジュールによってＡＶＳが作成します。計算関数は、作成するモジュールのメインになる機能を記述した関数ですのでユーザが作成しなければなりません。

- initialization function （初期化関数）
  モジュールがワークスペースに移動したときにモジュールを初期化する関数です。モジュールの初期化としてメモリの領域確保やウィンドウの生成などを行ないます。

- description function （記述関数）
  モジュールの以下のようなインターフェース部分を記述する関数です。

  – モジュールの名前

  – モジュールタイプ

  – 入出力ポート

  – パラメータ、パラメータのウィジェットタイプ

- computation function （計算関数）
  モジュールのメインとなる機能（計算ルーチン）を記述します。

以下に、各関数の内容を示します。

1) initialization function （初期化関数）
   初期化関数は、モジュールをワークスペースに移動したときに、2) に示す記述関数の内容に従ってモジュールを初期化します。この初期化関数は Module Generator モジュールが作成します。

   a) 初期化関数のプログラム

```
C ********************************************************************
C Initialization for modules contained in this file.
C ********************************************************************
subroutine AVSinit_modules
include 'avs/avs.inc'

external module_test_desc
integer module_test_desc
call AVSmodule_from_desc(module_test_desc)
end
```

   b) 初期化関数で使用する関数

   * subroutine AVSinit_modules
     モジュールをワークスペースに移動したときに記述関数を参照してその内容に従って初期化（メモリの確保、ウィンドウの生成）を行ないます。

* AVSmodule_from_desc(description_routine)
  初期化のために参照する記述関数を呼び出します。
  description_routine : 記述関数名 (integer)　カッコ内は記述関数が整数値を返すことを示しています。

2) description function (記述関数)
記述関数には、モジュールの名前、入出力ポート、パラメータ及びウィジェットタイプなどのインタフェース部分を記述する関数です。この記述関数は Module Generator モジュールが作成します。

a) 記述関数のプログラム

```
      C ***************************************
      C  Module Description
      C ***************************************
      integer function module_test_desc()
      implicit none
      include 'avs/avs.inc'

      integer in_port, out_port, param, iresult
      external module_test_compute
      integer module_test_compute
      call AVSset_module_name('module_test', 'filter')
      call AVSset_module_flags(single_arg_data)

      C       Input Port Specifications
      in_port = AVScreate_input_port('input_a',
     $  'field 3D 3-space 1-vector uniform float', REQUIRED)

      C       Output Port Specifications
      out_port = AVScreate_output_port('output_a',
     $  'field 3D 3-space 1-vector uniform integer')

      C       Parameter Specifications
      param = AVSadd_parameter('file name', 'string', ' ', ' ',
     $ '.dat')
      call AVSconnect_widget(param, 'browser')

      call AVSset_compute_proc(module_test_compute)
      C ----> START OF USER-SUPPLIED CODE SECTION #2 (ADDITIONAL
     $ SPECIFICATION INFO)
      C <---- END OF USER-SUPPLIED CODE SECTION #2
      module_test_desc = 1
      return
      end
```

b) 記述関数で使用する関数

以下に記述関数で使用する関数を示します。

* call AVSset_module_name(name, type)

モジュールのモジュール名とモジュールのカテゴリを記述します。

name： モジュール名 (character)

type： モジュールのカテゴリ (Data Input,Filter,Mapper,Data Output(renderer))

* inport_name = AVScreate_input_port(name, type, flags)

モジュールの入力ポート名とその入力ポートが受け入れ可能なデータタイプを記述します。

name： 入力ポート名 (character)

type： 入力ポートのデータタイプ (character)

flags： 入力ポートの接続が必修の場合は、 required

入力ポートの接続が必ずしも必要でない場合は、 optional

type には、データタイプを以下のように記述します。

byte	'byte'
integer	'integer'
real	'real'
string	'string'
field	'field'
colormap	'colormap'
geometry	'geom'
ucd	'ucd'

またフィールドデータについては、更に以下のような記述が必要です。

' field lD m-space n-vector mapping_type data_type '

lD	次元数 (l には次元数を記述)
m-space	物理空間の次元数
n-vector	各ノードのデータ数
mapping_type	フィールドのタイプ
data_type	データのデータタイプ

* outport_name = AVScreate_output_port(name, type)

モジュールの出力ポート名と出力ポートから出力するデータのデータタイプを記述します。

name： 出力ポート名 (character)

type： 出力ポートのデータタイプ (character)

* param = AVSadd_parameter(name, type, init, minival, maxval)

パラメータ名、パラメータのデータタイプ、初期値、最小値、最大値を記述します。

name ……パラメータの名前 (character)

type ……パラメータのタイプ (character)

init, minival, maxval ……初期値、最小値、最大値 (type)

* call AVSconnect_widget(param, widget_type)
  コントロールウィジェットのタイプを記述します。
  param ……コントロールウィジェットで設定するパラメータの名前 (character)
  widget_type ……ウィジェットタイプ (character)
* call AVSset_compute_proc(compute_routine)
  計算関数を呼び出します。
  compute_routine ……計算関数名 (integer)　カッコ内は計算関数が整数値を返すことを示しています。

入力ポート、出力ポート、パラメータポートの型宣言は以下のように宣言します。

AVSのデータタイプ	入力 / パラメータのタイプ	出力データタイプ
byte	byte	byte
integer	integer	integer
real	real	real
string	character*(*)	character*(*)
field	integer	integer
colormap	integer	integer
geometry	integer	integer
ucd	integer	integer
ユーザ定義データ	integer	integer

3) computation function (計算関数)

計算関数は、メインとなる処理を行ないます。計算関数のプログラムの以下に示す部分はModule Generatorモジュールが作成します。

a) 計算関数のプログラム

```
      C ****************************************
      C Module Compute Routine
      C ****************************************
      integer function module_test_compute(input_a,output_a,
     $ file_name)
      implicit none
      include 'avs/avs.inc'
      integer input_a
      integer output_a
      character*(*) file_name
       C  THIS IS A 'HINTS' AREA - YOU MAY CUT AND PASTE AT WILL FROM IT
       integer dims0
       dimension dims0(3)
       C Free old field data
      if (output_a .ne. 0) call AVSfield_free(output_a)
       C Allocate space for new field output
      dims0(0) = FILL_IN_DIMENSIONS_HERE!
      dims0(1) = FILL_IN_DIMENSIONS_HERE!
      dims0(2) = FILL_IN_DIMENSIONS_HERE!

       C YOU MUST FILL IN THE FOLLOWING "field" STRING
       C WITH A MORE SPECIFIC DESCRIPTION!
      output_a = AVSdata_alloc('field 3D 3-space 2-vector integer',
     $ dims0)
      if (output_a .eq. 0) then
      call AVSerror('Allocation of output field failed.')
          module_test_compute = 0
          return
       endif
       C  FILL IN THE OUTPUT FIELD  Here!
       C THIS IS THE END OF THE 'HINTS' AREA. ADD YOUR OWN CODE BETWEEN THE
       C FOLLOWING COMMENTS. THE CODE YOU SUPPLY WILL NOT BE OVERWRITTEN.
       C ---- START OF USER-SUPPLIED CODE SECTION \#3 (COMPUTE ROUTINE BODY)
       C ---- END OF USER-SUPPLIED CODE SECTION \#3
      module\_test\_compute = 1
       return
      end
```

さらにメインとなる処理は以下で説明する計算関数を作成する上で必要な基礎知識を頭に入れて作成します。

b) 計算関数で使用する関数

- データの流れ

ＡＶＳでは、以下のようにしてデータの受渡しを行ないます。ここでは、データを送る側のモジュールを上流モジュール、データを受ける側のモジュールを下流モジュールとします。

上流モジュールは、出力したデータをメモリ上に保存します。保存するために必要な出力領域をメモリ上に確保し、その領域に出力したいデータを書き込む処理を行ないます。

一方、下流モジュールは、上流モジュールによってデータが書き込まれたメモリ上の出力領域にアクセスしてデータを読み込みます。

データのアクセスと書き込みにはＡＶＳ関数を用います。

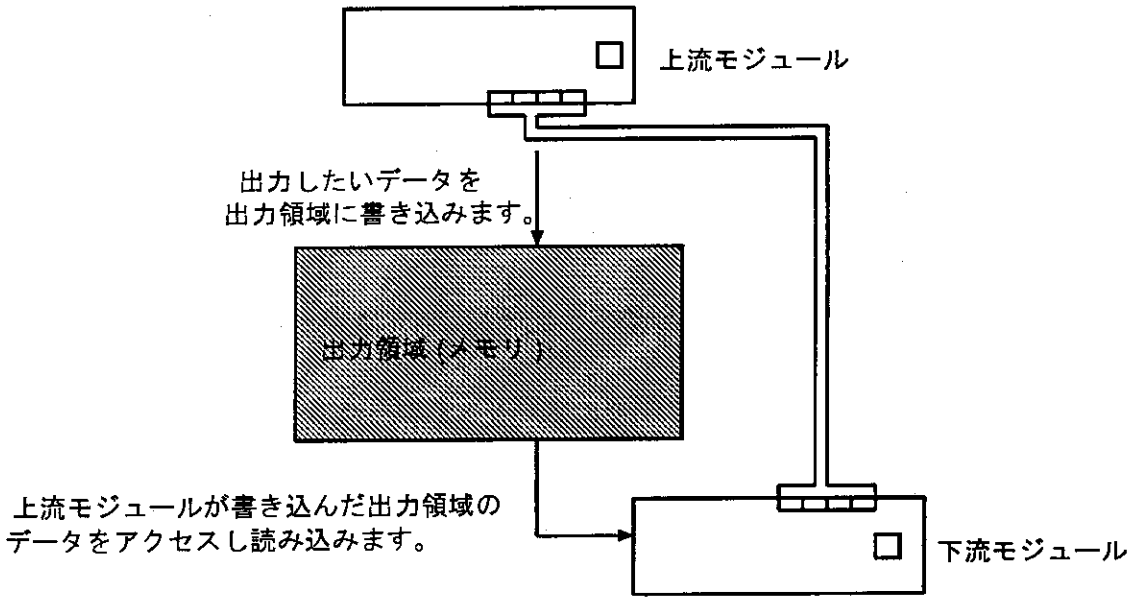

図 4.5　データの流れ

1) 入力データのアクセス

フィールドデータのアクセスに使う主なＡＶＳ関数を以下に示します。

a. AVSfield_get_int

フィールドデータの中の整数値データ (integer) と呼ばれるデータをアクセスするために使用します。

```
integer AVSfield_get_int(field,selector)
  integer field
  integer selector
```

どの整数値データにアクセスするかは、selector にアクセスしたい情報の引数を記述します。整数値データには、以下のものがあります。

AVS_field_ndim	データの次元数
AVS_field_nspace	物理空間の次元数
AVS_field_veclen	ベクトル長
AVS_field_type	データタイプ
AVS_field_size	データのサイズ
AVS_field_uniform	フィールドのタイプ

b. AVSfield_get_dimension

フィールドデータの次元毎のサイズをアクセスするためのＡＶＳ関数です。アクセスした次元毎のサイズを配列 dims に格納します。

```
integer AVSfield_get_dimension(field,dims)
  integer field
  integer dims(ndim)
```

dims に次元毎のサイズを格納する配列の名前を指定します。

c. AVSfield_data_offset

フィールドデータのデータ領域にアクセスするためのＡＶＳ関数です。

AVSfield_data_offset(field, basevec, offset)

integer	field	アクセスするフィールド
(type)	basevec(1)	データ領域を格納するための配列
integer	offset	配列のオフセット値がセットされる

d. AVSfield_points_offset

フィールドデータの座標領域にアクセスするためのＡＶＳ関数です。

AVSfield_data_offset(field, basevec, offset)

integer	field	アクセスするフィールド
(type)	basevec(1)	座標領域を格納するための配列
integer	offset	配列のオフセット値がセットされる

2) 出力データの書き込み

フィールドデータをメモリ上に書き込むためのＡＶＳ関数を示します。

a. AVSdata_alloc

出力データを書き込むために必要な出力領域を確保するためのＡＶＳ関数です。

AVSdata_alloc('field ', dims)

b. AVSfield_set_int

フィールドデータ内の整数データ（integer）を出力領域に書き込むためのＡＶＳ関数です。整数データには、以下の項目があります。

integer AVSfield_set_int(field,selector)

selector にセットしたい情報の引数を指定します。

AVS_field_ndim	データの次元数
AVS_field_nspace	物理空間の次元数
AVS_field_veclen	ベクトル長
AVS_field_type	データタイプ
AVS_field_size	データのサイズ
AVS_field_uniform	フィールドのタイプ

c. AVSfield_set_dimension

フィールドデータの次元毎のサイズを出力領域に書き込むためのＡＶＳ関数です。以下のように記述します。

integer AVSfield_set_dimension(field,dimensions)

integer field

integer dimensions(ndim)

dimensions に配列の次元（dim1,dim2,・・・）を指定します。

d. AVSfield_data_offset

出力領域にデータを書き込むためのＡＶＳ関数です。以下のように記述します。

AVSfield_data_offset(field, basevec, offset)

integer	field	書き込むフィールド
(type)	basevec(1)	データ領域をセットするための配列
integer	offset	配列のオフセット値がセットされる

e. AVSfield_points_offset

出力領域に座標情報を書き込むためのＡＶＳ関数です。以下のように記述します。

AVSfield_data_offset(field, basevec, offset)

integer	field	書き込むフィールド
real	basevec(1)	座標領域をセットするための配列
integer	offset	配列のオフセット値がセットされる

3) サブルーチンモジュールの完成

作成した module_test_f.f （4.1.1 (1) の 1) の a)、2) の a)、3) の a)）の中に、入力ポートから取り込んだフィールドデータのデータタイプを実数から整数へ変換する処理を組み込みます。

入力データのアクセスと出力データの出力領域へのセット部分の作成

1) モジュールはデータを書き込むときに、書き込むデータの大きさに従ってメモリ上に出力領域を確保します。このときデータの大きさを求めるために配列の大きさが必要となります。

作成するモジュールは、データの配列の大きさは入力時と出力時で変わらないので入力データの配列の大きさを出力データの配列の大きさとします。入力データの配列の大きさは以下のＡＶＳ関数を使ってアクセスします。

```
AVSfield_get_dimensions(input_a,dims)
    integer   dims   (dims(1),dims(2),dims(3) に各次元の大きさが入ります)
    integer   input_a
```

2) 取り込むデータにアクセスします。

データ領域のアクセスには、 AVSfield_data_offset を使います。

```
AVSfield_data_offset(input_a, ifield, offset1)
 integer       input_a
 dimension     ifield(1)
 integer       offset1
```

3) モジュールが出力するデータ（実数値から整数値に変換したデータ）を出力領域に書き込みます。出力領域へ書き込むには、 AVSfield_data_offset を使います。

```
AVSfield_data_offset(output_a, ofield, offset2)
 integer output_a
 dimension ofield(1)
 integer offset2
```

モジュール内の処理（実数値から整数値への変換）部分の作成

実数値から整数値への変換の処理を行なう部分を計算関数のサブルーチンとして作成します。関数名は module_test_compute2 とします。

```
integer function module_test_compute2(in,out,dims,filename)
real in(*)
integer out(*)
integer dims(*)
character*(*) filename
open(1,file = filename)
do i = 1, dims(1) * dims(2) * dims(3)
   out(i) = int( in(i) )
   write(1,100) out(i)
enddo
  100   format(i3)
close(1)
module_test_compute2 = 1
return
end
 C <---- END OF USER-SUPPLIED CODE SECTION #4
```

上で作成したサブルーチンを含めた完成した計算関数を以下に示します。

```
 C ****************************************
 C Module Compute Routine
 C ****************************************
integer function module_test_compute( input_a, output_a,
      $  file_name)
implicit none
include 'avs/avs.inc'
integer input_a
integer output_a
character*(*) file_name

 C  THIS IS A 'HINTS' AREA - YOU MAY CUT AND PASTE AT WILL FROM IT
 integer dims0
 dimension dims0(3)
 C
 integer module_test_compute2
 external module_test_compute2
 integer offset1, offset2
 real ifield
 integer ofield
```

```
 dimension ifield(1), ofield(1)
 integer iret
iret = AVSfield_get_dimensions(input_a, dims0)
 C
 C
 C Free old field data
if (output_a .ne. 0) call AVSfield_free(output_a)
 C Allocate space for new field output
 C       dims0(0) = <<FILL_IN_DIMENSIONS_HERE!>>
 C       dims0(1) = <<FILL_IN_DIMENSIONS_HERE!>>
 C       dims0(2) = <<FILL_IN_DIMENSIONS_HERE!>>
output_a = AVSdata_alloc('
      $ field 3D 3-space 1-vector uniform integer',
      $  dims0)
if (output_a .eq. 0) then
call AVSerror('Allocation of output field failed.')
    module_test_compute = 0
    return
 endif
 C  FILL IN THE OUTPUT FIELD  Here!
 C THIS IS THE END OF THE 'HINTS' AREA. ADD YOUR OWN CODE BETWEEN THE
 C FOLLOWING COMMENTS. THE CODE YOU SUPPLY WILL NOT BE OVERWRITTEN.

 C ----> START OF USER-SUPPLIED CODE SECTION #3 (COMPUTE ROUTINE BODY)
 iret = AVSfield_data_offset(input_a, ifield, offset1)
 iret = AVSfield_data_offset(output_a, ofield, offset2)
 iret = module_test_compute2(
      $  ifield(offset1+1),ofield(offset2+1),dims0,file_name)
 C <---- END OF USER-SUPPLIED CODE SECTION #3
 C <---- END OF USER-SUPPLIED CODE SECTION #3
module_test_compute = 1
return
end

 C ----> START OF USER-SUPPLIED CODE SECTION #4 (SUBROUTINES, FUNCTIONS,
 UTILITY ROUTINES)
integer function module_test_compute2(in,out,dims,filename)
real in(*)
integer out(*)
integer dims(*)
character*(*) filename
open(1,file = filename)
do i = 1, dims(1) * dims(2) * dims(3)
   out(i) = int( in(i) )
   write(1,100) out(i)
```

```
enddo
  100   format(i3)
close(1)
module_test_compute2 = 1
return
end
 C <---- END OF USER-SUPPLIED CODE SECTION #4
```

4.2.2 モジュールの登録

完成したプログラムを、モジュールとしてモジュールパレットに登録するまでの方法を説明します。

1) 図4.1の Module Generator モジュールの Read Source ボタンをクリックして完成したプログラム（以降プログラムという）を読み込みます。

2) 図4.1の Module Generator モジュールのコントロールウィジェットの Compile(debug) ボタンまたは Compile(opt) ボタンをクリックするとプログラムのコンパイルが開始され、コンパイルの状況をウィンドウ（図4.6）に表示します。

3) コンパイルが正常終了すると、図4.6のウィンドウの下2行のメッセージを表示して、実行ファイルを作成します。エラーがあるとウィンドウにエラーメッセージを表示します。

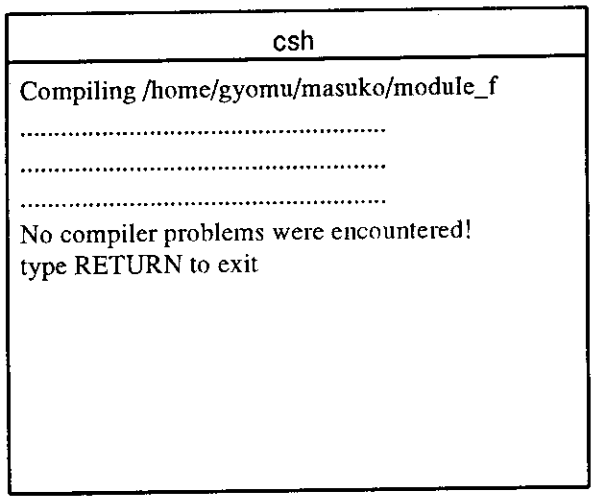

図 4.6 コンパイル時に表示されるウィンドウ

4) ウィンドウ（図4.6）は、リターンキーで終了します。エラーがある場合は、エディタで再び訂正してからコンパイルを行ないます。

5) 図4.1の Module Generator モジュールのコントロールウィジェットの Load ボタンをクリックするとモジュールパレットに実行ファイルが登録されます。後は、ＡＶＳのライセンスにより提供されているモジュールと同様に扱うことができます。

5．動画の録画

5.1　動画の録画に必要な環境

ＡＶＳで作成した静止画を用い、動画として録画するために必要なシステムを説明します。情報システム管理課にある動画像処理システムの構成（図5.1）を以下に示します。

- ＥＷＳ
  ＡＶＳを実行させるためのワークステーションです。
  （情報システム管理課では、HP9000/730CRX-24z を画像サーバとして使用しており、末尾の CRX-24z は高速な描画を可能とするグラフィックスアクセラレータを示します）

- フレームスキャンコンバータ
  ワークステーションからの映像用 RGB 信号を標準の TV 信号に変換します。また、録画したい部分の選択や、画面の拡大、縮小などの調整を行ないます。（情報システム管理課では、（株）フォトロン FSC-64000vz を用いています）

- V-LAN
  ビデオをコマ撮りするための制御装置です。（情報システム管理課では、（株）フォトロン RS-422A を用いています）

- ＶＴＲ
  ビデオテープに動画を録画するために用います。（情報システム管理課では、Victor BR-S822B （業務用の S-VHS ビデオ）を用いています）

- モニター
  フレームスキャンコンバータを介して実際にビデオに録画する画面を表示します。（情報システム管理課では、Victor VM-R150S を用いています）

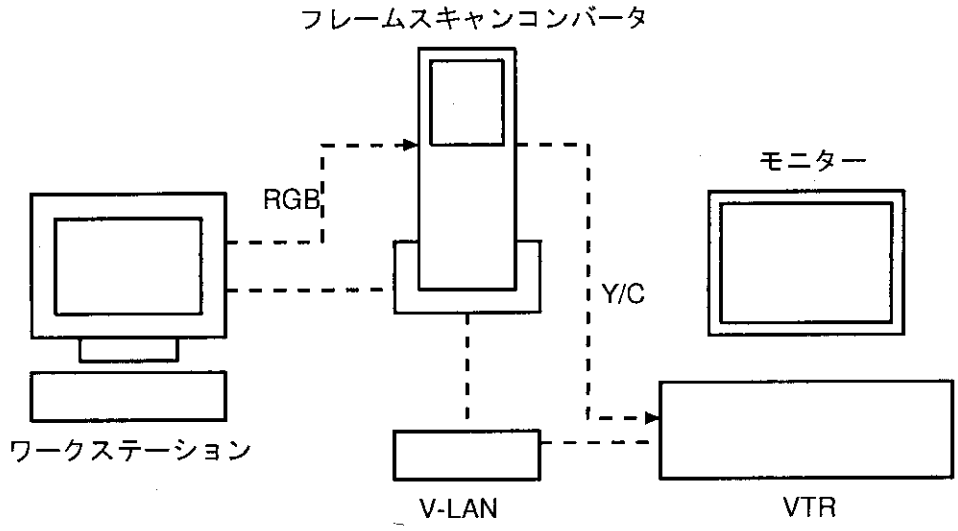

図 5.1　システム構成図

5.2以降でスクリプトファイルによる動画の作成を、5.4以降でＡＶＳビデオキット（旭化成情報システム（株）製）による動画の作成を説明します。

5.2　スクリプトファイルによる動画の作成

5.2.1　スクリプトファイルの作成

ＡＶＳの制御は、ＣＬＩ言語を使います。ＣＬＩ言語で記述したファイルをスクリプトファイルといいます。スクリプトファイルを実行することによって自動的にＡＶＳを起動したり、ネットワークを作成したり、可視化したイメージを描画したりすることができます。

スクリプトファイルは、ユーザがＣＬＩ言語の文法に従って記述して作成する方法、ＡＶＳが自動的に一連の操作をＣＬＩ言語で記述して作成する方法の２通りの作成方法があります。

ここでは、後者の方法でスクリプトファイルを作成します。スクリプトファイルのファイル名は、＊＊＊.scr です。ここではファイル名を video_cli.scr とします。

1) 以下のコマンドによりＡＶＳを起動します。この起動によりメインメニューが表示されます。

```
% avs -cli
```

2) 起動後、起動を行なったウィンドウ内でリターンキーを入力すると以下のようなプロンプトを表示します。

```
avs>
```

3) この状態で以下のコマンドを入力します。

```
avs>script -open video_cli.scr
```

ＡＶＳは、この3)の操作をしてからＡＶＳの終了までに行なう一連の操作（ネットワークを読み込む、パラメータの値を変更する、イメージを表示させる等）を自動的にＣＬＩ言語でファイルに書き込みます。

5.2.2 スクリプトファイルの作成例

引続き以下の操作を行ないます。

1) メインメニューのNetwork Editorボタンをクリックしネットワークエディターを起動します。

2) Network Toolsメニューの Read Networkボタン（図1.5）をクリックしネットワークファイル vtr_network.net を読み込ませます。ネットワーク vtr_network.net は、/home/guest/avsuser/vtr/data_vtr を読み込んで可視化を行ないます。

3) /home/guest/avsuser/vtr/data_vtr/vtr_data020.fld を読み込ませて可視化を実行させます。

4) /home/guest/avsuser/vtr/data_vtr/vtr_data030.fld を読み込ませて可視化を実行させます。

5) ネットワークコントロールパネルの Closeボタンをクリックしネットワークエディタを終了し、次に AVS ExitボタンをクリックしＡＶＳを終了します。

以上の手順のうち、ＡＶＳは、1)～5)の操作をＣＬＩ言語でスクリプトファイル video_cli.scr に以下のように記述します。

```
net_clear
net_read /home/gyomu/masuko/usr/vtr/vtr_network.net
parm_set "file browser.user.1":"File Browser" /home/guset/avsuser/vtr/
data_vtr/vtr\_data020.fld
parm_set "file browser.user.1":"File Browser" /home/guset/avsuser/vtr/
data_vtr/vtr\_data030.fld
AVSmessage Ok
```

5.2.3 スクリプトファイルの実行

作成したvideo_cli.scrは以下のコマンドで実行できます。ただし、スクリプトファイルを実行した場合は、ネットワークエディタのネットワークコンストラクションウィンドウ（図1.2）は表示されません。

```
% avs -cli "script -play video_cli.scr"
```

5.2.4 スクリプトファイルの編集

スクリプトファイルによる動画の作成では読み込むデータを順次変えていくことによって動画を作成します。

しかしスクリプトの作成をする場合にマウス操作を繰り返すことによってスクリプトファイルを作成するのでは効率的でないのでここでは、エディタのコピー&ペースト機能を使ってスクリプトファイルを編集します。以下に示すスクリプトファイルvideo_cli.scrの一部分をコピーします。

```
parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/
data_vtr/vtr_data020.fld
```

スクリプトファイルvideo_cli.scrを以下のように書きかえます。

```
net_clear
net_read /home/guest/avsuser/vtr/vtr_network.net
parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/
data_vtr/vtr_data020.fld
parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/
data_vtr/vtr_data030.fld
parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/
data_vtr/vtr_data040.fld ←コピー部
parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/
data_vtr/vtr_data050.fld ←コピー部
AVSmessage Ok
```

5.2.5 タイムコードの記録（動画の録画の準備）

ＶＴＲは、テープ上に記録したタイムコードで制御します。
ここでは、ビデオテープにタイムコードを記録する方法としてVictor BR-S822Bを使用した例を示します。

(1) ＶＴＲの準備

ＶＴＲには、フロントパネル（図5.2）とフロントサブパネル（図5.3）があります。フロントサブパネルは、フロントパネルの操作パネルの部分を上に開くと、内側にあります。

1) VTRの電源を投入し、VTRにビデオテープを入れます。
2) [INT/EXT]スイッチ（図5.3のm）をINTにします。
3) [PRESET/REGEN]スイッチ（図5.3のl）をPRESETにします。
4) [FREE/REC]スイッチ（図5.3のk）をFREEにします。
5) [DF/NDF]スイッチ（図5.3のj）をNDFにします。
6) [VITC REC]スイッチ（図5.3のi）をONにします。
7) [COUNTER]スイッチ（図5.2のa）をTCにします。
8) [HOLD]ボタン（図5.2のc）を押します。
9) [COUNTER RESET]ボタン（図5.2のe）を押し、カウンターの全ての桁を0にします。

(2) タイムコードの記録

ＶＴＲの準備ができたらタイムコードの記録を以下のようにして行ないます。

1) Playボタンと Recボタンを同時に押します。
2) [PRESET]ボタン（図5.2のd）を押すとカウンターの点滅が止まり、同時にタイムコードの記録が始まります。

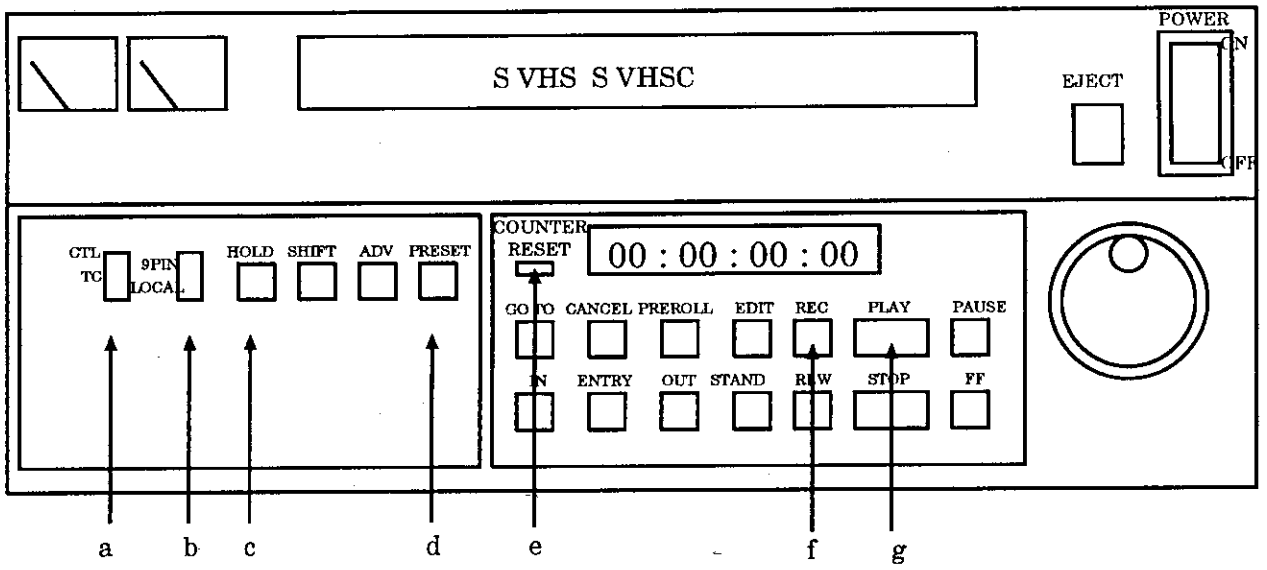

図 5.2 ＶＴＲのフロントパネル

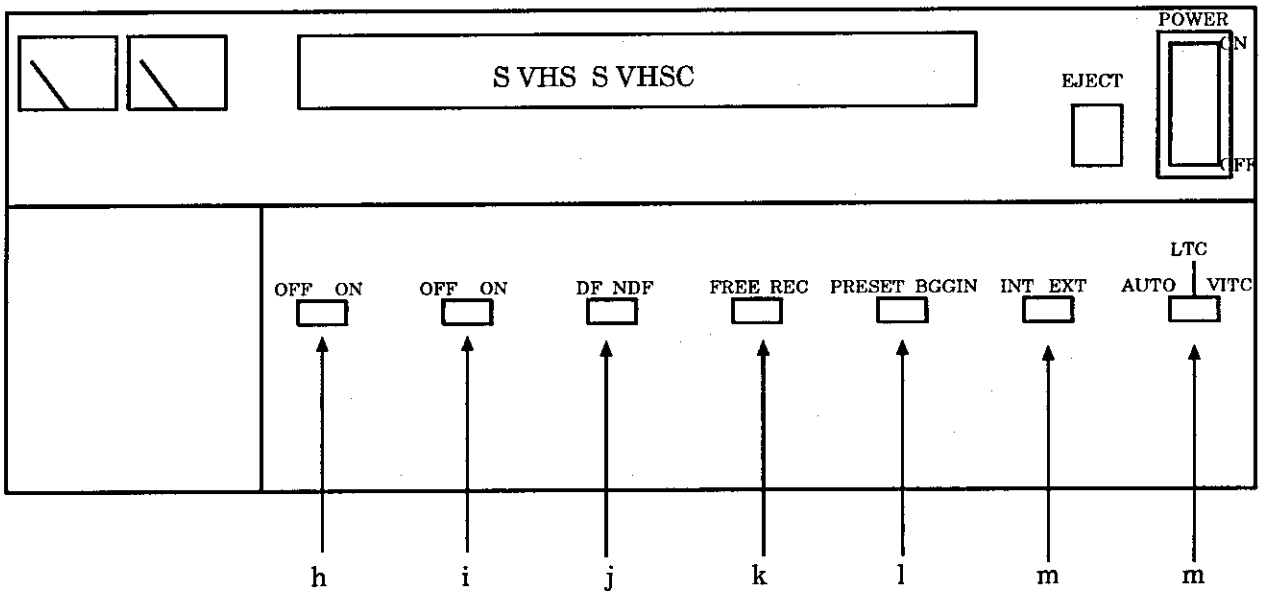

図 5.3 ＶＴＲのフロントサブパネル

5.2.6 動画の録画用のスクリプトファイルの作成（動画の録画の準備）

図 5.1に示した環境での動画録画の制御について説明します。

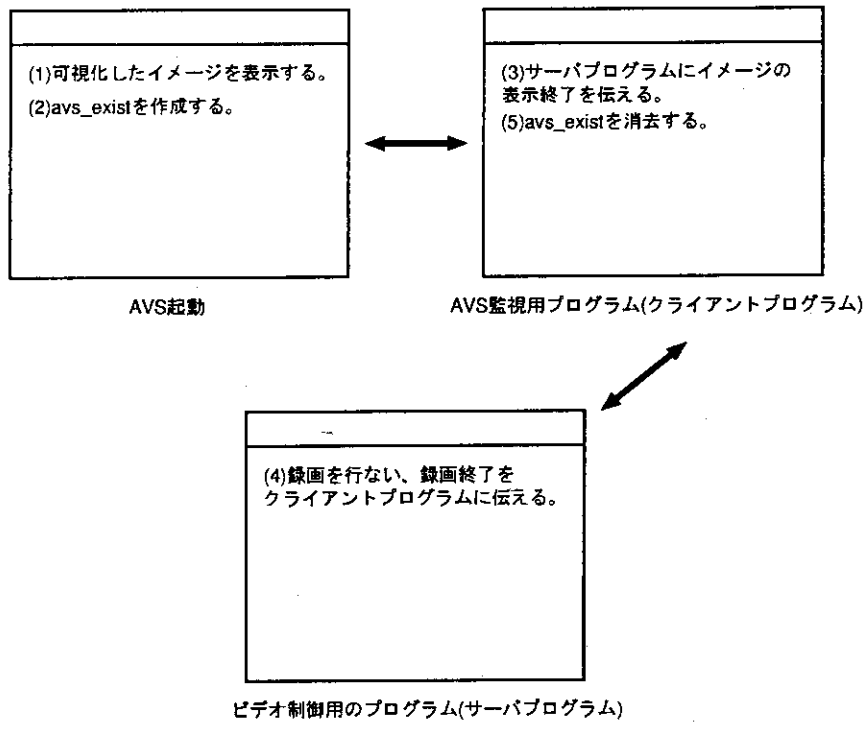

図 5.4 クライアントプログラム、サーバプログラム

1) スクリプトファイルに従って可視化したイメージをＡＶＳが表示します。

2) 可視化したイメージの表示が終了するとビデオの録画のトリガーとなるファイル avs_exist を作成し、ＡＶＳを約１０秒止めます。また最後の可視化したイメージの表示が終了するとファイル avs_finish を作成します。

3) クライアントプログラムは、ファイル avs_exist が存在した場合に終了をサーバプログラムに伝えます。

4) サーバプログラムは、クライアントプログラムから終了を受けるとビデオをリモート操作で録画し、録画終了をクライアントプログラムに伝えます。

5) クライアントプログラムは、録画終了を受けるとファイル avs_exist を削除します。

6) 1) の手順に戻り新たに avs_exist が存在するかどうかをチェックします。ファイル avs_exist が存在すれば 2) ～ 6) を引続き行ないます。しかし、チェックの際にファイル avs_finish が存在したときには、サーバとクライアントのプログラムが終了します。

これらの制御のためにスクリプトファイルに以下のような準備が必要となります。

動画の録画の制御の際に、クライアントプログラムはファイル avs_exist またはファイル avs_finish が存在するかをチェックします。このファイルは、スクリプトファイルで作成します。スクリプトファイルで avs_exist、または、avs_finish を作成するにはＣＬＩ言語の sh を用い以下のよう

に記述します。

(スクリプトファイルの例)

1) avs_exist の挿入例

```
net_read /home/guest/avsuser/vtr/vtr_network.net
sh touch /home/guest/avsuser/vtr/avs_exixt
```

2) avs_finish の挿入例

```
param_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/data_vtr
/vtr_data020.fld
sh touch /home/guest/avsuser/vtr/avs_finish
AVSmessage OK
```

5.2.1で作成したスクリプトファイル vtr_cli.scr に以下の内容を加えます。また、可視化したイメージを表示し、録画の開始から終了までの間ＡＶＳの実行を小休止（約１０秒）させるのに script_sleep を使います。

(スクリプトファイルの例)

```
sh touch /home/guest/avsuser/avs_exist
script_sleep 11
```

完成した動画の録画用のスクリプトファイルを以下に示します。

```
net_clear

net_read /home/guest/avsuser/vtr/vtr_network.net
sh touch /home/guest/avsuser/vtr/avs_exist
script_sleep 11

parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/data_vtr/
vtr_data020.fld
sh touch /home/guest/avsuser/vtr/avs_exist
script_sleep 11

parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/data_vtr/
vtr_data030.fld
sh touch /home/guest/avsuser/vtr/avs_exist
script_sleep 11

parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/data_vtr/
vtr_data040.fld
sh touch /home/guest/avsuser/vtr/avs_exist
script_sleep 11

parm_set "file browser.user.1":"File Browser" /home/guest/avsuser/vtr/data_vtr/
```

```
vtr_data050.fld
sh touch /home/guest/avsuser/vtr/avs_exist
script_sleep 11

sh touch /home/guest/avsuser/vtr/avs_finish

AVSmessage Ok
```

5.2.7 モニターとフレームスキャンコンバータの準備（動画の録画の準備）

1) モニターの電源を投入します。

2) フレームスキャンコンバータ POWER ボタンを押して電源を投入します。（図 5.5）

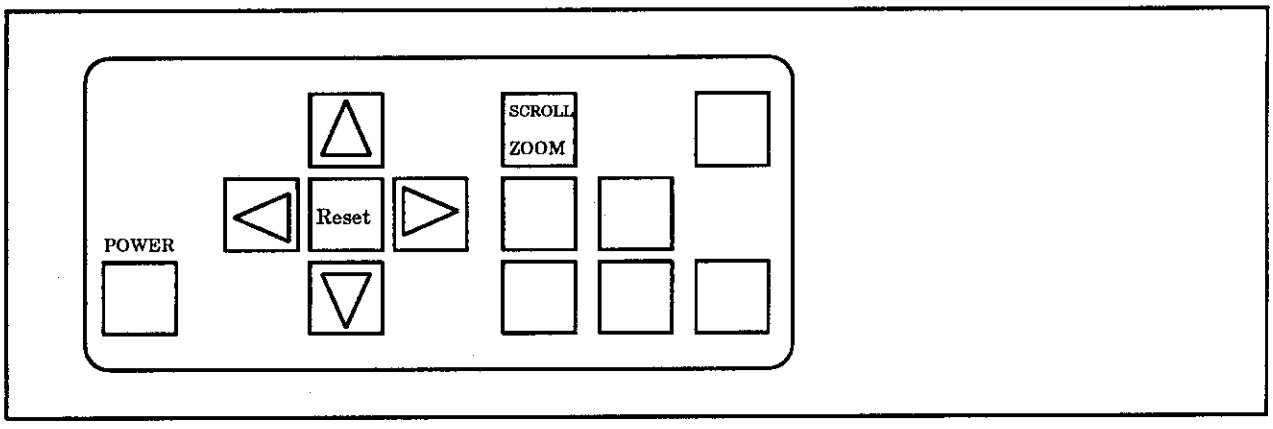

図 5.5 フレームスキャンコンバータ

5.2.8 ＶＴＲの準備（動画の録画の準備）

1) タイムコードが記録されたテープをセットします。

2) [INT/EXT] スイッチ（図 5.3の m）を INT にします。

3) [PRESET/REGEN] スイッチ（図 5.3の l）を PRESET にします。

4) [FREE/REC] スイッチ（図 5.3の k）を FREE にします。

5) [DF/NDF] スイッチ（図 5.3の j）を NDF にします。

6) [VITC REC] スイッチ（図 5.3の i）を ON にします。

7) [REMOTE] スイッチ（図 5.2の b）を 9pin にします。

8) [COUNTER] スイッチ（図 5.2の a）を TC にします。

5.2.9 ＥＷＳの準備（動画の録画の準備）

1) ＥＷＳにログインして、Ｘウィンドウを起動します。

2) ビデオテープに可視化したイメージを録画する時にスクリーンセーバが起動しないように以下のコマンドを実行します。

```
% xset s off
```

3) ＡＶＳを起動して、録画に使用するネットワークを読み込みます。

4) ジオメトリビューワーに表示されたイメージがモニターの真中にくるようにフレームスキャンコンバータを調整します。以下のボタンを使います。

- SCROLL/ZOOM ボタン
  SCROOL　矢印ボタンで画面がスクロールします。
  ZOOM　矢印ボタンで画面がズームします。
- FULL/WINDOW ボタン
  FULL　ディスプレイ全体をモニターに表示します。
  WINDOW　ディスプレイの 1/4 をモニターに表示します。

5) Network Tools メニューの Write Network ボタンをクリックしてネットワークを保存します。

6) ＡＶＳを終了します。

7) ３つのウィンドウを開きます。

5.2.10 動画の録画

ＥＷＳ上に開いた３つのウィンドウをそれぞれ以下のように使用して動画を録画します。

(1) １つ目のウィンドウでサーバプログラムを起動します。

1) 前準備として前回の録画を途中でやめた場合は avs_exist、正常終了した場合は avs_finish ファイルが作成されて残っているので以下のように消去します。

```
a) 途中でやめた場合
% rm avs_exist
b) 正常終了した場合
% rm avs_finish
```

2) 以下のコマンドでサーバプログラムを起動します。サーバプログラム server05 はＡＶＳが可視化したイメージを１枚につき５フレームずつビデオテープに録画していきます。

```
% server05
```

3) 録画開始時間と開始フレームを指定します。以下では0時間0分0秒の0フレームを指定しています。フレームの指定は0～29です。

```
% Begin recording at SMPTE timecode ? 00:00:00:00
```

4) 全録画フレーム数を指定します。大きめに設定しておけば問題ありません。以下では1000フレームを指定しています。

```
% Number of frames to record ? 1000
```

以上を実行するとＶＴＲは指定した録画開始時間までビデオテープを巻き戻し、また、早送りしクライアントプログラムの実行を待ちます。

2) クライアントプログラムを以下のコマンドで起動します。クライアントプログラムを実行するとサーバ側と通信を行ない、ＶＴＲは録画開始時間より3秒前で停止し、録画待機状態となります。

```
% client
```

3) サーバプログラムとクライアントプログラムが実行されていることを確認した上で、以下のようにスクリプトファイルを起動します。スクリプトファイルが実行されビデオ録画が自動的に開始します。

```
% avs -cli "script -play vtr_cli.scr"
```

4) 以上の操作で動画の録画の操作は終了です。

5.3 ＡＶＳビデオキットによる動画の作成

ＡＶＳビデオキットは、ＡＶＳのモジュールで動画のコマ撮りを行えます。ＡＶＳビデオキットは動画のコマ撮りをするモジュールです。コマ撮りをするためにネットワークコントロールパネル上で録画開始フレーム、１画面録画フレーム数、録画フレーム数などを設定します。これだけでビデオとＡＶＳを自動的に制御し動画を作成してくれます。

ＡＶＳビデオキットの構成

ＡＶＳビデオキットは、主に以下のモジュールから成っています。モジュールの実行ファイルは、ディレクトリ /usr/avs/vtr 以下にあります。

- v_lan controller モジュール (/usr/avs/vtr/vtr_cont/v_lan)
  ビデオの制御をするためのモジュールです。
- animated geom モジュール (/usr/avs/vtr/tools/animated_geom/geom_anim)
  ジオメトリデータを連続的に読み込むためのコルーチンモジュールです。
- animated image モジュール (/usr/avs/vtr/tools/animated_image/image_anim)
  イメージデータを連続的に読み込むためのコルーチンモジュールです。
- animated field モジュール (/usr/avs/vtr/tools/animated_field/fld_anim)
  フィールドデータを連続的に読み込むためのコルーチンモジュールです。
- animated ucd モジュール (/usr/avs/vtr/tools/animated_ucd/ucd_anim)
  ＵＣＤデータを連続的に読み込むためのコルーチンモジュールです。
- frame_no モジュール (/usr/avs/vtr/tools/frame_no/frame_no)
  録画しているフレームナンバーを出力するためのモジュールです。

5.3.1 コルーチンモジュールによる動画の作成

例えば animated float モジュールは、常に実数を決められた刻み（例：0.1）で出力するコルーチンモジュールの１つでありＡＶＳがサポートしているコルーチンモジュールには以下のものがあります。このコルーチンモジュールの特性を利用しＡＶＳビデオキットにより動画を作成する方法について説明します。

- animated integer
- animated float
- particle advector

以下にコルーチンモジュールを使ったネットワーク例を示します。

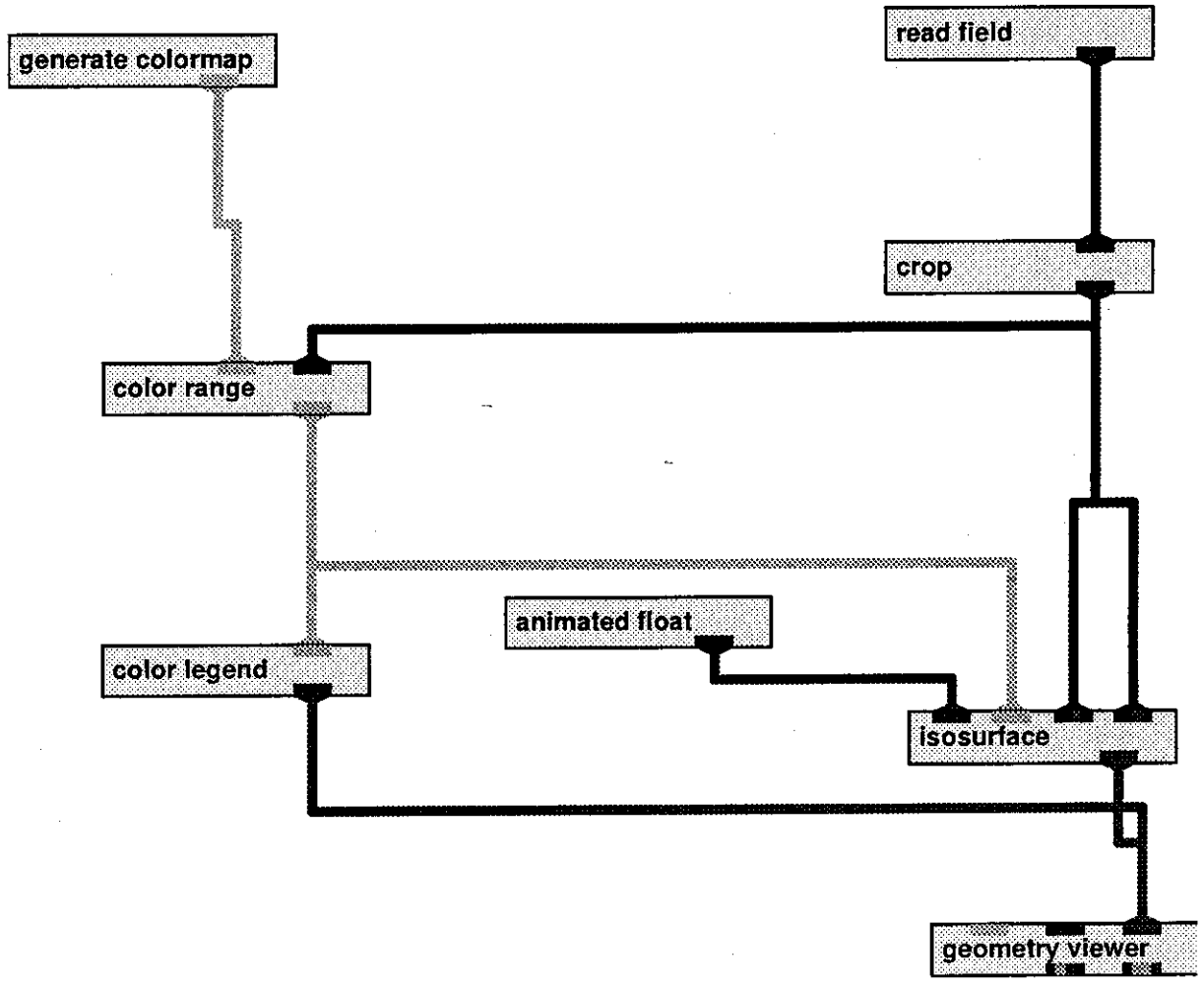

図 5.6 コルーチンモジュールを使ったネットワーク例 animated_isosurface.net

(1) animated_isosurface.net の概要

animated_isosurface.net は、等値面を生成するネットワークです。等値面の値（実数値）を animated float モジュールで連続的に与えることで生成する等値面を変化させて動画を実現します。

上の例のようにコルーチンモジュールを用いたネットワークでは、コルーチンモジュールから数値などが出力されるたびに可視化したイメージを表示します。可視化したイメージの表示が終了した時点で再びコルーチンモジュールが数値などを出力して次のイメージを出力させます。

(2) animeted float モジュールの使用方法

1) ネットワークコントロールパネル上の animated float ボタンをクリックすると以下のようなコントロールウィジェットを表示します。

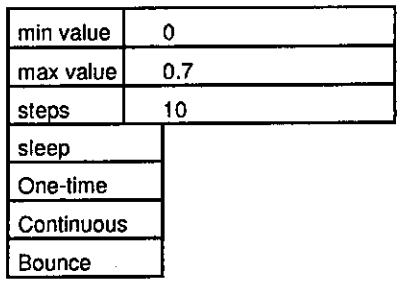

図 5.7 animated float モジュール

2) min value、max value の右側にカーソルを移動して出力する実数値の開始値、終了値を入力します。

3) steps (刻み) の右側にカーソルを移動して出力するデータのステップ数を入力します。animeted float モジュールは、min value から max value までを設定したステップ数で出力します。例えば、図5.7のように設定した場合には、以下のように順次出力します。

```
0.0 0.07 0.14 0.21 0.28 0.35 0.42 0.49 0.56 0.63 0.7
```

4) 実行方法を以下の3通りの中から選択してクリックします。

* One-time
  min value から max value まで1回だけ出力します。
* Continuous
  min value から max value までを繰り返し出力します。
* Bounce
  min value から max value を1回出力した後に、今度は max value から min value まで逆に出力していきます。この一連の実行を繰り返して行ないます。

5) 白く光っているコントロールウィジェット sleep をクリックすると、実行を開始します。

6) 実行を途中で中止したい場合は、sleep ボタンをクリックするとコントロールウィジェット sleep ボタンが白く光ってコルーチンの実行を停止します。

ネットワーク animated_isosurface.net は、コルーチンモジュールである animated float モジュールの動作により実行を開始します。

5.3.2 ＡＶＳビデオキットによる動画の録画方法

ＡＶＳビデオキットを使って動画をビデオテープに録画する方法について、以下では (1) に動画の準備を (2) にコマ撮りの実行を説明します。

(1) 録画の準備

1) ＶＴＲの電源を投入し、タイムコードを記録したビデオテープをセットします。ビデオテープには、事前にタイムコードを記録しておきます。

2) 以下のようにＶＴＲの設定を行ないます。

   [INT/EXT] スイッチを INT にします。

   [PRESET/REGEN] スイッチを REGEN にします。

   [FREE/REC] スイッチを FREE にします。

   [DF/NDF] スイッチを NDF にします。

   [VITC REC] スイッチを ON にします。

   [REMOTE] スイッチを 9pin にします。

   [COUNTER] スイッチを TC にします。

3) モニター、フレームスキャンコンバータの電源を投入します。

4) ＡＶＳを起動します。

5) ネットワークエディタを起動し、録画するネットワークを読み込みます。ここでは、ネットワーク例 animated_isosurface.net を読み込みます。

6) ビデオの制御をするためのモジュールである v_lan controller モジュールをネットワークエディタメニューの Module Tools のサブメニュー Read Module(s) ボタンを使って
   /usr/avs/vtr/vtr_cont/v_lan
   をモジュールパレットに追加します。

7) v_lan controller モジュールをワークスペースに移動して geometry viewer モジュールに接続します。（図 5.8）

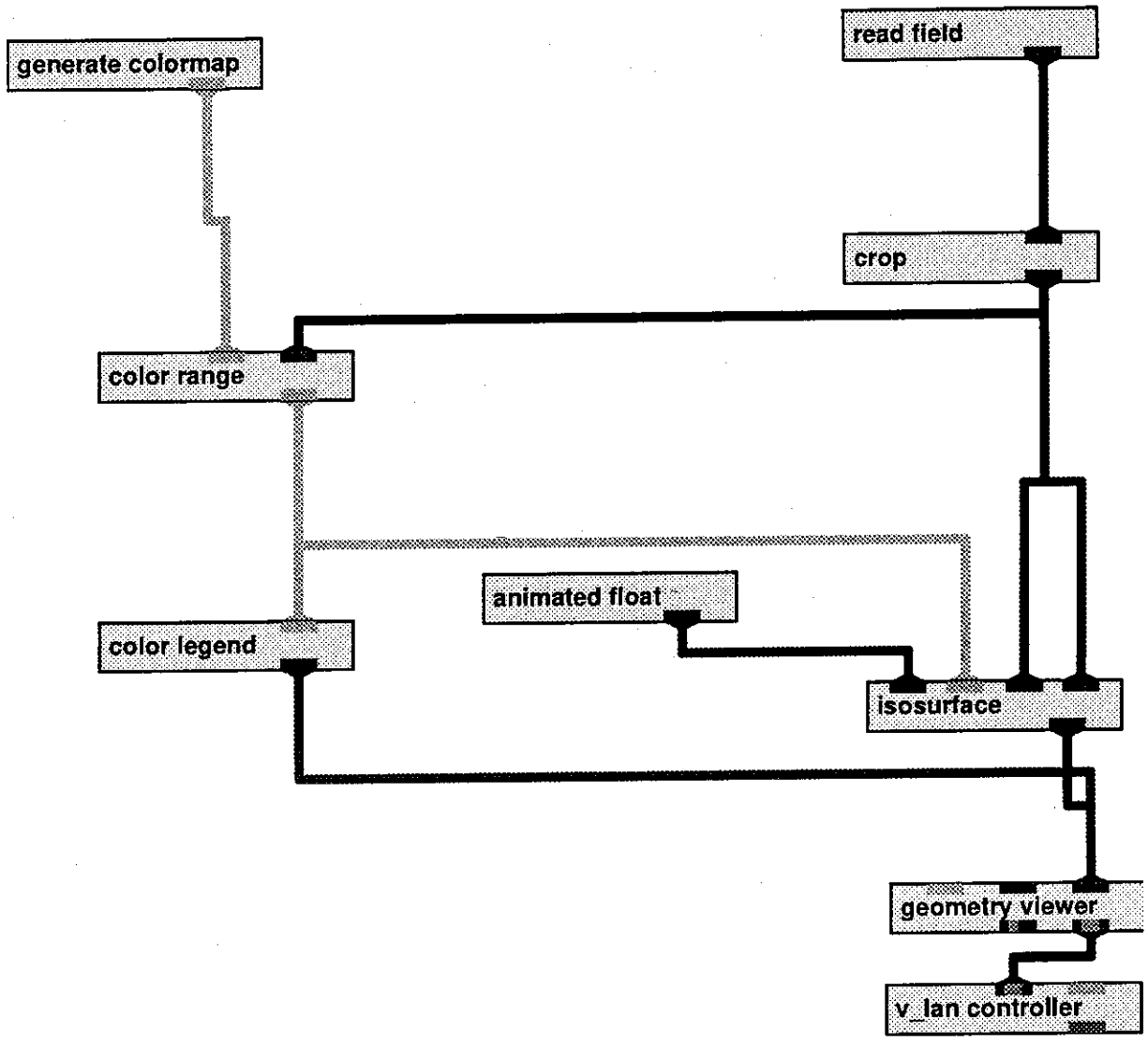

図 5.8 v_lan controller モジュールの接続

8) ネットワークコントロールパネルの v_lan controller ボタンをクリックすると図 5.9のような v_lan controller モジュールのコントロールウィジェットを表示します。

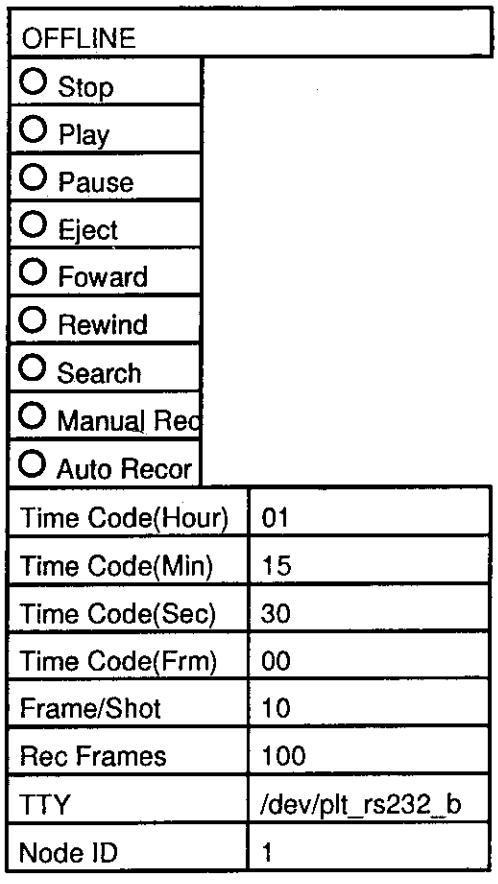

図 5.9 v_lan controller モジュールのコントロールウィジェット

9) TTY の右側に、V-LAN を接続しているシリアルポート番号を入力します。シリアルポート番号については、システム管理者に尋ねて下さい。（情報システム管理課にある furies では /dev/plt_rs232_b です。）

10) v_lan controller モジュールのコントロールウィジェットの以下のボタン（図 5.9）をクリックすると、ＶＴＲに対して以下の制御を行なうことができます。

Stop ボタン
　停止

Play ボタン
　再生

Pause ボタン
　一時停止

Eject ボタン
　ビデオテープの排出

Forward ボタン
　早送り

Rewind ボタン
　巻き戻し

Search ボタン
　頭出し

Manual Record ボタン
　普通の録画の実行（タレ流し的に録画する）

Auto Record ボタン
　コマ撮り録画の実行

11) ビデオテープの録画開始時間と１枚の画像を何フレームづつ録画するのかを v_lan controller モジュールのコントロールウィジェットで設定します。コントロールウィジェットにカーソルを移動して各値を入力します。ＶＴＲは、ビデオテープに映像を１秒間に 30 フレーム単位で録画していきます。

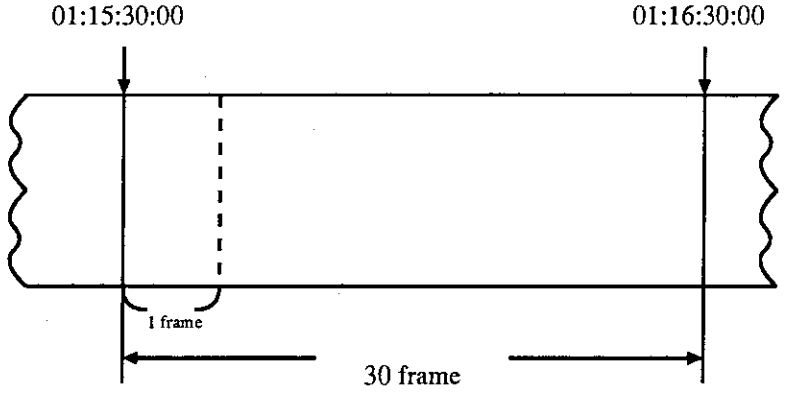

図 5.10　フレーム

例としてビデオテープのタイムコードの1時間15分30秒の1フレームから1枚の描画につき10フレームづつ録画し、全部で100フレーム分の録画をしたい場合を示します。

Time Code(Hour.)
録画開始のタイムコード（時間）を入力します。例では、01と入力します。

Time Code(Min.)
録画開始のタイムコード（分）を入力します。例では、15と入力します。

Time Code(Sec.)
録画開始のタイムコード（秒）を入力します。例では、30と入力します。

Time Code(Frm.)
録画開始のフレーム番号を入力します。例では、00と入力します。

Frame/Shot
1枚の映像を何フレームづつ録画するのかを入力します。
例では、10と入力します。

Rec Frames
全録画フレーム数を入力します。例では、100と入力します。

(2) 動画のコマ撮りの実行

録画の準備ができたら以下のようにして動画のコマ撮りを行ないます。

1) 録画したい画像をフレームスキャンコンバータでモニターに映るように調節します。調節方法については、スクリプトファイルによる動画の録画を参照して下さい。

2) v_lan controllerモジュールのコントロールウィジェットのAuto Recordボタン（図5.9）をクリックします。

3) ネットワークを実行します。v_lan controllerモジュールにデータが送られるとＶＴＲが指定したタイムまで早送り又は巻き戻しを行ないます。

4) 可視化したイメージの表示（フレーム）を開始します。可視化したイメージを1フレーム表示するとＶＴＲに設定したフレーム数（例では10フレーム）だけビデオテープに録画します。

5) 録画を終了するとanimated floatモジュールは次のデータを出力してネットワークを実行し可視化したイメージを表示します。v_lan controllerモジュールにデータが送られるとビデオテープに録画を行ないます。

6) 指定した全録画フレーム数の録画が終了するか、ネットワークの実行が終了すると動画のコマ撮りが終了します。

5.3.3 連続データによる動画の録画

この方法では、同じネットワークで読み込むデータファイルを順次変えていくことで動画を作成します。

ネットワーク animated field.net で可視化したイメージを使って動画を作成し録画する例を示します。

1) 以下のようなネットワークを作成します。

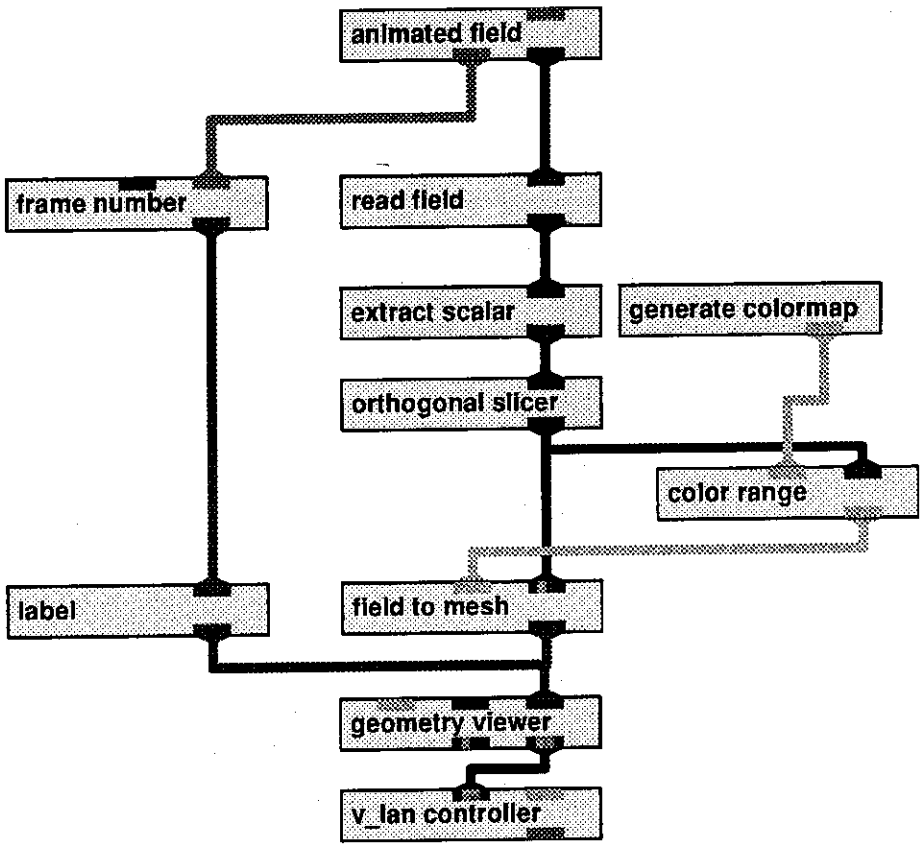

図 5.11 ネットワーク animated field.net

animated field モジュールは、フィールドデータが保存されているファイルを順次読むことを指示するコルーチンモジュールです。ネットワーク animated field.net で読み込むファイルは以下のとおりです。ディレクトリ /home/guest/avsuser/vtr/data_vtr にあります。

```
vtr_data010.fld
vtr_data020.fld
vtr_data030.fld
 ..................
vtr_data190.fld
vtr_data200.fld
```

2) モニター、フレームスキャンコンバータ、ＶＴＲの準備をします。

3) ネットワークコントロールパネルの v_lan controller ボタンをクリックして v_lan controller モジュールのコントロールウィジェット（図 5.9）を表示します。各設定値を入力します。

4) v_lan controller モジュールの Auto Record ボタンをクリックします。

5) 録画開始時間などを入力します。

6) ネットワークコントロールパネルの animated field ボタンをクリックして animated field モジュールのコントロールウィジェットを表示します。（図 5.12）

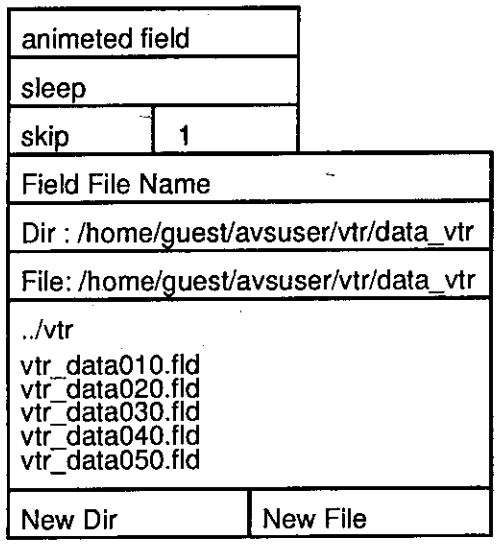

図 5.12 animated field モジュール

7) skip の右側（図 5.12）に次にいくつ目のファイルを読み込むかを設定します。連続して読み込む場合は、1 に設定します。

8) sleep ボタン（図 5.12）をクリックするとコマ撮りを開始します。

おわりに

本報告書では可視化ソフトウェアＡＶＳの使用方法を中心にＡＶＳを使った動画の作成方法、本システムのような画像処理環境を整備するための情報について解説した。本報告書が容易にＡＶＳを利用できるための一助となることを期待する。

謝辞

本報告書を執筆する機会を与えて下さいました計算科学技術推進センター長 浅井 清氏、次長 秋元 正幸氏及び情報システム管理課長 藤井 実氏、本報告書の内容について検討して下さいました数値実験技術開発グループの渡辺 正氏に深く感謝致します。また、 HP9000/730 CRX-24z 及び AVS 等を整備した情報システム管理課の方々に感謝致します。

参考文献

1) 加藤克海, 渡辺 正, 町田昌彦, 蕪木英雄 動画像処理システムの開発 JAERI-Tech(平成 6 年)

2) 株式会社クボタ AVS による実践ビジュアライゼーション オーム社 (1995 年)

3) クボタコンピュータ株式会社 AVS USER'S GUIDE クボタコンピュータ株式会社 (1991 年)

4) 旭化成情報システム（株） ＡＶＳビデオチェックキット取り扱い説明書 Ver.2.0 旭化成情報システム（株） (1993 年)

おわりに

本報告書では可視化ソフトウェアＡＶＳの使用方法を中心にＡＶＳを使った動画の作成方法、本システムのような画像処理環境を整備するための情報について解説した。本報告書が容易にＡＶＳを利用できるための一助となることを期待する。

謝辞

本報告書を執筆する機会を与えて下さいました計算科学技術推進センター長 浅井 清氏、次長 秋元 正幸氏及び情報システム管理課長 藤井 実氏、本報告書の内容について検討して下さいました数値実験技術開発グループの渡辺 正氏に深く感謝致します。また、 HP9000/730 CRX-24z 及び AVS 等を整備した情報システム管理課の方々に感謝致します。

参考文献

1) 加藤克海, 渡辺 正, 町田昌彦, 蕪木英雄 動画像処理システムの開発 JAERI-Tech(平成 6 年)

2) 株式会社クボタ AVS による実践ビジュアライゼーション オーム社 (1995 年)

3) クボタコンピュータ株式会社 AVS USER'S GUIDE クボタコンピュータ株式会社 (1991 年)

4) 旭化成情報システム（株） ＡＶＳビデオチェックキット取り扱い説明書 Ver.2.0 旭化成情報システム（株） (1993 年)

おわりに

本報告書では可視化ソフトウェアＡＶＳの使用方法を中心にＡＶＳを使った動画の作成方法、本システムのような画像処理環境を整備するための情報について解説した。本報告書が容易にＡＶＳを利用できるための一助となることを期待する。

謝辞

本報告書を執筆する機会を与えて下さいました計算科学技術推進センター長 浅井 清氏、次長 秋元 正幸氏及び情報システム管理課長 藤井 実氏、本報告書の内容について検討して下さいました数値実験技術開発グループの渡辺 正氏に深く感謝致します。また、 HP9000/730 CRX-24z 及び AVS 等を整備した情報システム管理課の方々に感謝致します。

参考文献

1) 加藤克海, 渡辺 正, 町田昌彦, 蕪木英雄 動画像処理システムの開発 JAERI-Tech(平成 6 年)

2) 株式会社クボタ AVS による実践ビジュアライゼーション オーム社 (1995 年)

3) クボタコンピュータ株式会社 AVS USER'S GUIDE クボタコンピュータ株式会社 (1991 年)

4) 旭化成情報システム（株） ＡＶＳビデオチェックキット取り扱い説明書 Ver.2.0 旭化成情報システム（株） (1993 年)

付録 A

GWSの性能

様々なGWSが発表されている中からヒューレットパッカード（以下HPという）、シリコングラフィックス（以下SGIという）及びSUNのGWSを例にし、GWSの導入のために必要な情報としてA.1にグラフィックスアクセラレータの性能、A.2にGWSの基本構成の性能と価格を示します。

A.1 グラフィックスアクセラレータの性能

1) グラフィックスアクセラレータの性能を示す値に使われる用語

GWSは一般に、3次元のグラフィックス処理をより高速に表示するとともに、より高詳細な表面近似（レンダリング）することが要求されます。グラフィックスアクセラレータは、これらのグラフィックス処理を専門に行なうハードウェアです。このグラフィックスアクセラレータの性能を示す値には主に以下のようなものが一般に使われています。

- GPC ベンチマーク

  グラフィックスの性能値は、これまでは、ベクタ / 秒やポリゴン / 秒で評価されてきました。しかし、最近では、GPC ベンチマークを参考にするのが一般的になっています。GPC（グラフィックスアクセラレータの性能を測定する標準ベンチマークの団体名）ベンチマークは、実際のモデルデータを使ってそれに良く使われる操作（回転、ズームなど）の性能を測定したものです。

  - PLBsurf93

    3次元サーフェスの表示性能を示す指標です。サーフェスは、立体を面データのみで表現したモデルです。

  - PLBwire93

    3次元ワイヤフレームの表示性能を示す指標です。ワイヤーフレームは、立体の骨組み構造を直線や曲線で表現したモデルです。

- AA(アンチエリアシング) ポイント / 秒、AA(アンチエリアシング) ベクタ / 秒

  1秒間に描画できるポイント数、ライン数。アンチエリアシング処理とは斜線や線画の辺の表示において、階段上のギザギザをぼかして目立たなくする処理です。

- 三角形メッシュ

  連続する三頂点の三角形で多角形を描画します。

- 三角形メッシュ (G, Z, L, T)

  三角形メッシュによる描画に以下の処理を施したものです。

- G　グローシェーディング処理。陰影効果を出す処理。
- Z　Zバッファ。陰面消去処理。
- T　テクスチャマッピング処理。
- L　光源効果処理。

2) グラフィックスアクセラレータの性能の一般的な見方

1. ポイントやベクタ性能が高いほどCAD/CAE分野のワイヤーフレーム表示に向いたGWSです。
2. 三角形メッシュや四角形の描画の性能が高いほどCGアニメーションやVR（バーチャルリアリティ）などのリアルタイムCG分野に向いたGWSです。
3. ピクセル描画の性能が高いほどテクスチャマッピングやアンチエリアシング処理が高速となるため写実的な映像の生成に向いたGWSです。

3) HPのグラフィックスアクセラレータ

VISUALIZEシリーズは、HPが1995年6月に発表したグラフィックアクセラレータです。VISUALIZEシリーズには、以下のようなものがあります。

- VISUALIZE-8/24
  対応機種は、HP9000モデル715-64/80/100, モデル725/100, モデルJ200, モデルJ210です。
- VISUALIZE-48
  対応機種は、HP9000モデル J200, J210です。特長としてVISUALIZE-48は、ハードウェアテクスチャーマッピング機能（オプション）をサポートしています。

	VISUALIZE 48	VISUALIZE 8/24
PLBwire93	130～182	66～87
PLBsurf93	200～235	88～144
3次元ベクタ	6500k	2300k
三角形ポリゴン	1850k	670k
四角形ポリゴン	890～925k	215～315k

4) SGIのグラフィックスアクセラレータ

SGIで発表しているグラフィックスアクセラレータには、主に1995年7月に発売したデスクトップワークステーション用のIMPACTと既存の最上位グラフィックスアクセラレータReality Engine2等があります。

- High IMPACT, Maximum IMPACT

ＣＡＤ分野だけでなく3次元ＣＧやアニメーション、シミュレーションなどの3次元ＧＷＳの利用分野全体をターゲットに作られました。HighIMPACTとMaximum IMPACTは、ハードウェアによるテクスチャマッピング機能を標準で備えています。

High IMPACTの対応機種は、Indigo2のR4400/200MHz搭載モデルとR4400/250MHz搭載モデルです。

Maximum IMPACTの対応機種は、Indigo2のR4400/250MHz搭載モデルとR8000/75MHz搭載モデルです。

以下にHigh IMPACTとMaximum IMPACTと既存のグラフィックスアクセラレータExtremeの性能比較表を示します。

	High IMPACT	Maximum IMPACT	Extreme
3次元ベクタ	1.2M	2.0M	255k
三角形 (G)	1.1M	2.2M	616k
三角形 (G,Z,L)	730k	1.46M	400k
三角形 (G,Z,T,L)	560k	1.1M	—
独立四角形 (G,Z,L)	300k	600k	145k

- Reality Engine2,VTX,Extreme

Reality Engine2は、世界最高速のグラフィックス描画性能を備えています。対応機種は、OnyxとPower Onyxです。

VTXは、Reality Engine2のアーキテクチャをベースとして作られたグラフィックスです。対応機種は、Onyxです。

Extremeは、汎用のグラフィックスです。対応機種は、OnyxとPower OnyxとIndigo2です。

以下にReality Engine2、VTX及びExtremeの性能比較を示します。

	Extreme	VTX	RealityEngine2
ポイント 3D	1.4M	1.98M	2.82M
ライン 3D	1.3M	1.05M	1.57M
AA ライン 3D	693k	1.0M	1.39M
独立三角形	201k	312k	480k
三角形 (2D)	646k	955k	1.35M
三角形 (G,Z)	464k	813k	1.07M
三角形 (G,Z,L)	370k	813k	1.0M
四角形 (G,Z)	225k	351k	535k

- Sun のグラフィックスアクセラレータ

  Sun で発表しているグラフィックスグラフィックスアクセラレータには、以下のようなものがあります。

  - TGX+

    CAD/CAE などのメカニカルデザインやエレクトリカルデザイン処理に対応します。対応機種は、S-4/5 のモデル 70/85/110、S-4/20 のモデル 50/502/514/61/612/71/712、S-4/20 のモデル HS14/21/22 等です。

  - SX8

    デザイン処理やシミュレーションやビデオ再生画面などに対応します。対応機種は、S-4/20 のモデル 50/502/514/61/612/71/712,S-4/20 のモデル HS14/21/22 等です。

  - ZX,TZX(Turbo ZX)

    スピードを要求されるシミュレーションやアニメーションなどに対応します。ZX の対応機種は、S-4/20 のモデル 50/502/61/612/71/712、S-4/20 のモデル HS14/21/22 等です。TZX の対応機種は、S-4/20 のモデル 71/712、S-4/20 のモデル HS21/22 です。

  - AG-600,AG-1600

    3D シミュレーションのみならずデザイン分野やプレゼンテーション、アミューズメント分野にも対応します。対応機種は、S-4/20 のモデル 50/502/61/612/71/712、S-4/20 のモデル HS14/21/22 等です。

	SX8	TGX+	ZX	TZX	AG-600	AG-1600
2D ライン	792k	1700k	—	—	—	—
3D ライン	346k	560k	750k	950k	600k	1600k
三角形 (G	35k	—	350k	616k	600k	1600k

A.2 GWSの構成例

以下にGWSの構成例を示します。

GWS	HP9000 J210	HP9000 J200	HP9000 J210	HP9000 J200
CPU	PA-7200	PA-7200	PA-7200	PA-7200
	120MHz	100MHz	120MHz	100MHz
SPECint92	168.7	222.5	168.7	222.5
SPECfp92	269.2	139.5	269.2	139.5
グラフィックス	VISUALIZE-48	VISUALIZE-48	VISUALIZE-8/24	VISUALIZE-8/24
PLBwire93	182	130	87	79
PLBsurf93	235	200	144	126
三角形 / 秒	—	—	—	—
四角形 / 秒	92 万 5000	89 万	31 万 5000	28 万 5000
価格	940 万	830 万	—	—
メモリ (Byte)	64M	64M	—	—
ディスク (Byte)	2G	2G	—	—

GWS	HP-9000 J715	HP-9000 J715	HP9000 J715	HP9000 J715
CPU	PA-7100LC	PA-7100LC	PA-7100LC	PA-7100LC
	64MHz	80MHz	100MHz	100MHz
SPECint92	80.6	96	115.1	132.2
SPECfp92	109.4	123.2	138.7	184.6
グラフィックス	VISUALIZE-8	VISUALIZE-8	VISUALIZE-24	VISUALIZE-24
PLBwire93	66	77	82	85
PLBsurf93	88	105	114	131
三角形 / 秒	—	—	—	—
四角形 / 秒	21 万 5000	28 万	31 万	—
価格	280 万	310 万	380 万	430 万
メモリ (Byte)	64M	64M	64M	64M
ディスク (Byte)	2G	2G	2G	2G

GWS	Indigo2	Indigo2	Indigo2	Indigo2
CPU	R8000	R4400	R4400	R4400
	75MHz	250MHz	250MHz	200MHz
SPECint92	113	176	176	140
SPECfp92	269	165	165	131
グラフィックス	MaximumIMPACT	MaximumIMPACT	High IMPACT	High IMPACT
PLBwire93	—	—	—	—
PLBsurf93	—	—	—	—
三角形 / 秒	146 万	146 万	73 万	73 万
四角形 / 秒	60 万	60 万	30 万	30 万
価格	922 万 3000	430 万	616 万 6000	540 万 1000
メモリ (Byte)	64M	64M	64M	64M
ディスク (Byte)	2G	2G	2G	2G

GWS	Indigo2	SPARCstation20
CPU	R8000	SuperSPARC2
	75MHz	75MHz
SPECint92	113	121
SPECfp92	269	125.8
グラフィックス	Extreme	Turbo ZX
PLBwire93	—	54
PLBsurf93	—	93
三角形 / 秒	—	46 万 6000
四角形 / 秒	—	23 万
価格	800 万	525 万 5000
メモリ (Byte)	64M	64M
ディスク (Byte)	2G	1G

GWS	Power Onyx	Onyx	Onyx
CPU	R8000	R4400	R4400
	90MHz	250MHz	200MHz
SPECint92	132.2	176	140
SPECfp92	396.1	165	131
グラフィックス	Reality Engine2	Reality Engine2	Reality Engine2
PLBwire93	—	—	—
PLBsurf93	—	—	—
三角形 / 秒	—	—	—
四角形 / 秒	—	—	—
価格	2544 万 8000	1602 万 8000	1522 万 8000
メモリ (Byte)	64M	64M	64M
ディスク (Byte)	2G	2G	2G

- SPEC ベンチマーク

  ワークステーションのＣＰＵの性能を評価をするための指標の１つです。これまでは、MIPS や FLOPS で性能を評価していました。最近では、SPEC ベンチマークで性能を表すようになりました。SPEC ベンチマークは、いくつかのプログラムを実行して、その実行時間から算出する値です。

  ＧＷＳにおけるグラフィック性能は、個々のグラフィックスの性能の他に、そのグラフィックスアクセラレータにいかに高速に３Ｄデータを送るかも重要であり、特に映像データは、浮動小数点データが主であるために、CPU の SPECfp 値もＧＷＳを検討する上で重要な項目となります。

  - SPECint92

    整数演算の性能を示すベンチマークです。

  - SPECfp92

    浮動小数点演算の性能を示すベンチマークです。

付録 B

索引

[A,a]
AA ポイント, 132
AA ベクタ, 132
Absolite, 82
animated float モジュール, 123
animated field モジュール, 129,130
ASCII 記述ファイル, 64
ASCII ヘッダー, 58
AVS の起動方法, 5
AVS Field Editor, 95
AVS Input Port Editor, 95
AVS Output Port Editor, 95
AVS Parameter Editor, 96
AVS ビデオキット, 121
.avsrc, 2

[B,b]
BoundingBox, 78

[C,c]
Cameras メニュー, 87
Clear Network, 14
CLI 言語, 111
color legend モジュール, 42
color range モジュール, 42
computation function, 102
crop モジュール, 46
Current Object Browser, 49,75

[D,d]
DataDirectory, 4
discription function, 99
Disable Flow Excutive, 15,28
Disable Module, 28
DisplayPixmapWindow, 4

[E,e]
Editing Tools メニュー, 35
extract scalar モジュール, 38
EWS 110,119

[F,f]
field to mesh モジュール, 38,44

[G,g]
generate colormap モジュール, 39
geometry viewer モジュール, 39,49
Geometry データ, 77
GPC ベンチマーク, 132
GWS, 132

[H,h]
hedgehog モジュール, 48
HelpPath, 3
HP, 132,133

[I,i]
image to postscript モジュール, 44
initialization function, 98
irregular, 55,59,60
isosurface モジュール, 47

[L,l]
Labels メニュー, 89
Layout Editor メニュー, 35
Lights メニュー, 86

[M,m]
ModuleLibraries, 4

Module Generator モジュール, 93
Module Tools メニュー, 31

[N,n]
NetworkDirectory, 3
Network Tools, 9

[O,o]
Object データ, 77
Objects メニュー, 38
orthogonal slicer モジュール, 38

[P,p]
PLBsurf93, 132
PLBwire93, 132
print field モジュール, 62
PrintNetwork, 4

[R,r]
Read Network, 10,11
rectilinear, 54,59
Relative, 82
Restore Parameters, 15

[S,s]
Save Parameters, 15
Scene, 73
Scene データ, 77
SGI, 132,134
Show Module Documentation, 28
SPEC ベンチマーク, 142
SPECint92, 142
SPECint92, 142
Sun, 132,135

[T,t]
Transformation Option パネル, 85

[U,u]
uniform, 54,59

[V,v]
V-LAN, 110
v_lan controller モジュール, 125,126,127,128
volume bounds モジュール, 48
VTR, 110,115,124

[W,w]
WindowMgr, 3
write field モジュール, 66
Write Network, 13
Write Network の保存情報, 14

3D bar chart モジュール, 45

[あ]
アンビニエントパネル, 86
イレギュラーフィールド, 55
イメージビューワーサブシステム, 72
オブジェクト, 75
オブジェクトの階層, 75

[か]
カット&ペースト, 32
カメラ, 75
カラーデータ, 52
カラーマップエディタ, 41
カレントオブジェクト, 79
幾何変換, 79
記述関数, 99
クライアントプログラム, 116,120
グラフィックスアクセラレータ, 132
グラフビューワーサブシステム, 72
計算関数, 102
コピー&ペースト, 32
コルーチンモジュール, 92,121
コントロールウィジェット, 23
コントロールウィジェットの選択, 25
コンター図, 37,50

[さ]
彩度, 40
サーバプログラム, 116,119
座標領域, 59
サブルーチンモジュール, 93
三角形メッシュ, 132
ジオメトリデータ, 52,66
ジオメトリビューワーサブシステム, 72,73
ジオメトリビューワーメニュー, 83
色相, 40
出力ポート, 15,56
出力領域, 103
初期化関数, 98

処理ルーチン, 97
シリアルポート番号, 126
シーン, 75
スカラーデータ, 36
スタートアップファイル, 2,78
スクリプトファイル, 111
スライーダーウィジェット, 25

[た]
ダイヤルウィジェット, 24
タイプインウィジェット, 24
タイムコード 114,124
鳥瞰図, 43 ディスプレイ番号, 5
データ領域, 59
等値面, 46
トグルウィジェット, 25

[な]
入力ポート, 15,56
ネットワーク, 6
ネットワークエディタ, 6
ネットワークエディタの起動, 7
ネットワークエディタの終了, 8
ネットワークエディタメニュー, 30
ネットワークの実行, 22
ネットワークの実行の停止, 15
ネットワークの保存, 13
ネットワークの読み込み, 10
ネットワークファイル, 6

[は]
バイナリ領域, 59
パラメータ, 23
パラメータポート, 27
パラメータエディタウィンドウ, 26
パラメータの保存, 15
パラメータのリセット, 15
ハンマー, 20
ビューウィンドウ, 74
ファイルブラウザ, 10,22
ファイルブラウザウィジェット, 25
フィールド, 54
フィールドデータ, 52,53
フィールドデータのフォーマット, 57
フィールドデータポート, 56
フィルター, 66
物理空間, 54,58
不透明度, 40
プリミティブデータ, 52
プリントアウト (レイアウト), 14
フレーム, 127
フレームスキャンコンバータ, 110,118,124
ベクトル図, 47,50
ベクトルデータ, 36
ポートの色, 16

[ま]
マクロモジュール, 33
明度, 40
メインメニュー, 5
モジュール, 6,17
モジュールエディタウィンドウ, 26
モジュール間の接続, 21
モジュール間の切断, 21
モジュールの移動, 18,20
モジュールのカテゴリ, 17
モジュールの検索, 18
モジュールのコンパイル, 109
モジュールの消去, 14,20
モジュールパレット, 17
モジュール名, 15
モジュールライブラリ, 17
モニター, 110,118,124

[や]
ユニフォームフィールド, 54

[ら]
ライティングパネル, 86
ライト, 75
ラインの色, 16
ラジオボタンウィジェット, 24
レクティリニアフィールド, 54
録画開始時間, 120,127

[わ]
ワークスペース, 6

その他
3 次元グラフ, 44
UCD データ, 52