JAERI-Tech
2000-077

シミュレーションモニタリングシステムの開発

2001年1月

加藤　克海*・渡辺　正・久米　悦雄

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，日本原子力研究所研究情報部研究情報課（〒319－1195 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（〒319－1195 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 Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 〒319－1195, Japan.

編集兼発行　日本原子力研究所

JAERI-Tech 2000-077

シミュレーションモニタリングシステムの開発

日本原子力研究所計算科学技術推進センター
加藤　克海*・渡辺　正・久米　悦雄

(2000年10月10日受理)

計算科学技術推進センターでは、原子力分野における計算科学の研究の一環として大規模数値シミュレーション技術に関する研究を行っており、シミュレーション結果の効率的な理解のための可視化並びに動画像処理技術の開発整備作業を進めている。本報告書では、シミュレーションの途中経過の実時間可視化、あるいは計算結果の連続可視化を行うためのシミュレーションモニタリングシステムの開発について記述する。シミュレーションモニタリングシステムは、汎用可視化ツールAVS5またはAVS/EXPRESSを使用しているため、様々な計算機環境での利用が可能である。

日本原子力研究所（東海駐在）：〒319-1195　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 2000-077

Development of the Simulation Monitoring System

Katsumi KATO*, Tadashi WATANABE and Etsuo KUME

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

(Received October 10, 2000)

Large-scale simulation technique is studied at the Center for Promotion of Computational Science and Engineering for the computational science research in nuclear fields. Visualization and animation processing techniques are developed for efficient understanding of simulation results. The development of the simulation monitoring system, which is used for real-time visualization of ongoing simulations or for successive visualization of calculated results, is described in this report. The standard visualization tool AVS5 or AVS/EXPRESS is used for the simulation monitoring system, and thus, this system can be utilized in various computer environments.

Keywords : Monitoring System, Simulation, Visualization, AVS5, AVS/EXPRESS

* RIST

目　　次

Contents

1. はじめに

日本原子力研究所・計算科学技術推進センターでは、原子炉で見られる複雑形状流路内の混相流などの複雑な熱流動現象を、粒子法を用いた数値シミュレーションを通して解明する計算科学研究を進めている。計算科学研究では、しばしば並列型スーパーコンピュータや計算サーバ用ワークステーションクラスターを用いて長時間のシミュレーションを行うが、近年の計算機の発展により、シミュレーションはより大規模（大容量）なものになってきている。大規模シミュレーションでは、理想的な条件下であらゆる変数の数値データが得られる反面、その膨大なデータの処理、保存が問題となっている。例えば、直接シミュレーションモンテカルロ法による3次元対流計算において1億6千万個の粒子を使用すると[1]、粒子の位置と速度だけで$7.3GB$になる。過渡変化が重要となる場合に時系列データを保存しようとすると、タイムステップ数だけこれらのデータが必要となる。並列計算機を用いることにより、計算速度の向上と同時に計算に使用できる変数の量、すなわちメモリーの容量も増加しており、計算結果を理解、判断するための効率的なデータ処理技術、あるいは大規模数値計算支援技術の研究・開発はますます重要なものとなってきている。

シミュレーションにもとづく計算科学研究において、データ処理技術として最も重要と考えられるものの一つは可視化技術である。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、シミュレーション結果の可視化を、高度な作画機能を持つ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行うということが普及してきた。可視化の結果は、ワークステーションの画面で直接見たり、あるいは論文等への使用のためにプリントするばかりでなく、時系列データを連続的に可視化表示したアニメーションとして、過渡現象の動的な理解や効果的なプレゼンテーションのためにも利用される[2]。さらに、シミュレーション結果の保存方法としてアニメーションビデオを作成することも行われており[3]、動画像処理技術も計算科学研究には欠くことのできないものとなっている。

可視化技術や動画像処理技術は、これまではシミュレーション結果に対するデータ処理技術として研究・開発が行われてきた。実際の計算科学研究の手順としても、シミュレーションはセンターマシンとしてのスーパーコンピュータや高計算性能の計算サーバ用ワークステーションで実行し、計算終了後に結果をグラフィックワークステーションやパソコン等に転送し、可視化、アニメーション作成を行うのが一般的であった。これは，スーパーコンピュータがフロント／バックエンドとして構成されていたり、計算をバッチジョブとしてサブミットする形式を採用しているため、計算の開始や終了が任意ではなく、ファイル操作を含めてパソコンのように使い勝手が良くないためである。しかしながら、計算科学研究で行われるシミュレーションは長時間に及ぶことが多いため、可視化による結果の理解が効率良く行われるとしても、シミュレーション終了後にのみデータ処理や可視化を行っていては、実質的なシミュレーション研究の効率を上げることはできない。シミュレーション研究の効率をあげ、計算機資源の有効利用を計るためには、パソコンのように計算を実行しながらその結果を描画し、任意に計算を中断し、パラメータを変更し再実行する，ということがスーパーコンピュータを用いた大規模シミュレーションにおいても必要となる[3]。

このため、計算科学技術推進センターでは、スーパーコンピュータや計算サーバ用ワークステーションで実行中のシミュレーションの途中経過を、逐次、画像処理サーバ用ワークステーションへ転送し、実時間可視化を行うリアルタイムモニターシステムを開発し [4] これまで使用してきた。同様の可視化システムはスーパーコンピュータのメーカーなどでも開発が行われており、例えば富士通では実時間可視化ソフトウェアとして VisLink を発表している [5]。VisLink は、画像用データを計算サーバから直接可視化サーバに転送する機能を備え、また、データの選択や間引きを指定できるため、データ転送は高速に行われる。このため、計算と同時に計算サーバ上でデータ処理を行うプログラムを起動させ、データ転送用のソケットを用意しておかなければならない。計算サーバ上で画像データまで生成し、実時間可視化の高速化を図るシステムも開発されている [6]。ここでは、並列処理、画像圧縮、Java アプレットの利用などが行われており、インターネットからの実時間可視化が可能となっている。しかしながら、これらの可視化システムでは、データ転送用のライブラリの利用や画像データの生成のために、計算プログラム自体を修正する必要があり、またバッチジョブを主体とするスーパーコンピュータシステムにおいては、運用上の制限を受ける場合もある。

リアルタイムモニターシステムでは、複数の種類の計算プログラムの修正、実行を様々な計算機システムにおいて行う場合を想定しており、計算環境の依存性を極力排除し、計算プログラムに変更は加えないことを前提としている。また、東海研及び那珂研の計算機システにおいて、以前からの可視化用の資産を引続き利用することを考慮して汎用可視化ツール AVS の利用を継続している。これまでのリアルタイムモニターシステムでは、描画機能を制御するために、AVS5 の server-cliant 機能のオプションを用い、シェルプログラムで描画用データを渡す方法を用いていた。このため、AVS5 の立ち上げや CLI コマンドの入力など、操作がやや繁雑であった。今回は、描画用データの監視や連続描画などの機能を AVS5 のモジュールとして作成することにより、通常の AVS5 の利用と同様にモニタリングを行えるよう改良し、大幅に操作性を向上させた。また、AVS5 用ばかりでなく、AVS シリーズの最新バージョンである AVS/EXPRESS 用のものも作成し、いずれの可視化ツールにおいてもモニタリング機能を利用できるようにした。また、ある時点までに生成されているシミュレーションのデータを、連続的に描画する機能を付加した。シミュレーションモニタリングシステムは、特殊な画像処理技術や装置を使用せず、汎用可視化ツール AVS5 あるいは AVS/EXPRESS のみを用いているため、様々な計算機環境において利用することが可能である。第 2 章では、可視化ツール AVS5 及び AVS/EXPRESS の概要を述べ、第 3 章では，シミュレーションモニタリングシステムのモジュールについて記述する。第 4 章では，モニタリングシステムの使用例を示し、第 5 章では，モジュールのコンパイル方法について記述する。本報告書では、可視化用のモジュールについてのみ詳述し、計算の制御、可視化データ転送用のシェル、モニターと同時に行うビデオ作成等については記述しない。これらの詳細については、前報 [4] 参照のこと。

2. AVS5 及び AVS/EXPRESS

シミュレーションモニタリングシステムは、画像処理サーバ用として導入整備されている東海研究所のシリコングラフィックス (SGI) 社製 i-station （CPU:R10000 、 メモリ:1GB 、 ハードディスク:110GB 、 OS:IRIX 6.2）において、汎用可視化ツール AVS5 および AVS/EXPRESS を用いて開発している。i-station 以外であっても、AVS5 または AVS/EXPRESS が導入されている計算機システムであればシミュレーションモニタリングシステムは利用可能である。ここでは、それぞれの可視化ツールについて概略を説明する。

2.1 AVS5

可視化ツール AVS （Application Visualization System）は米国 AVS （Advanced Visual Systems）社が開発したコンピュータグラフィックス用ソフトウェアである [7, 8]。AVS はシミュレーション結果や実験データなどの数値データを、直接プログラミングすることなく容易に可視化することができる。ここでは、モジュールと呼ばれる処理単位を組み合わせて必要な画像処理の一連の作業を構成し、そのモジュールの組合せ（ネットワークと呼ばれる）を順次実行することによって、データの読み込み、処理、軸やラベルの設定、色指定、画像ファイルの作成などを行う。また、連続的な描画や自動的な画像処理などを行うために、AVS の実行を CLI （Command Language Interpreter）と呼ばれる機能により制御することができる。

この AVS5 はすでに後継の AVS/EXPRESS に変わり、機能追加などのバージョンアップが行われる予定はないが、ワークステーション上の OS のバージョンアップには対応することになっている。

モジュール、ネットワークの使用例については、JAERI-Tech 95-014[2]、JAERI-Tech 96-044[4] 等を参照のこと。

2.2 AVS/EXPRESS

AVS/EXPRESS は AVS5 と同じく米国 AVS 社が開発したコンピュータグラフィックス用ソフトウェアである [9, 10]。位置付けとしては AVS6 ということになるが、大幅な変更が加えられ、各種可視化がモジュールの組合せだけで可能となっている。また今回のバージョンからパソコン版も加わっている。主な変更点は、まず操作画面と操作方法である。操作画面、操作方法ともパソコン版と同一とし、従来 3 ボタンマウスで行っていた操作も 2 ボタンマウスで全て行えるようになった。次にモジュールの組合せのことを AVS5 ではネットワークと呼んでいたが、AVS/EXPRESS ではアプリケーションと呼ぶようになっている。過去に作成したものをロードする場合もファイルメニューの Load Application で行う。また、AVS5 の機能を拡張し、従来プログラミングでしか作成できなかった機能も、モジュールを細かく分けた「部品」を組み合わせることによりプログラミングすることなく実現できるようになった。

実際に今回開発したモジュールも、AVS5 では 2 つの基本モジュールを組み合わせてマクロ化して Check Step、Anim Step としたが、AVS/EXPRESS ではそれぞれ 1 つのモジュール

で実現できている.

ユーザインターフェース部分も細かくカスタマイズが可能となったのをはじめ、従来のAVS5では不可能であった基本モジュールのカスタマイズもプログラミングすることなく可能となった。部品レベルよりもさらに細かいカスタマイズもCまたはFORTRAN言語を用いてプログラミングすることができる。

さらにAVS5と同様に自動化も可能でAVS/EXPRESSではV言語と呼ばれる専用の言語を用いる。それぞれの詳細は参考文献[9, 10]を参照のこと.

3. シミュレーションモニタリングシステム用モジュール

シミュレーションモニタリング用にデータファイルを読み込む部分のモジュールとしてCheck StepとAnim Stepの2つを開発した。Check Stepモジュールは実時間モニタリング用であり、解析結果ファイルの転送終了を知らせるロックファイルと呼ばれる空のファイルの有無を監視し、存在した場合は解析結果ファイル名を下流のモジュール（可視化データ読み込みモジュール）に出力し、可視化を行うものである。ここで、解析結果ファイル、ロックファイルはそれぞれdata.001、lock.001などのように、拡張子部分を同一のステップ番号（桁は任意）としなければならない。シミュレーション実行中のファイルの転送やシミュレーションの制御などについては、参考文献[4]を参照のこと。Anim Stepモジュールはある時点までに作成されているシミュレーション結果の連続描画を行うものであり、ロックファイル監視を省略してディレクトリ内にある複数の解析結果ファイルを順次読み込み可視化を行うものである。これら2つのモジュールをAVS5用とAVS/EXPRESS用の2種類開発した。

3.1 AVS5用Check Stepモジュール

AVS5用Check Stepモジュールは、基本モジュールであるcheck stepとdirectry browserの2つを組合せてマクロ化することにより操作性を考慮したものである。check step基本モジュールはCheck Stepモジュールの本体をなす部分でロックファイルの監視と解析結果ファイル名の出力を行う。Check Stepモジュールの下流にはこの出力を受け取り、ファイルを読み込むモジュールを接続する。監視ディレクトリの設定はdirectory browser基本モジュールを利用して行う。監視ディレクトリの設定はAnim Stepモジュールでも行う必要があるため、この部分を基本モジュールとして独立させたのがdirectory browserである。Check Stepモジュールで使用しているパラメータについて、パラメータ名、入力（あるいは表示）形態、概要、の順に以下に説明する。

`filter` `string`

検索ディレクトリの直接入力。ディレクトリ検索開始パスを指定する時に使用する。デフォルトはAVS変数であるDataDirectoryの設定値になっている。

`search` `choise`

ディレクトリ検索リスト。filterで指定されたパスのサブディレクトリがリストされる。さらに下の階層、もしくは上の階層に移動する際にはこのリストから選択する。

`select` `choise`

対象ディレクトリ選択リスト。リスト上にはsearchと同様なディレクトリがならんでいる。対象ディレクトリを選択する際にはこのリストから選択する。ここで選択されたディレクトリが解析結果ファイルが格納されるディレクトリと認識する。

`prefix` `string`

解析結果ファイル名の接頭子。本モジュールでは、解析結果ファイル名は prefix.ステップ番号となっていることを想定している。

`first` `integer`

最初のステップ。このステップが解析ステップのうち可視化対象になる最初のステップ番号であると認識する。

`current` `integer`

現在監視対象になっているステップ番号。これはユーザが設定するパラメータではない。

`delta` `integer`

ステップ間引き間隔。解析プログラムがステップを間引いて吐き出しているとき、もしくは可視化対象ステップを間引きたいときにこのパラメータを制御する。通常、ステップは1ずつ増加していくものと想定しているが、このパラメータを制御することで、解析プログラムの出力に合わせた可視化が可能である。また、解析結果が1ずつ増加する場合にも、可視化したいステップ間隔をあけたいときにはこのパラメータを制御する。

`interval` `integer`

ファイル監視時間サイクル。このパラメータが0の時、本モジュールは可能な限り迅速な監視処理を行っている。ただし、この処理は場合によってマシンリソースを大量に消費してしまう場合がある。解析プログラムがステップを吐き出す時間がある程度推測できる場合には、このパラメータを調節することにより、稼働マシンの負担を軽減できる。設定値の単位は秒。

`mode` `choise(undelete|delete)`

ファイル監視モード切替。本モジュールは、解析結果をリアルタイムに可視化するモジュールであるが、ファイル整理を行う機能も有している。deltaによって設定されたステップ間隔で可視化対象にならなかった解析結果ファイルが存在する場合、その解析結果ファイルを削除することができる。デフォルトでは可視化対象にならないファイルもそのまま保存されている。

`run` `boolean`

監視開始ボタン。このボタンをTrue(1)にすることで、監視を開始する。リアルタイム可視化の途中で、あるステップを詳細に検討したい場合、いったんこのパラメータをFalseすることによって、監視を中断することが可能である。再度Trueにすることで監視を再開する。

lock　　string (lock)

監視するロックファイルの接頭子。解析結果ファイルが完全に書き込まれたかを確認するためのロックファイルの接頭子のこと。本モジュールはロックファイル名が、lock.ステップ番号となっていることを想定している。

3.1.1 ソースプログラム

```
#!/usr/avs/bin/avs -network
version 5.4 (50.74 DAO ogl)
#
module "Check Step.user.0" -macro -type 1
module "directory browser.user.1" -parent "Check Step.user.0"
module "check step.user.2" -parent "Check Step.user.0"
#
port_add -out "Check Step.user.0":filename string -flags 0x2
port_vis "check step.user.2":directory -on
port_connect "directory browser.user.1":0 "check step.user.2":0
port_connect "check step.user.2":0 "OUT-> Check Step.user.0":0
#
parm_set "directory browser.user.1":filter $DataDirectory
#
shell "ui" shell
 panel "check step.user.2" -w panel -p ui -xy 400,0 -wh 760,290\
   -P title string "Check Step"
  manipulator "check step.user.2:prefix" -w typein -p "check step.user.2" \
      -xy 510,10 -P width integer 4
  manipulator "check step.user.2:mode" -w radio_buttons -p "check step.user.2" \
      -xy 510,230
  manipulator "check step.user.2:run" -w toggle -p "check step.user.2" \
      -xy 630,230
  manipulator "check step.user.2:lock prefix" -w typein -p "check step.user.2" \
      -xy 510,40 -P width integer  4 -P title string "lock    "
  manipulator "check step.user.2:first step" -w typein_integer -p "check step.user.2" \
      -xy 510,110 -P width integer 4 -P title string "first   "
  manipulator "check step.user.2:current step" -w typein_integer -p "check step.user.2" \
      -xy 510,170 -P width integer 4 -P title string "current "
  manipulator "check step.user.2:delta step" -w typein_integer -p "check step.user.2" \
      -xy 510,140 -P width integer 4 -P title string "delta   "
  manipulator "check step.user.2:interval" -w typein_integer -p "check step.user.2" \
      -xy 510,200 -P width integer 4 -P title string "interval"
  panel "directory browser.user.1" -w panel -p "check step.user.2" \
      -xy 10,10 -wh 496,270
   manipulator "directory browser.user.1:filter" -w typein -p "directory browser.user.1" \
       -xy 10,10 -P width integer 8
   manipulator "directory browser.user.1:search" -w choice_browser -p "directory browser.user.1" \
       -xy 10,40 -wh 236,211
   manipulator "directory browser.user.1:select" -w choice_browser -p "directory browser.user.1" \
       -xy 250,40 -wh 236,211
manipulator "check step.user.2":directory -w none
# End of file
```

3.1.2 AVS5用check step基本モジュール

このモジュールは解析プログラムから送られてくるステップごとの解析結果ファイルを監視し、順次ファイル名を下流モジュールに流すものである。下流モジュールはこのモジュールから渡されるファイル名を受け取りそのファイルを読み込む。

`directory` `string`

監視対象となるディレクトリ。このディレクトリに、解析結果ファイルが書き込まれる。

`prefix` `string`

解析結果ファイル名の接頭子。本モジュールでは、解析結果ファイル名はprefix.ステップ番号となっていることを想定している。

`first` `integer`

最初のステップ。このステップが解析ステップのうち可視化対象になる最初のステップ番号であると認識する。

`current` `integer`

現在監視対象になっているステップ番号。これはユーザが設定するパラメータではない。

`delta` `integer`

ステップ間引き間隔。解析プログラムがステップを間引いて吐き出しているとき、もしくは可視化対象ステップを間引きたいときにこのパラメータを制御する。通常、ステップは1ずつ増加していくものと想定しているが、このパラメータを制御することで、解析プログラムの出力に合わせた可視化が可能である。また、解析結果が1ずつ増加する場合にも、可視化したいステップ間隔をあけたいときにはこのパラメータを制御する。

`interval` `integer`

ファイル監視時間サイクル。このパラメータが0の時、本モジュールは可能な限り迅速な監視処理を行っている。ただし、この処理は場合によってマシンリソースを大量に消費してしまう場合がある。解析プログラムがステップを吐き出す時間がある程度推測できる場合には、このパラメータを調節することにより、稼働マシンの負担を軽減できる。設定値の単位は秒。

`mode` `choise(undelete|delete)`

ファイル監視モード切替。本モジュールは、解析結果をリアルタイムに可視化するモジュールであるが、ファイル整理を行う機能も有している。deltaによって設

定されたステップ間隔で可視化対象にならなかった解析結果ファイルが存在する場合、その解析結果ファイルを削除することができる。デフォルトでは可視化対象にならないファイルもそのまま保存されている。

run　　boolean

監視開始ボタン。このボタンをTrue(1)にすることで、監視を開始する。リアルタイム可視化の途中で、あるステップを詳細に検討したい場合、いったんこのパラメータをFalseすることによって、監視を中断することが可能である。再度Trueにすることで監視を再開する。

lock　　string (lock)

監視するロックファイルの接頭子。解析結果ファイルが完全に書き込まれたかを確認するためのロックファイルの接頭子のこと。本モジュールはロックファイル名が、lock.ステップ番号となっていることを想定している。

3.1.2.1　ソースプログラム

```
/*****************************************************************************/

#include <stdio.h>
#include <sys/types.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>

#include <avs/avs.h>

/*****************************************************************************/

extern int checkstep_descript( void );
static int compute_proc( void );

static int file_comp( char** s1, char** s2 );

/*****************************************************************************/

#define MODULE_NAME "check step"

/*****************************************************************************/

main(
     int ac,
     char* av[]
     )
{
    AVScorout_init( ac, av, checkstep_descript );
```

```
    compute_proc();
}

/*****************************************************************************/

int
checkstep_descript(
                   void
                   )
{
    int port, param;

    AVSset_module_name( MODULE_NAME, MODULE_DATA );

    port = AVScreate_output_port( "filename", "string" );

    param = AVSadd_parameter( "directory", "string", "", "", "" );
    param = AVSadd_parameter( "prefix", "string", "", "", "" );
    param = AVSadd_parameter( "first step", "integer", 0, 0, INT_UNBOUND );
    param = AVSadd_parameter( "current step", "integer", 0, 0, INT_UNBOUND );
    param = AVSadd_parameter( "delta step", "integer", 1, 1, INT_UNBOUND );
    param = AVSadd_parameter( "interval", "integer", 0, 0, INT_UNBOUND );
    param = AVSadd_parameter(
                             "mode", "choice",
                             "undelete", "delete!undelete", "!"
                             );
    param = AVSadd_parameter( "run", "boolean", 0, 0, 0 );
    param = AVSadd_parameter( "lock prefix", "string", "lock", "", "" );

    return (1);
}

/*****************************************************************************/

static int
compute_proc(
             void
             )
{
    char *directory = NULL, *prefix = NULL, *lock = NULL, *mode = NULL;
    int first, current, delta, interval, run;
    char filename[256];
    char fullpath[256];
    char **files;
    DIR *dp;
    FILE *fp;
    struct dirent *dent;
    int nfiles, n, i, j, no;
    char *sp;

    run = 0;
```

```
while ( 1 )
{
    while ( 0 == run )
    {
        AVScorout_wait();
        AVScorout_input(
                        &directory, &prefix,
                        &first, &current, &delta,
                        &interval, &mode, &run, &lock
                        );
    }

    AVScorout_input(
                    &directory, &prefix,
                    &first, &current, &delta,
                    &interval, &mode, &run, &lock
                    );

    if ( NULL != (dp = opendir( directory )) )
    {
        nfiles = 0;
        n = strlen( lock );
        while ( NULL != (dent = readdir( dp )) )
        {
            if ( 0 == strncmp( lock, dent->d_name,  n ) )
                nfiles++;
        }
        rewinddir( dp );

        if ( NULL != (files = malloc( nfiles * sizeof (char*) )) )
        {
            nfiles = 0;
            while ( NULL != (dent = readdir( dp )) )
            {
                if ( 0 == strncmp( lock, dent->d_name,  n ) )
                {
                    files[nfiles] = strdup( dent->d_name );
                    nfiles++;
                }
            }
        }

        else if ( 0 < nfiles )
        {
            AVSfatal( "memory fatal error" );
            nfiles = 0;
        }

        closedir( dp );

        qsort( files, nfiles, sizeof (char*), file_comp );
```

```
if ( current < first )
{
    current = first;
    AVSmodify_parameter(
                        "current step", AVS_VALUE,
                        current, 0, 0
                        );
}

for ( i = 0; i < nfiles; i++ )
{
    if ( NULL != (sp = strrchr( files[i], '.' )) )
    {
        no = atoi( sp + 1 );

        if ( 0 == strcmp( "delete", mode ) && no < current )
        {
            if ( 0 != (no - first) % delta )
            {
                sprintf( fullpath, "%s/%s", directory, files[i] );
                if ( NULL != (fp = fopen( fullpath, "r" )) )
                {
                    fclose( fp );
                    unlink( fullpath );
                }

                sprintf(
                        fullpath, "%s/%s%s",
                        directory, prefix, sp
                        );
                if ( NULL != (fp = fopen( fullpath, "r" )) )
                {
                    fclose( fp );
                    unlink( fullpath );
                }
            }
        }

        if ( no == current )
        {
            sprintf( filename, "%s/%s%s", directory, prefix, sp );
            AVScorout_output( filename );
            AVScorout_exec();

            current += delta;
            AVSmodify_parameter(
                                "current step", AVS_VALUE,
                                current, 0, 0
                                );

            for ( j = i; j < nfiles; j++ )
```

```
                        free( files[j] );

                        break;
                    }
                }

                free( files[i] );
            }

            if ( NULL != files )
                free( files );
        }

        else
        {
            AVSerror( "can not open directory(%s)", directory );

            run = 0;
            AVSmodify_parameter( "run", AVS_VALUE, 0, 0, 0 );
        }

        sleep( interval );
    }

    return (1);
}

/*****************************************************************************/

static int
file_comp(
          char** s1,
          char** s2
          )
{
    return (strcmp( *s1, *s2 ));
}

/*****************************************************************************/
```

3.1.3 AVS5用directory browser基本モジュール

本モジュールはディレクトリをリストから選択できるようにするためのユーティリティモジュールである。モニタリングを行うには解析結果ファイルが格納されるディレクトリを指定する必要がある。ディレクトリの階層が深い場合にはディレクトリ名をフルパスで直接入力することは、ユーザの負担であるばかりでなく入力ミスのもとにもなる。AVS5にはこのようなディレクトリを選択できるようなブラウザウィジェットが実装されていないため開発したモジュールである。

filter　　string

検索ディレクトリの直接入力．ディレクトリの検索開始パスを指定する．

search　　choice

ディレクトリ検索リスト．filterで指定されたパスのサブディレクトリがリストされる．さらに下の階層もしくは上の階層へ移動する際にはこのリストから選択する．

select　　choice

対象ディレクトリ選択リスト．リスト上には，searchと同様なディレクトリがならんでいる．対象ディレクトリを選択する際にはこのリストから選択する．

3.1.3.1　ソースプログラム

```
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <string.h>

#include <avs/avs.h>

/*****************************************************************************/

typedef int (*CMP_FUNC)(const void*, const void*);

/*****************************************************************************/

extern int dirbrowser_descript( void );
static int compute_proc();
static int filter_proc( char* );
static int search_proc( char*, char* );
static int select_proc( char*, char*, char* );
static char* pathname_parse( char*, char* );

/*****************************************************************************/

#define MODULE_NAME "directory browser"

/*****************************************************************************/

int
AVSinit_modules(
                void
```

```
            )
{
    AVSmodule_from_desc( dirbrowser_descript );

    return (1);
}

/*****************************************************************************/

int
dirbrowser_descript(
                    void
                    )
{
    int port;
    int param;

    AVSset_module_name( MODULE_NAME, MODULE_DATA );

    port = AVScreate_output_port( "folder", "string" );

    param = AVSadd_parameter( "filter", "string", ".", "", "" );

    param = AVSadd_parameter( "search", "choice", "", "", "!" );
    AVSconnect_widget( param, "choice_browser" );

    param = AVSadd_parameter( "select", "choice", "", "", "!" );
    AVSconnect_widget( param, "choice_browser" );

    AVSset_compute_proc( compute_proc );

    return (1);
}

/*****************************************************************************/

static
compute_proc(
             char** folder,
             char* filter,
             char* search,
             char* select
             )
{
    int status = 0;
    char buffer[256];

    if ( AVSparameter_changed( "filter" ) )
        filter_proc( filter );

    if ( AVSparameter_changed( "search" ) )
```

```
        search_proc( filter, search );

    if ( AVSparameter_changed( "select" ) )
    {
        status = select_proc( filter, select, buffer );

        if ( 1 == status )
        {
            if ( 0 != *folder )
                free( *folder );

            *folder = strdup( buffer );
        }
    }

    return (status);
}

/*****************************************************************************/

static int
filter_proc(
            char* filter
            )
{
    int status = 0;
    char dir[256];
    DIR *dp;
    struct dirent *dinfo;
    struct stat stat_buff;
    char work[256];
    char *buffer, *buffer2;
    size_t nbyte;
    char (*dirs)[256];
    int ndirs, i;

    pathname_parse( filter, dir );
    AVSmodify_parameter( "filter", AVS_VALUE, dir, "", "" );

    dp = opendir( dir );
    if ( 0 != dp )
    {
        nbyte = 0;
        ndirs = 0;
        for ( dinfo = readdir( dp ); 0 != dinfo; dinfo = readdir( dp ) )
        {
            sprintf( work, "%s/%s", dir, dinfo->d_name );
            status = stat( work, &stat_buff );
            if ( 0 == status && S_ISDIR( stat_buff.st_mode ) )
            {
                nbyte += strlen( dinfo->d_name ) + 1;
```

```
                ndirs++;
            }
        }

#ifdef D_DEBUG_D
        printf( "nbyte(%d) ndirs(%d)\n", nbyte, ndirs ); fflush( 0 );
#endif /* D_DEBUG_D */

        buffer = malloc( nbyte + 1 );
        buffer2 = malloc( nbyte + 1 );
        dirs = calloc( ndirs, sizeof (char[256]) );

        if ( 0 != buffer && 0 != buffer2 && 0 != dirs )
        {
            rewinddir( dp );

            i = 0;
            for ( dinfo = readdir( dp ); 0 != dinfo; dinfo = readdir( dp ) )
            {
                sprintf( work, "%s/%s", dir, dinfo->d_name );
                status = stat( work, &stat_buff );
                if ( 0 == status && S_ISDIR( stat_buff.st_mode ) )
                {
                    strcpy( dirs[i], dinfo->d_name );
                    i++;
#ifdef D_DEBUG_D
                    printf( "(%s:%d) i(%d)\n", __FILE__, __LINE__, i );
                    fflush( 0 );
#endif /* D_DEBUG_D */
                }
            }

            qsort(
                  dirs, ndirs, sizeof (char[256]),
                  (CMP_FUNC)strcmp
                  );

            buffer[0] = '\0';
            buffer2[0] = '\0';
            for ( i = 0; i < ndirs; i++ )
            {
                strcat( buffer, dirs[i] );
                strcat( buffer, "!" );

                if (
                    strcmp( dirs[i], "." ) &&
                    strcmp( dirs[i], ".." )
                    )
                {
                    strcat( buffer2, dirs[i] );
                    strcat( buffer2, "!" );
                }
```

```
            }

            buffer[strlen(buffer) - 1] = '\0';
            buffer2[strlen(buffer2) - 1] = '\0';

            AVSmodify_parameter( "search", AVS_MINVAL, 0, buffer, 0 );
            AVSmodify_parameter( "select", AVS_MINVAL, 0, buffer2, 0 );
            free( buffer );
            free( buffer2 );
            free( dirs );
        }

        else
        {
            fprintf( stderr, "can not alloc memory\n" );
        }

        closedir( dp );
    }

    else
    {
        fprintf( stderr, "can not open directory\n" );
    }

    return (status);
}

/*****************************************************************************/

static int
search_proc(
            char* filter,
            char* search
            )
{
    int status = 0;
    char buffer[256];

    sprintf( buffer, "%s/%s", filter, search );

    status = filter_proc( buffer );

    return (status);
}

/*****************************************************************************/

static int
select_proc(
            char* filter,
```

```
            char* select,
            char* folder
            )
{

    int status = 1;

    sprintf( folder, "%s/%s", filter, select );

    return (status);
}

/*****************************************************************************/

static char*
pathname_parse(
               char* pathname,
               char* buffer
               )
{
    char *sp, *dp;

    sp = pathname;
    dp = buffer;

    while ( '\0' != *sp )
    {
        switch ( *sp )
        {
          case '/':
            switch ( *(sp + 1) )
            {
              case '/':
                sp++;

                break;

              case '.':
                switch ( *(sp + 2) )
                {
                  case '\0':
                  case '/':
                    sp += 2;

                    break;

                  case '.':
                    switch ( *(sp + 3) )
                    {
                      case '\0':
                      case '/':
                        *dp = '\0';
```

```
                    dp = strrchr( buffer, '/' );
                    if ( '\0' == dp )
                        dp = buffer;

                    sp += 3;

                    break;

                  default:
                    *dp++ = *sp++;
                }

                break;

              default:
                *dp++ = *sp++;
            }

            break;

          default:
            *dp++ = *sp++;
        }

        break;

      default:
        *dp++ = *sp++;
    }
  }

  *dp = '\0';
  if ( 0 == strlen( buffer ) )
      strcpy( buffer, "/" );

  return (buffer);
}

/*****************************************************************************/
```

3.2 AVS5用Anim Stepモジュール

AVS5用Anim Stepモジュールは基本モジュールであるanim stepとdirectory browserを組み合わせてマクロ化することにより、操作性の向上を考慮したものである。anim step基本モジュールは、シミュレーション実行中のある時点（あるいは終了時）までに生成された複数の解析結果ファイルを順次読み込み描画するものである。監視ディレクトリの設定はdirectory browser基本モジュールを利用して行い、読み込むファイル名を出力する。

```
filter          string
```

検索ディレクトリの直接入力。ディレクトリ検索開始パスを指定する時に使用する。デフォルトはAVS変数である。DataDirectoryの設定値になっている。

`search` `choise`

ディレクトリ検索リスト。filterで指定されたパスのサブディレクトリがリストされる。さらに下の階層もしくは、上の階層に移動する際にはこのリストから選択する。

`select` `choise`

対象ディレクトリ選択リスト。リスト上にはsearchと同様なディレクトリがならんでいる。対象ディレクトリを選択する際にはこのリストから選択する。ここで選択されたディレクトリが解析結果ファイルが格納されるディレクトリと認識する。

`prefix` `string`

解析結果ファイル名の接頭子。本モジュールでは、解析結果ファイル名はprefix.ステップ番号となっていることを想定している。

`first` `integer`

最初のステップ。このステップが解析ステップのうち可視化対象になる最初のステップ番号であると認識する。

`last` `integer`

可視化最終ステップ。連続可視化の終了ステップを指定する。

`delta` `integer`

ステップ間引き間隔。解析プログラムがステップを間引いて吐き出しているとき、もしくは可視化対象ステップを間引きたいときにこのパラメータを制御する。通常、ステップは1ずつ増加していくものと想定しているが、このパラメータを制御することで、解析プログラムの出力に合わせた可視化が可能である。また、解析結果が1ずつ増加する場合にも、可視化したいステップ間隔をあけたいときにはこのパラメータを制御する。

`mode` `choise(undelete|delete)`

ファイル監視モード切替。本モジュールは、解析結果をリアルタイムに可視化するモジュールであるが、ファイル整理を行う機能も有している。deltaによって設定されたステップ間隔で可視化対象にならなかった解析結果ファイルが存在する場合、その解析結果ファイルを削除することができる。デフォルトでは可視化対象にならないファイルもそのまま保存されている。

run boolean

監視開始ボタン。このボタンをTrue(1)にすることで、監視を開始する。リアルタイム可視化の途中で、あるステップを詳細に検討したい場合、いったんこのパラメータをFalseすることによって、監視を中断することが可能である。再度Trueにすることで監視を再開する。

3.2.1 ソースプログラム

```
#!/usr/avs/bin/avs -network
version 5.4 (50.74 DAO ogl)
#
module "Anim Step.user.0" -macro -type 1
module "directory browser.user.1" -parent "Anim Step.user.0"
module "anim step.user.2" -parent "Anim Step.user.0"
#
port_add -out "Anim Step.user.0":filename string -flags 0x2
port_vis "anim step.user.2":directory -on
port_connect "directory browser.user.1":0 "anim step.user.2":0
port_connect "anim step.user.2":0 "OUT-> Anim Step.user.0":0
#
parm_set "directory browser.user.1":filter $DataDirectory
#
shell "ui" shell
 panel "anim step.user.2" -w panel -p ui -xy 400,0 -wh 760,290\
   -P title string "Anim Step"
  manipulator "anim step.user.2:prefix" -w typein -p "anim step.user.2" \
      -xy 510,10 -P width integer 4
  manipulator "anim step.user.2:mode" -w radio_buttons -p "anim step.user.2" \
      -xy 510,230
  manipulator "anim step.user.2:run" -w toggle -p "anim step.user.2" \
      -xy 630,230
  manipulator "anim step.user.2:first step" -w typein_integer -p "anim step.user.2" \
      -xy 510,90 -P width integer 4 -P title string "first"
  manipulator "anim step.user.2:last step" -w typein_integer -p "anim step.user.2" \
      -xy 510,120 -P width integer 4 -P title string "last"
  manipulator "anim step.user.2:current step" -w typein_integer -p "anim step.user.2" \
      -xy 510,180 -P width integer 4 -P title string "current"
  manipulator "anim step.user.2:delta step" -w typein_integer -p "anim step.user.2" \
      -xy 510,150 -P width integer 4 -P title string "delta"
  panel "directory browser.user.1" -w panel -p "anim step.user.2" \
      -xy 10,10 -wh 496,270
   manipulator "directory browser.user.1:filter" -w typein -p "directory browser.user.1" \
       -xy 10,10 -P width integer 8
   manipulator "directory browser.user.1:search" -w choice_browser -p "directory browser.user.1" \
       -xy 10,40 -wh 236,211
   manipulator "directory browser.user.1:select" -w choice_browser -p "directory browser.user.1" \
       -xy 250,40 -wh 236,211
manipulator "anim step.user.2":directory -w none
# End of file
```

3.2.2 AVS5用anim step基本モジュール

check stepモジュールのロック監視機能を省略し、解析結果ファイルが格納されているディレクトリ内のファイル名を連続的に下流モジュールに流すものである。下流モジュールはこのモジュールから渡されるファイル名を受け取りそのファイルを読み込む。

directory **string**

解析結果ファイルが格納されたディレクトリ。

prefix **string**

解析結果ファイル名の接頭子。本モジュールでは、解析結果ファイル名はprefix.ステップ番号となっていることを想定している。

first **integer**

可視化開始ステップ。連続可視化を行なう先頭ステップを指定する。

last **integer**

可視化終了ステップ。可視化最終ステップを指定する。

current **integer**

現在監視対象になっているステップ番号。これはユーザが設定するパラメータではない。

delta **integer**

ステップ間引き間隔。途中のステップを飛ばしたい時にこのパラメータを制御する.

mode **choise(undelete|delete)**

アニメーションモード。連続可視化をfirstステップから、lastステップまで行なった後終了するか、firstステップから再開するかを指定する。このパラメータがloopのとき、繰り返し連続可視化を実行し続ける。

run **boolean**

連続可視化開始実行ボタン。

3.2.2.1 ソースプログラム

```
/*****************************************************************************/

#include <stdio.h>
#include <sys/types.h>
```

```
#include <string.h>
#include <dirent.h>
#include <unistd.h>

#include <avs/avs.h>

/*****************************************************************************/

extern int animstep_descript( void );
static int compute_proc( void );
static int file_comp( char** s1, char** s2 );

/*****************************************************************************/

#define MODULE_NAME "anim step"

/*****************************************************************************/

main(
     int ac,
     char* av[]
     )
{
    AVScorout_init( ac, av, animstep_descript );

    compute_proc();
}

/*****************************************************************************/

int
animstep_descript(
                  void
                  )
{
    int port, param;

    AVSset_module_name( MODULE_NAME, MODULE_DATA );

    port = AVScreate_output_port( "filename", "string" );

    param = AVSadd_parameter( "directory", "string", "", "", "" );
    param = AVSadd_parameter( "prefix", "string", "", "", "" );
    param = AVSadd_parameter( "first step", "integer", 0, 0, INT_UNBOUND );
    param = AVSadd_parameter( "last step", "integer", 0, 0, INT_UNBOUND );
    param = AVSadd_parameter( "current step", "integer", 0, 0, INT_UNBOUND );
    param = AVSadd_parameter( "delta step", "integer", 1, 1, INT_UNBOUND );
    param = AVSadd_parameter( "mode", "choice", "one", "one!loop", "!" );
    param = AVSadd_parameter( "run", "boolean", 0, 0, 0 );

    return (1);
}
```

```
/*****************************************************************************/

static int
compute_proc(
            void
            )
{
    char *directory, *prefix, *mode;
    int first, last, current, delta, run;
    char filename[256];
    char **files = NULL;
    DIR *dp;
    FILE *fp;
    struct dirent *dent;
    int nfiles, n, i, no;

    nfiles = 0;
    files = NULL;
    while ( 1 )
    {
        AVScorout_input(
                        &directory, &prefix,
                        &first, &last, &current, &delta,
                        &mode, &run
                        );

        while ( 0 == run )
        {
            AVScorout_wait();
            AVScorout_input(
                            &directory, &prefix,
                            &first, &last, &current, &delta,
                            &mode, &run
                            );

            if (
                1 == run &&
                NULL != directory && NULL != (dp = opendir( directory ))
                )
            {
                if ( NULL != files )
                {
                    for ( i = 0; i < nfiles; i++ )
                        free( files[i] );

                    free( files );
                    files = NULL;
                }

                nfiles = 0;
                n = strlen( prefix );
```

```
    while ( NULL != (dent = readdir( dp )) )
    {
        if ( 0 == strncmp( prefix, dent->d_name,  n ) )
            nfiles++;
    }

    rewinddir( dp );

    if (
        NULL != (files = malloc( nfiles * sizeof (char*) )) )
    {
        nfiles = 0;
        while ( NULL != (dent = readdir( dp )) )
        {
            if ( 0 == strncmp( prefix, dent->d_name,  n ) )
            {
                files[nfiles] = strdup( dent->d_name );
                nfiles++;
            }
        }
    }

    else
    {
        AVSfatal( "memory fatal error" );
        nfiles = 0;
    }

    closedir( dp );

    if ( 1 < nfiles )
        qsort( files, nfiles, sizeof (char*), file_comp );

    if ( last >= nfiles )
    {
        last = nfiles - 1;
        AVSmodify_parameter( "last step", AVS_VALUE, last, 0, 0 );
    }

    AVSmodify_parameter(
                        "first step",
                        AVS_MAXVAL, 0, 0, nfiles - 1
                        );

    AVSmodify_parameter(
                        "last step",
                        AVS_MAXVAL, 0, 0, nfiles - 1
                        );

    break;
}
```

```
            else
            {
                run = 0;
                AVSmodify_parameter( "run", AVS_VALUE, 0, 0, 0 );
            }
        }

        if ( current < first )
        {
            current = first;
            AVSmodify_parameter( "current step", AVS_VALUE, current, 0, 0 );
        }

        if ( current < last && current < nfiles )
        {
            sprintf( filename, "%s/%s", directory, files[current] );
            AVScorout_output( filename );
            AVScorout_exec();

            current += delta;
            AVSmodify_parameter( "current step", AVS_VALUE, current, 0, 0 );
        }

        else
        {
            current = first;
            AVSmodify_parameter( "current step", AVS_VALUE, current, 0, 0 );

            if ( 0 == strcmp( "one", mode ) )
            {
                run = 0;
                AVSmodify_parameter( "run", AVS_VALUE, run, 0, 0 );
            }
        }
    }

    return (1);
}

/*****************************************************************************/

static int
file_comp(
          char** s1,
          char** s2
          )
{
    return (strcmp( *s1, *s2 ));
}

/*****************************************************************************/
```

3.3 AVS/EXPRESS 用 Check Step モジュール

このモジュールも AVS5 用と同様、解析プログラムから送られてくるステップごとの解析結果ファイルを監視し、順次ファイル名を下流モジュールに流すものである。下流モジュールはこのモジュールから渡されるファイル名を受け取りそのファイルを読み込む。また AVS5 用 Check step は監視間隔時間 (interval) 設定をモジュール内で行っているが、AVS/EXPRESS 用 Check Step では AVS/EXPRESS に用意されている、一定間隔でモジュールを呼び出す機能を利用している。

`directory`

監視対象となるディレクトリ。このディレクトリに、解析結果ファイルが書き込まれる。

`prefix`

解析結果ファイル名の接頭子。本モジュールでは、解析結果ファイル名は prefix.ステップ番号となっていることを想定している。

`lock`

監視するロックファイルの接頭子。解析結果ファイルが完全に書き込まれたかを確認するためのロックファイルの接頭子。本モジュールはロックファイル名が lock.ステップ番号となっていることを想定している。

`first`

最初のステップ。このステップが解析ステップのうち可視化対象になる最初のステップ番号であると認識する。

`current`

現在監視対象になっているステップ番号。これはユーザが設定するパラメータではない。

`delta`

ステップ間引き間隔。解析プログラムがステップを間引いて吐き出しているとき、もしくは可視化対象ステップを間引きたいときにこのパラメータを制御する。通常、ステップは 1 ずつ増加していくものと想定しているが、このパラメータを制御することで、解析プログラムの出力に合わせた可視化が可能である。また、解析結果が 1 ずつ増加する場合にも、可視化したいステップ間隔をあけたいときにはこのパラメータを制御する。

`interval`

ファイル監視時間サイクル。このパラメータが0の時、本モジュールは可能な限り迅速な監視処理を行っている。ただし、この処理は場合によってマシンリソースを大量に消費してしまう場合がある。解析プログラムがステップを吐き出す時間がある程度推測できる場合には、このパラメータを調節することにより、稼働マシンの負担を軽減できる。設定値の単位は秒。

undelete|delete

ファイル監視モード切替。本モジュールは、解析結果をリアルタイムに可視化するモジュールであるが、ファイル整理を行う機能も有している。deltaによって設定されたステップ間隔で可視化対象にならなかった解析結果ファイルが存在する場合、その解析結果ファイルを削除することができる。デフォルトでは可視化対象にならないファイルもそのまま保存されている。

run

監視開始ボタン。このボタンをTrue(1)にすることで、監視を開始す る。リアルタイム可視化の途中で、あるステップを詳細に検討したい場合、いったんこのパラメータをFalseすることによって、監視を中断することが可能である。再度Trueにすることで監 視を再開する。

3.3.1 ソースプログラム

```
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>

#include "express.h"

/*****************************************************************************/

static int file_comp( char** s1, char** s2 );

/*****************************************************************************/

int
checkstep_update(
                OMobj_id top_id,
                OMevent_mask ev,
                int sq
                )
{
```

```
char *dir = NULL, *prefix = NULL, *lock = NULL;
int first, current, previous, delta, mode;
char filename[256];
char fullpath[256];
char **files;
DIR *dp;
FILE *fp;
struct dirent *dent;
int nfiles, n, i, no;
char *sp;

OMget_name_str_val( top_id, OMstr_to_name( "directory" ), &dir, 0 );
OMget_name_str_val( top_id, OMstr_to_name( "prefix" ), &prefix, 0 );
OMget_name_str_val( top_id, OMstr_to_name( "lock_prefix" ), &lock, 0 );
OMget_name_int_val( top_id, OMstr_to_name( "current_step" ), &current );
OMget_name_int_val( top_id, OMstr_to_name( "first_step" ), &first );
OMget_name_int_val( top_id, OMstr_to_name( "delta_step" ), &delta );
OMget_name_int_val( top_id, OMstr_to_name( "delete_mode" ), &mode );

if ( NULL != (dp = opendir( dir )) )
{
    nfiles = 0;
    n = strlen( lock );
    while ( NULL != (dent = readdir( dp )) )
    {
        if ( 0 == strncmp( lock, dent->d_name, n ) )
            nfiles++;
    }
    rewinddir( dp );

    if ( NULL != (files = malloc( nfiles * sizeof (char*) )) )
    {
        nfiles = 0;
        while ( NULL != (dent = readdir( dp )) )
        {
            if ( 0 == strncmp( lock, dent->d_name, n ) )
            {
                files[nfiles] = strdup( dent->d_name );
                nfiles++;
            }
        }
    }

    else if ( 0 < nfiles )
    {
        /* error */
    }

    closedir( dp );
}

if ( 0 < nfiles )
```

```
{
    qsort( files, nfiles, sizeof (char*), file_comp );

    if ( current < first )
    {
        current = first;
        OMset_name_int_val(
                        top_id, OMstr_to_name( "current_step" ),
                        current
                        );
    }

    previous = current - delta;
    previous = (first > previous ? first : previous);

    for ( i = 0; i < nfiles; i++ )
    {
        if ( NULL != files[i] )
        {
            if ( NULL != (sp = strrchr( files[i], '.' )) )
            {
                no = atoi( sp + 1 );

                if ( no == current )
                {
                    sprintf( filename, "%s/%s%s", dir, prefix, sp );

                    if ( NULL != (fp = fopen( filename, "r" )) )
                    {
                        fclose( fp );
                        OMset_name_str_val(
                                        top_id,
                                        OMstr_to_name( "filename" ),
                                        filename
                                        );
                        current += delta;
                        OMset_name_int_val(
                                        top_id,
                                        OMstr_to_name( "current_step" ),
                                        current
                                        );

                        break;
                    }
                }

                else if ( no < current )
                {
                    if (
                        1 == mode && no > first &&
                        (no - previous) % delta
```

```
                        )
                    {
                        sprintf( filename, "%s/%s%s", dir, prefix, sp );
                        if ( NULL != (fp = fopen( filename, "rw" )) )
                        {
                            fclose( fp );
                            unlink( filename );
                        }

                        sprintf( filename, "%s/%s%s", dir, lock, sp );
                        unlink( filename );
                    }
                }

                else
                    break;
            }
        }
    }

    for ( i = 0; i < nfiles; i++ )
        if ( NULL != files[i] )
            free( files[i] );
    free( files );
  }

  return (OM_STAT_SUCCESS);
}

/*****************************************************************************/

static int
file_comp(
          char** s1,
          char** s2
          )
{
  return (strcmp( *s1, *s2 ));
}

/*****************************************************************************/
```

3.4 AVS/EXPRESS 用 Anim Step モジュール

AVS5 用と同様に check step モジュールのロック監視機能を省略し、解析結果ファイルが格納されているディレクトリ内のファイル名を連続的に下流モジュールに流すものである。下流モジュールはこのモジュールから渡されるファイル名を受け取りそのファイルを読み込む。

```
directory       string
```

解析結果ファイルが格納されたディレクトリ.

prefix　　string

解析結果ファイル名の接頭子。本モジュールでは、解析結果ファイル名は prefix. ステップ番号となっていることを想定している。

first　　integer

可視化開始ステップ. 連続可視化を行なう先頭ステップを指定する

last　　integer

可視化終了ステップ. 可視化最終ステップを指定する.

current　　integer

現在監視対象になっているステップ番号。これはユーザが設定するパラメータではない。

delta　　integer

ステップ間引き間隔。途中のステップを飛ばしたい時にこのパラメータを制御する.

mode　　choise(undelete|delete)

アニメーションモード. 連続可視化を first ステップから, last ステップまで行なった後終了するか, first ステップから再開するかを指定する. このパラメータが lo op のとき, 繰り返し連続可視化を実行し続ける.

run　　boolean

連続可視化開始実行ボタン.

3.4.1 ソースプログラム

```
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>

#include "express.h"

/*****************************************************************************/
```

```
static int file_comp( char** s1, char** s2 );

/*****************************************************************************/

int
animsetup_update(
                 OMobj_id top_id,
                 OMevent_mask ev,
                 int sq
                )
{
    OMobj_id files_id;
    char *dir = NULL, *prefix = NULL, **files;
    int nfiles;
    char filename[256];
    DIR *dp;
    struct dirent *dent;
    FILE *fp;
    int i, n;

    OMget_name_str_val( top_id, OMstr_to_name( "directory" ), &dir, 0 );
    OMget_name_str_val( top_id, OMstr_to_name( "prefix" ), &prefix, 0 );

    if ( NULL != (dp = opendir( dir )) )
    {
        nfiles = 0;
        n = strlen( prefix );

        while ( NULL != (dent = readdir( dp )) )
        {

            if ( 0 == strncmp( prefix, dent->d_name, n ) )
                nfiles++;
        }

        rewinddir( dp );

        if ( NULL != (files = malloc( nfiles * sizeof (char*) )) )
        {
            nfiles = 0;
            while ( NULL != (dent = readdir( dp )) )
            {
                if ( 0 == strncmp( prefix, dent->d_name, n ) )
                {
                    files[nfiles] = strdup( dent->d_name );
                    nfiles++;
                }
            }
        }

        closedir( dp );
```

```
        if ( NULL != files )
        {
            files_id = OMfind_subobj(
                                top_id, OMstr_to_name( "files" ),
                                OM_OBJ_RW
                                );
            OMset_array_size( files_id, nfiles );

            for ( i = 0; i < nfiles; i++ )
            {
                sprintf( filename, "%s/%s", dir, files[i] );
                OMset_str_array_val( files_id, i, filename );
            }
        }
    }

    return (OM_STAT_SUCCESS);
}

/*****************************************************************************/
```

4. モニタリング例

4.1 AVS5 によるモニタリング

4.1.1 Check Step モジュール

Fig.4.1に AVS5 用 Check Step モジュールのコントロールパネルを示す。

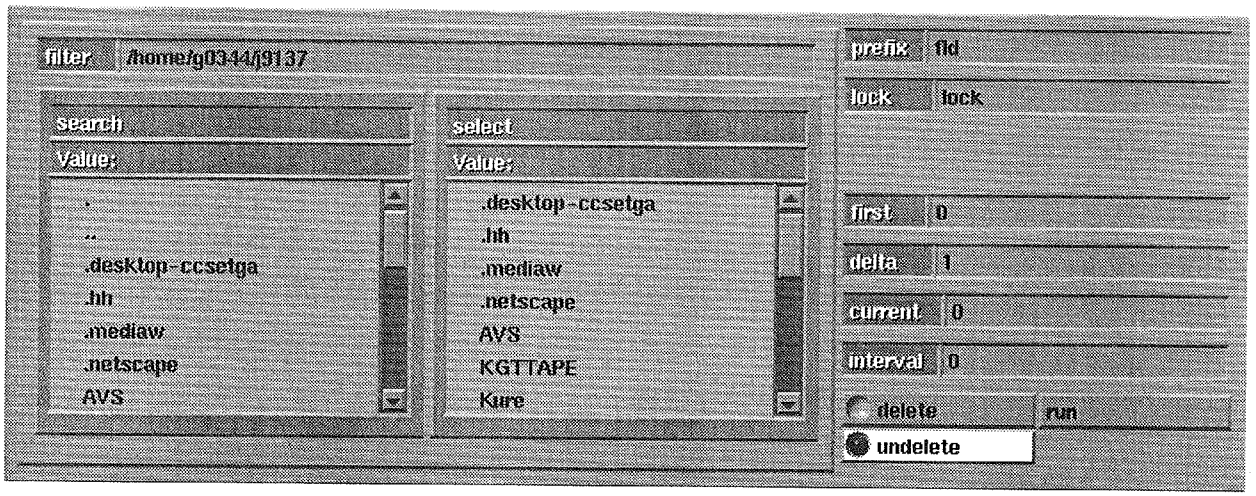

Fig. 4.1 AVS5 用 Check Step コントロールパネル

通常の描画を行うネットワークの read field モジュールに Check Step モジュールを接続し、Fig.4.1のコントロールパネルを表示させる。次に search、select を用いてロックファイル、解析結果ファイルの格納されるディレクトリを設定し、prefix には解析結果ファイルの接頭子、lock にはロックファイルの接頭子、first にはモニタ開始ステップ番号、current には入力なし、delta にはステップ間引き間隔、interval にはファイル監視時間サイクルをそれぞれ設定する。描画後に解析結果ファイルを削除する場合には delete を選択し、run を on することによってモニタが開始される。

4.1.2 Anim Step モジュール

Fig.4.2に AVS5 用 Anim Step モジュールのコントロールパネルを示す。

Check Step マクロモジュールと同様に、通常の描画を行うネットワークの read field モジュールに Anim Step モジュールを接続し、Fig.4.2のコントロールパネルを表示させる。search、select を用いて解析結果ファイルの格納されるディレクトリを設定し、prefix には解析結果ファイルの接頭子、first には可視化開始ステップ番号、last には可視化終了ステップ番号、delta にはステップ間引き間隔をそれぞれ設定する。繰り返し連続可視化を行なう時には loop を on、1回で終了する場合は one を on とし、run を on することで開始される。

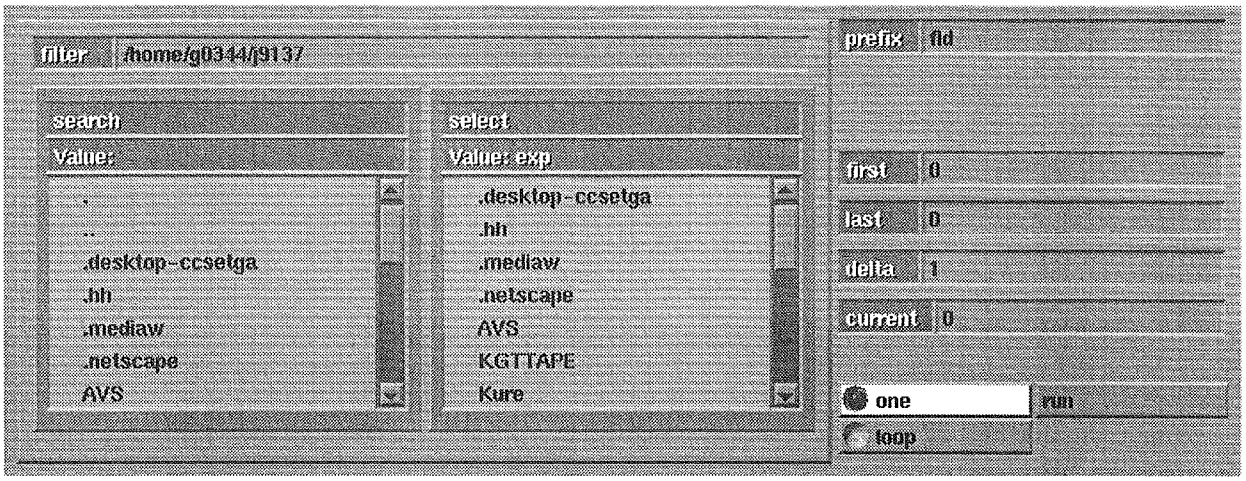

Fig. 4.2　AVS5 用 Anim Step コントロールパネル

4.1.3　表示例

AVS5 によるモニタリングの例を以下に示す。これは 3 次元空間での等値面と断面内分布を描画する例で、使用したネットワークは Fig.4.3に示すものである。Check Step モジュールにより実時間モニタリングを行うものである。

Fig.4.3のネットワークで使用しているモジュールの機能の概略は以下の通りである。

- Check Step （ロックファイルを監視し、ファイル名を出力する）

  ロックファイルを監視し、検出した際に対応するファイル名を下流のジュールに渡す。

- read field （フィールド・フォーマットのファイルを読み込み、フィールドデータを出力する）

  フィールドファイルを読み込み、フィールドデータを出力する。ファイル名の入力は Check Step モジュールから行う。

- crop （任意の範囲のデータを抽出する）

  読み込んだデータのうち、描画に用いるデータを抽出する。このモジュールがなければ読み込んだデータ全てを描画するが、接続することによってデータの範囲を指定し、一部分だけの描画も可能である。

- extract scalar （複数成分を持つコンポーネント・データの中から 1 つのサブコンポーネントを抽出する）

  1 つのフィールドデータファイルの中に複数種類のデータ（速度、温度、圧力など）が存在するときにその中の 1 つを抽出する。

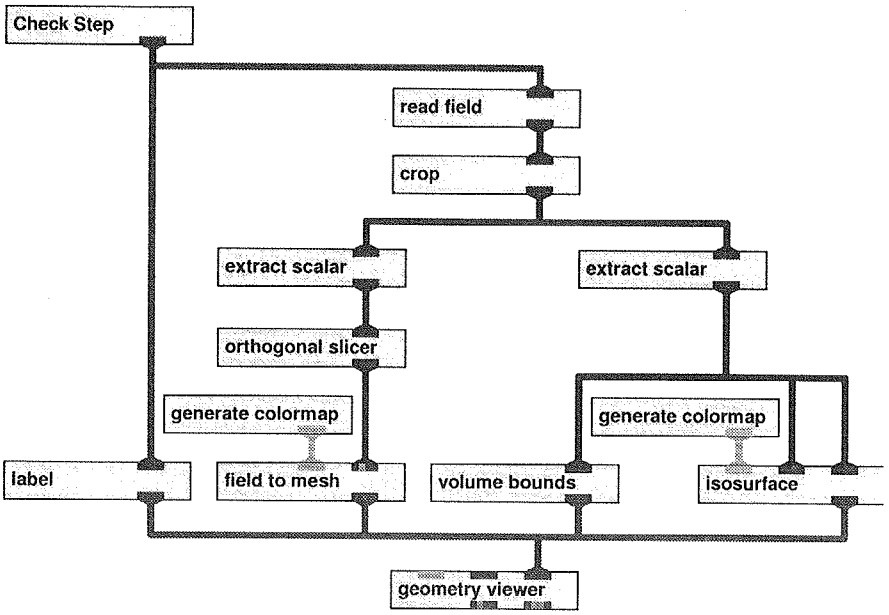

Fig. 4.3　AVS5 用ネットワーク

- orthogonal slicer （座標軸に垂直な平面でボリュームデータを切断する）

  3次元データの中から指定した部分の断面で切断する。

- field to mesh （2次元のスカラーフィールドを3次元の面に変換する）

  2次元のデータを geometory viewer モジュールで描画可能な3次元の面に変換する。

- generate colormap （ＡＶＳカラーマップを出力する）

  カラー表示のためのカラーマップデータ構造を生成し出力する。

- volume bounds （3次元の構造格子に境界枠を生成する）

  読み込んだデータの大きさの枠を描画する。

- isosurface （等値面を作成する）

  設定した値の等値面（3次元コンタ図）を描画する。

- geometry viewer （描画されたデータを表示する）

  上流のモジュールで作成された描画イメージを描画させている。

モニタリングの表示は Fig.4.4に示すようなものとなる。

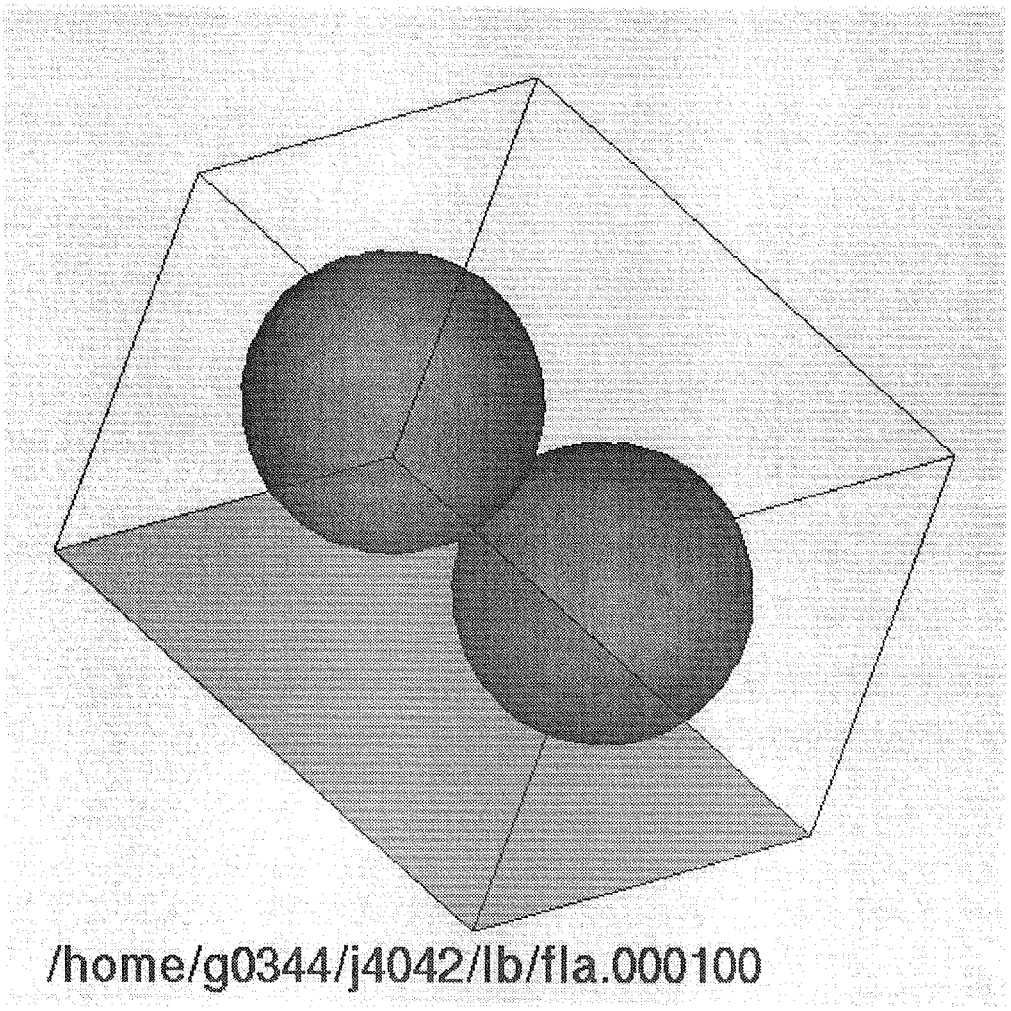

Fig. 4.4　AVS5 でのモニタリング例

4.2 AVS/EXPRESSによるモニタリング

4.2.1 Check Stepモジュール

Fig.4.5にCheck Stepモジュールのコントロールパネルを示す。

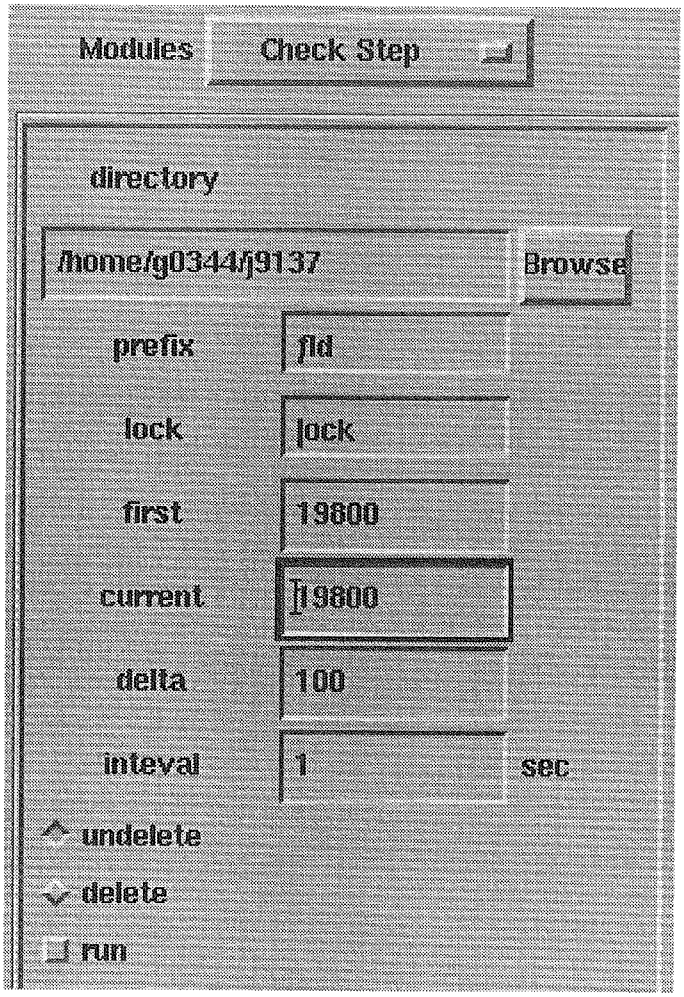

Fig. 4.5 AVS/EXPRESS用Check Stepコントロールパネル

通常の描画を行うアプリケーションのRead Fieldモジュールに Check Stepモジュールを接続し、Fig.4.5のコントロールパネルを表示させる。次にdirectoryにはロックファイル及び解析結果ファイルの格納されるディレクトリ、prefixには解析結果ファイルの接頭子、lockにはロックファイルの接頭子、firstにはモニタ開始ステップ番号、currentには入力なし、deltaにはステップ間引き間隔、intervalにはファイル監視時間サイクルをそれぞれ設定する。ここで各パラメータは入力後必ずenterすることに注意する。描画後に解析結果ファイルを削除する場合にはdeleteを選択し、runをonすることによってモニタが開始される。

4.2.2 Anim Step モジュール

Fig.4.6に Anim Step モジュールのコントロールパネルを示す。

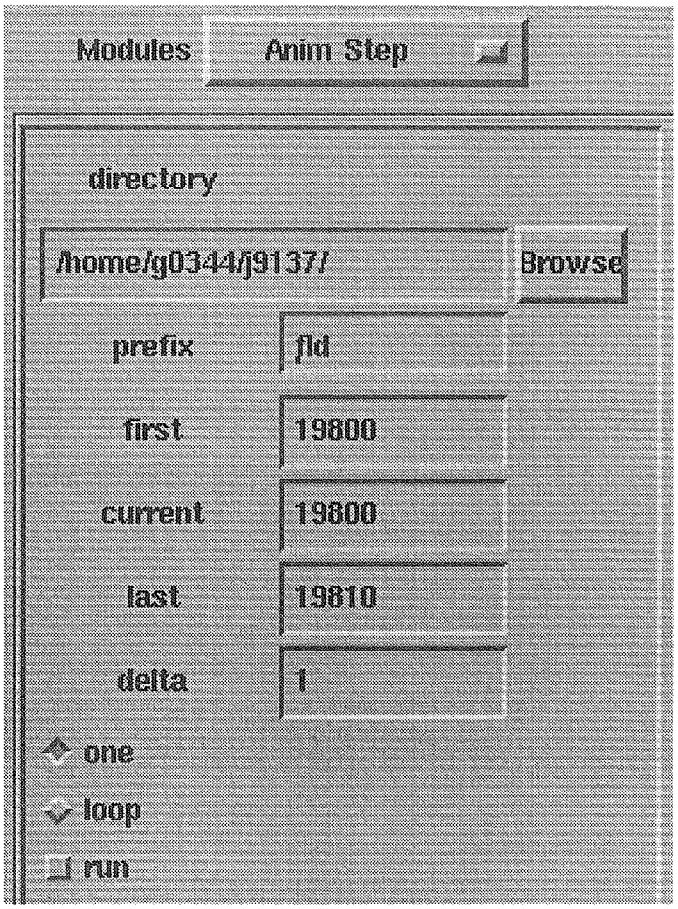

Fig. 4.6 AVS/EXPRESS用 Anim Step コントロールパネル

Check Step と同様に、通常の描画を行うアプリケーションの Read Field モジュールにモジュールを接続し、Fig.4.6のコントロールパネルを表示させる。directory には解析結果ファイルの格納されているディレクトリ、prefix には解析結果ファイルの接頭子、first には可視化開始ステップ番号、last には可視化終了ステップ番号、delta にはステップ間引き間隔をそれぞれ設定する。また Check Step モジュールと同様、パラメータ入力後は enter を行う。繰り返し連続可視化を行う時には loop を on、1回で終了する場合は one を on とし、run を on することで開始される。

4.2.3 表示例

AVS/EXPRESS によるモニタリングの例を以下に示す。AVS5 の例と同様3次元空間での等値面と断面内分布を描画する例で、使用したアプリケーションは Fig.4.7に示すものである。ここでは、描画するファイル名を画面に表示するようにしている。AVS5 の例と同様、Check

Step モジュールにより実時間モニタリングを行うものである。

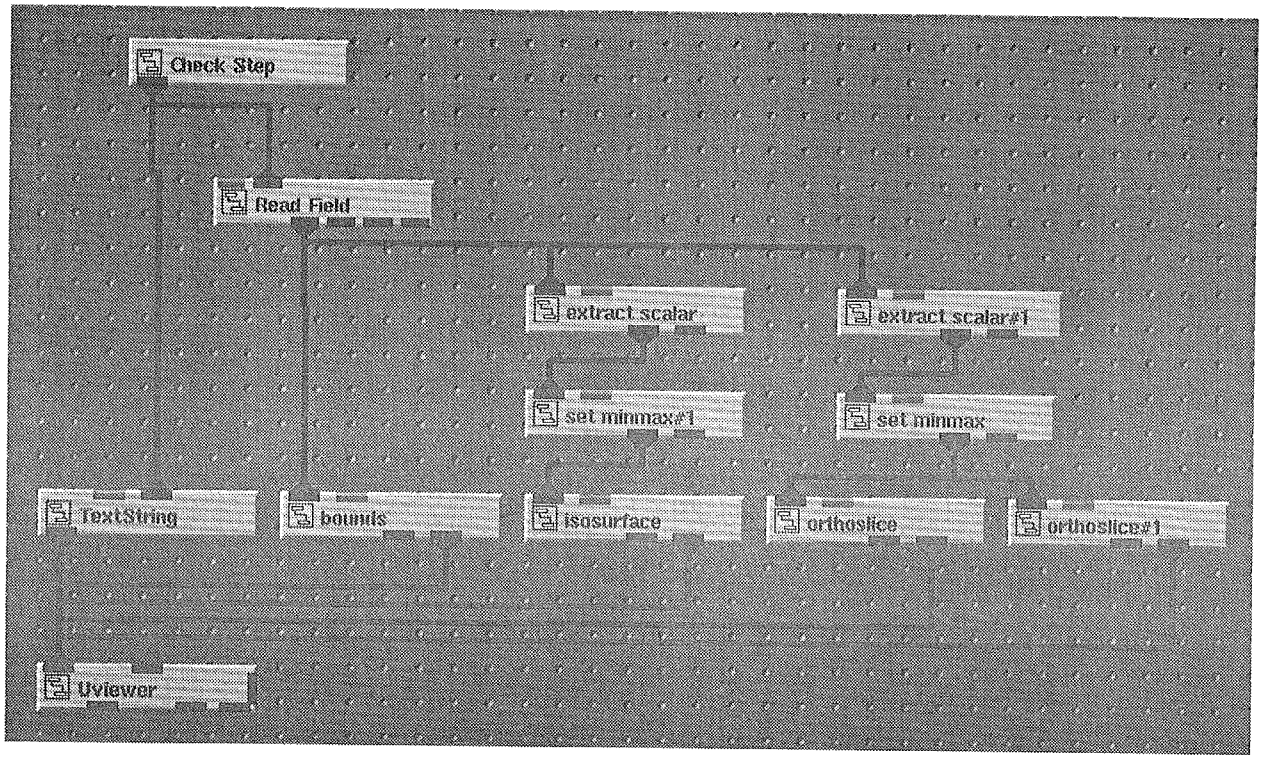

Fig. 4.7　AVS/EXPRESS 用アプリケーション

Fig.4.7のネットワークで使用しているモジュールの機能の概略は、以下の通りである。

- Check Step （ロックファイルを監視し、ファイル名を出力する）

  ロックファイルを監視し、検出した際に対応するファイル名を下流のジュールに渡す。

- Read Field （フィールド・フォーマットのファイルを読み込み、フィールドデータを出力する）

  AVS/EXPRESS のフィールドファイルを読み込み、フィールドデータを出力する。ファイル名はコントロールパネルから入力するのではなく、Check Step モジュールから取り込む。

- extract scalar （複数成分を持つコンポーネント・データの中から１つのサブコンポーネントを抽出する）

  １つのフィールドデータファイルの中に複数種類のデータ（速度、温度、圧力など）が存在するときにその中の１つを抽出する。

- set minmax （ノード・データの範囲、最大最小値を設定する）

AVS/EXPRESSは入力されたデータの最大、最小を検出し、描画の際のカラーデータを自動で設定する。これでは動画のように複数のデータファイルを読み込む際に、ファイルごとにカラーデータ（描画の色）が変わってしまい動画の意味が無くなる。このモジュールに最大、最小を設定し接続しておけば入力データにかかわらず最小、最大を固定することができる。

- bounds （3次元の構造格子に境界枠を生成する）

  読み込んだデータの大きさの枠を描画する。

- orthoslice （構造格子型のデータに対し、格子断面を抽出する。また、格子断面のコンタ図を作成する）

  指定した断面の2次元コンタ図を描画させている。

- isosurface （等値面を作成する）

  設定した値の等値面（3次元コンタ図）を描画する。

- Uviewer （表示オブジェクトを表示し、編集、対話的操作を行うビューワ・ウィンドウを作成する）

  上流のモジュールで作成された描画イメージを描画させている。

モニタリングの表示はFig.4.8のようなものである。

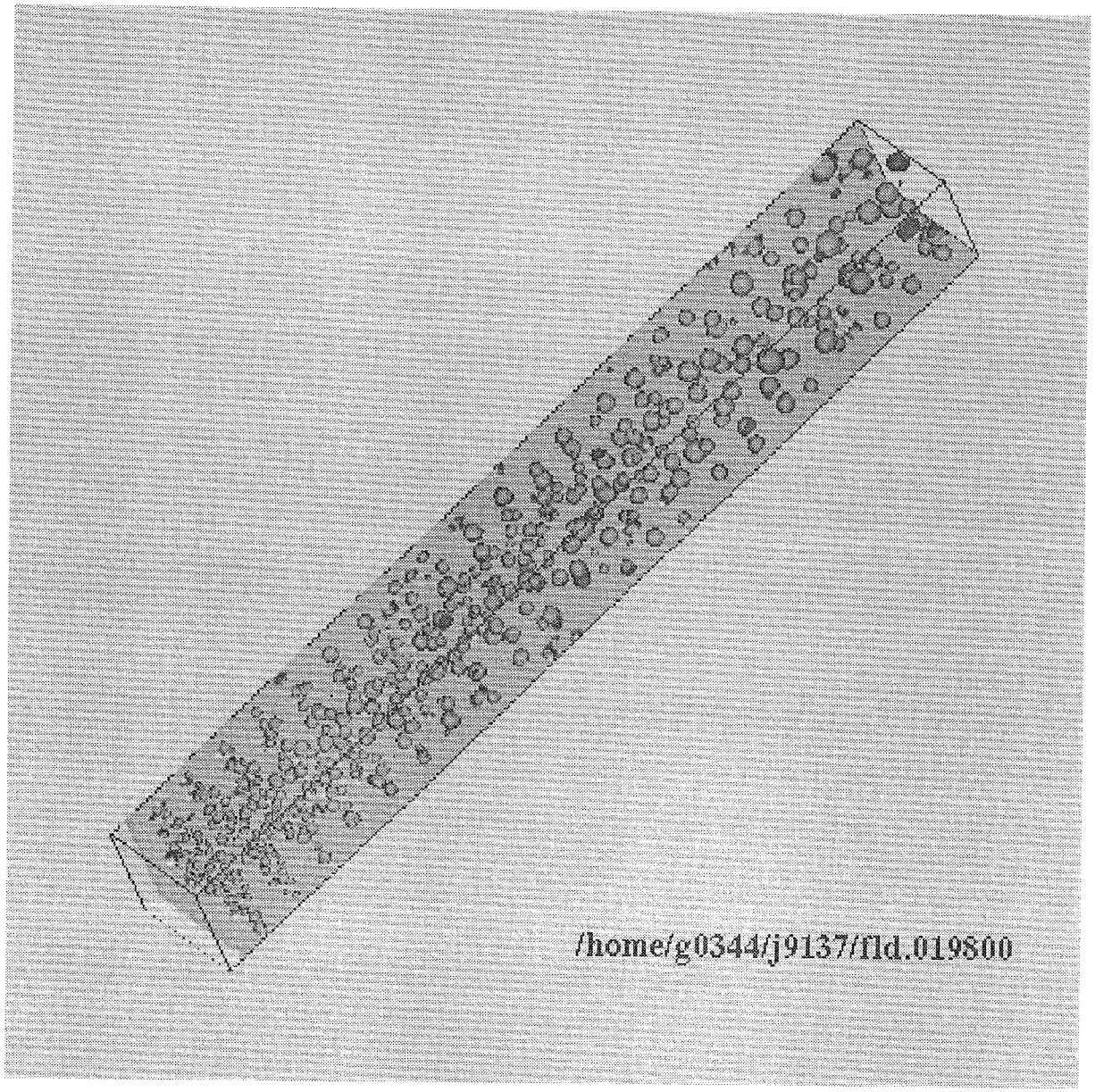

Fig. 4.8　AVS/EXPRESS でのモニタリング例

5. モジュールのコンパイル

開発したモジュールを東海研究所の可視化サーバー以外のワークステーションで使用するためには、可視化を行うワークステーション上で各モジュールのソースプログラムをコンパイルする必要がある。ワークステーションには、AVS5 あるいは AVS/EXPRESS が導入されていなければならない。

5.1 AVS5 用モジュールのコンパイル

check step、anim step、directory browser の各基本モジュールをコンパイルし、マクロモジュールである Check Step、Anim Step を作成するための Makefile を以下に示す。この Makefile を実行することによって実行形式（AVS5 上で自作モジュールとして読み込める）が作成される。

```
AVS_PATH=/usr/avs
CPP_MAIN_LIB=-lC
INC_FILE=$(AVS_PATH)/include/Makeinclude
MLIB=-lm
include $(INC_FILE)

AVS_LIBS=$(AVS_PATH)/lib
BASELIBS=-lgeom -lutil -lm
FBASELIBS=-lgeom -lutil
FLOWLIBS=-L$(AVS_LIBS) -lflow_c $(BASELIBS) $(LASTLIBS)
CSIMLIBS=-L$(AVS_LIBS) -lsim_c $(BASELIBS) $(LASTLIBS)
MODLIBS=-L$(AVS_LIBS) -lmdata -lmfilt -lmmapp -lmrend -lrf
AVS_INC=-I. -I$(AVS_PATH)/include
CFLAGS=$(ACFLAGS) $(AVS_INC)

CPPFLOWLIBS=-L$(AVS_LIBS) -lflow_C $(BASELIBS) $(LASTLIBS)
CPPSIMLIBS=$(CSIMLIBS)
CPPFLAGS=$(CFLAGS) $(CPPLIBS)

F77_FLOWLIBS=-L$(AVS_LIBS) -lflow_f   $(FBASELIBS) $(MLIB) $(LASTFLIBS)
F77_SIMLIBS=-L$(AVS_LIBS) -lsim_f  $(FBASELIBS) $(MLIB) $(LASTFLIBS)
F77_BIND= $(AVS_PATH)/bin/f77_binding
F77_INC= $(AVS_PATH)/include
F77FLAGS=$(AFFLAGS) -I$(F77_INC)
FFLAGS=$(F77FLAGS)

AVS_EXE = $(AVS_PATH)/bin/avs

.SUFFIXES: .o .C .cpp .c

.C.o: $*.C
$(CPP) $(CFLAGS) -c $*.C
```

```
.cpp.o: $*.cpp
$(CPP) $(CFLAGS) -c $*.cpp

MODULES=\
checkstep\
animstep\
dirbrowser\
Read_Vsc
MACRO_MODULES=\
CheckStep\
AnimStep
MODLIB=Monitoring
MODLIB_SRC=ModLib.src
MODLIB_TEMPL=ModLib.templ

TOP_PATH=$(PWD)/..

#*****************************************************************************

default: all
all: $(MODULES)
clean:
-rm -f $(MODULES) $(MODLIB) $(MODLIB_SRC) *.o core
install: all
test -d ../avs_library || mkdir ../avs_library
cp $(MODULES) $(MACRO_MODULES) ../avs_library
make mod_lib
cp $(MODLIB) ../avs_library

mod_lib: $(MODLIB)
$(MODLIB):
sed -e "s+@MONITORING_PATH@+$(TOP_PATH)+" $(MODLIB_TEMPL) >\
$(MODLIB_SRC)
$(AVS_EXE) -compile_library $(MODLIB_SRC) $@

#*****************************************************************************

checkstep: checkstep.o
$(CC) $(CFLAGS) -o $@ checkstep.o $(CSIMLIBS)
animstep: animstep.o
$(CC) $(CFLAGS) -o $@ animstep.o $(CSIMLIBS)
dirbrowser: dirbrowser.o
$(CC) $(CFLAGS) -o $@ dirbrowser.o $(FLOWLIBS)

#*****************************************************************************
#
# data read sample module.
#
#*****************************************************************************

Read_Vsc: Read_Vsc.o
```

```
$(F77) $(F77FLAGS) -o $@ Read_Vsc.o $(F77_FLOWLIBS)

#############################################################################
```

5.2 AVS/EXPRESS 用モジュールのコンパイル

まず、AVS/EXPRESS を起動するため、AVS/EXPRESS の格納場所と、実際に AVS/EXPRESS を実行するディレクトリのパスを以下のように設定する。これは、コンパイル及び実行の際必要となるものである。

```
XP_PATH=/home/g0344/j0000/exp  /wka0/AVSVIZ/express
```

ここで設定した/home/g0344/j0000/exp をプロジェクトルートと呼びコンパイルしたモジュールを使用して AVS/EXPRESS を実行する際は、プロジェクトルートの直下の

```
/home/g0344/j0000/exp/bin/sgi/express
```

を実行しなければならない。

次に、コンパイルの作業を、exp というディレクトリで行うこととし、ソースファイルを exp/src/ に作成する。

```
exp/src/animstep.c
exp/src/checkstep.c
```

次に、AVS/EXPRESS を起動し、addmodule 機能により、作成するべきモジュールの入出力パラメータの定義を AVS/EXPRESS 画面上で行う。これにより、以下に示す v ファイルと呼ばれるファイルが生成される。

```
exp/v/
exp/v/lib_vxp.v
exp/v/macro/
exp/v/macro/AnimStep.v
exp/v/macro/CheckStep.v
exp/v/macro/templ.v
exp/v/module/
exp/v/module/animstep.v
exp/v/module/checkstep.v
exp/v/module/templ.v
exp/v/templ.v
```

続いて、先に設定したプロジェクトルートにおいて、以下のように base コマンドを実行する。

```
% /wka0/AVSVIZ/express/bin/sgi/base -project . -gen_proc express -exit
```

このbaseコマンドにより、makefileに相当するexpress.mkが生成される。このexpress.mkを実行することによりモジュール作成が終了する。

```
% make -f express.mk
```

6. おわりに

グラフィックスのためのソフトウェアや技術開発が細分化し、可視化ツールや機器のグラフィックス機能が豊富になるにともない、可視化ツールは使用方法が簡単かつ不変的であることが利用上重要なこととなってきている。シミュレーション研究のかたわら可視化や画像処理技術の開発を行うのは困難であり、また新しいソフトウェアや機器を導入する度に使用方法を習得するのは効率的ではないためである。

本報告書では、シミュレーションの途中経過の実時間可視化、あるいは計算結果の連続可視化を行うためのシミュレーションモニタリングシステムの開発と使用方法について解説した。実時間可視化、あるいはスーパーコンピュータのモニタリングといった概念や手法は多数考えられるものであるが、ここでは汎用可視化ツールであるグラフィックスソフトウェアAVS5およびAVS/EXPRESSを使用している。このため、異なる計算機環境であってもAVS5あるいはAVS/EXPRESSが用意されていれば、モニタリングシステムを利用することが可能である。また、通常の利用方法として、東海研究所の画像処理サーバ用ワークステーションをネットワークから利用することにより、モニタリング機能をユーザーの端末から利用することが可能である。

スーパーコンピュータにより詳細かつ大規模な計算が可能となり、計算結果が膨大なものになると、全ての数値結果を詳細に検討することは到底不可能である。膨大な数値の中から何をどのように表示すると何が理解できるかが、シミュレーション研究の現場における可視化の課題であるが、効率的で扱いの容易な可視化表示方法を含む大規模数値シミュレーション技術の開発・整備により、シミュレーション結果の正確かつ効率的な理解が可能となる。本報告書が効率的なシミュレーション研究の一助となることを期待する。

謝　辞

本作業に協力していただいた（株）KGT、ソルーショングループ内田賢一郎氏、同、ビジュアリゼーショングループ宮地英生氏他に感謝致します。また、有益な助言をいただいた並列処理支援技術開発グループ村松一弘氏に感謝致します。

参考文献

[1] T.Watanabe and H.Kaburaki: "Particle simulation of three-dimensional convection patterns in a Rayleigh-Benard system", Phys. Rev. E, 56, 1218(1997).

[2] 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 95-014(1995).

[3] 渡辺 正: "計算科学研究における可視化技術の応用", JAERI-Conf 96-002(1996).

[4] 加藤 克海, 渡辺 正, 蕪木 英雄: "リアルタイムモニターシステムの開発", JAERI-Tech 96-044(1996).

[5] 富士通株式会社: "VisLink 利用マニュアル", (1997).

[6] 村松 一弘, 大谷 孝之, 松本 秀樹, 竹井 利文, 土肥 優: "並列計算機上での流体解析のための実時間可視化システムの開発", JAERI-Data/Code 98-014(1998).

[7] クボタコンピュータ株式会社: "AVS USER'S GUIDE", (1993).

[8] Advanced Visual Systems Inc.: "AVS User's Guide Release 4", (1992).

[9] 株式会社ケイ・ジー・ティー: "AVS/EXPRESS ユーザーズ・ガイド", (1999).

[10] 株式会社ケイ・ジー・ティー: "AVS/EXPRESS チュートリアル・ガイド", (1999).

国際単位系 (SI) と換算表

表 1　SI 基本単位および補助単位

量	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd
平面角	ラジアン	rad
立体角	ステラジアン	sr

表 2　SI と併用される単位

名称	記号
分，時，日	min，h，d
度，分，秒	°，′，″
リットル	l，L
トン	t
電子ボルト	eV
原子質量単位	u

1 eV = 1.60218 × 10^{-19} J

1 u = 1.66054 × 10^{-27} kg

表 5　SI 接頭語

倍数	接頭語	記号
10^{18}	エクサ	E
10^{15}	ペタ	P
10^{12}	テラ	T
10^{9}	ギガ	G
10^{6}	メガ	M
10^{3}	キロ	k
10^{2}	ヘクト	h
10^{1}	デカ	da
10^{-1}	デシ	d
10^{-2}	センチ	c
10^{-3}	ミリ	m
10^{-6}	マイクロ	μ
10^{-9}	ナノ	n
10^{-12}	ピコ	p
10^{-15}	フェムト	f
10^{-18}	アト	a

表 3　固有の名称をもつ SI 組立単位

量	名称	記号	他の SI 単位による表現
周波数	ヘルツ	Hz	s^{-1}
力	ニュートン	N	m·kg/s^2
圧力，応力	パスカル	Pa	N/m^2
エネルギー，仕事，熱量	ジュール	J	N·m
工率，放射束	ワット	W	J/s
電気量，電荷	クーロン	C	A·s
電位，電圧，起電力	ボルト	V	W/A
静電容量	ファラド	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	V·s
磁束密度	テスラ	T	Wb/m^2
インダクタンス	ヘンリー	H	Wb/A
セルシウス温度	セルシウス度	℃	
光束	ルーメン	lm	cd·sr
照度	ルクス	lx	lm/m^2
放射能	ベクレル	Bq	s^{-1}
吸収線量	グレイ	Gy	J/kg
線量当量	シーベルト	Sv	J/kg

表 4　SI と共に暫定的に維持される単位

名称	記号
オングストローム	Å
バーン	b
バール	bar
ガル	Gal
キュリー	Ci
レントゲン	R
ラド	rad
レム	rem

1 Å = 0.1 nm = 10^{-10} m

1 b = 100 fm^2 = 10^{-28} m^2

1 bar = 0.1 MPa = 10^5 Pa

1 Gal = 1 cm/s^2 = 10^{-2} m/s^2

1 Ci = 3.7 × 10^{10} Bq

1 R = 2.58 × 10^{-4} C/kg

1 rad = 1 cGy = 10^{-2} Gy

1 rem = 1 cSv = 10^{-2} Sv

（注）

1. 表 1 — 5 は「国際単位系」第 5 版，国際度量衡局 1985 年刊行による。ただし，1 eV および 1 u の値は CODATA の 1986 年推奨値によった。
2. 表 4 には海里，ノット，アール，ヘクタールも含まれているが日常の単位なのでここでは省略した。
3. bar は，JIS では流体の圧力を表わす場合に限り表 2 のカテゴリーに分類されている。
4. EC 閣僚理事会指令では bar，barn および「血圧の単位」mmHg を表 2 のカテゴリーに入れている。

換　算　表

力	N(=10^5 dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

粘　度　1 Pa·s(N·s/m^2) = 10 P(ポアズ)(g/(cm·s))

動粘度　1 m^2/s = 10^4 St(ストークス)(cm^2/s)

圧力	MPa(=10 bar)	kgf/cm^2	atm	mmHg(Torr)	lbf/in^2(psi)
	1	10.1972	9.86923	7.50062 × 10^3	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322 × 10^{-4}	1.35951 × 10^{-3}	1.31579 × 10^{-3}	1	1.93368 × 10^{-2}
	6.89476 × 10^{-3}	7.03070 × 10^{-2}	6.80460 × 10^{-2}	51.7149	1

エネルギー・仕事・熱量	J(=10^7 erg)	kgf·m	kW·h	cal(計量法)	Btu	ft·lbf	eV
	1	0.101972	2.77778 × 10^{-7}	0.238889	9.47813 × 10^{-4}	0.737562	6.24150 × 10^{18}
	9.80665	1	2.72407 × 10^{-6}	2.34270	9.29487 × 10^{-3}	7.23301	6.12082 × 10^{19}
	3.6 × 10^6	3.67098 × 10^5	1	8.59999 × 10^5	3412.13	2.65522 × 10^6	2.24694 × 10^{25}
	4.18605	0.426858	1.16279 × 10^{-6}	1	3.96759 × 10^{-3}	3.08747	2.61272 × 10^{19}
	1055.06	107.586	2.93072 × 10^{-4}	252.042	1	778.172	6.58515 × 10^{21}
	1.35582	0.138255	3.76616 × 10^{-7}	0.323890	1.28506 × 10^{-3}	1	8.46233 × 10^{18}
	1.60218 × 10^{-19}	1.63377 × 10^{-20}	4.45050 × 10^{-26}	3.82743 × 10^{-20}	1.51857 × 10^{-22}	1.18171 × 10^{-19}	1

1 cal = 4.18605 J(計量法)

= 4.184 J　(熱化学)

= 4.1855 J　(15 °C)

= 4.1868 J(国際蒸気表)

仕事率　1 PS(仏馬力)

= 75 kgf·m/s

= 735.499 W

放射能	Bq	Ci
	1	2.70270 × 10^{-11}
	3.7 × 10^{10}	1

吸収線量	Gy	rad
	1	100
	0.01	1

照射線量	C/kg	R
	1	3876
	2.58 × 10^{-4}	1

線量当量	Sv	rem
	1	100
	0.01	1

（86 年 12 月 26 日現在）

シミュレーションモニタリングシステムの開発