つくる情熱を、支える情熱。
CYBERNET

第16回ビジュアリゼーションカンファレンス

AVS/Expressチュートリアル

バッチ処理入門

サイバネットシステム株式会社

もくじ

AVS/Expressチュートリアル（バッチ処理入門）

- AVS/Expressおさらい
- バッチ処理（定型処理）を行うための仕組みの紹介
- 画像出力方法
- 動画出力方法
- （GFA出力方法）
- おわりに

CYBERNET

AVS/Expressおさらい

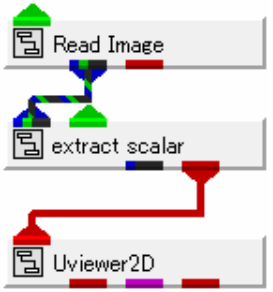
「モジュール」と呼ばれる四角い箱を繋ぎ独自の表示（機能）を実現
→ モジュールプログラミング

２つのモジュールで機能を実現する例

イメージデータを読み込む処理

データを表示する処理

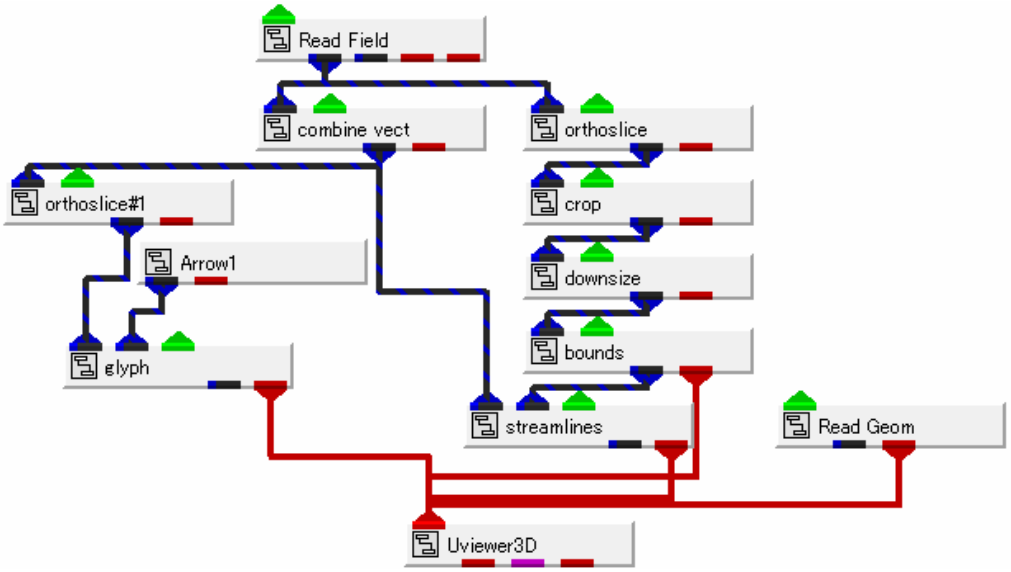
更にモジュールを追加した例



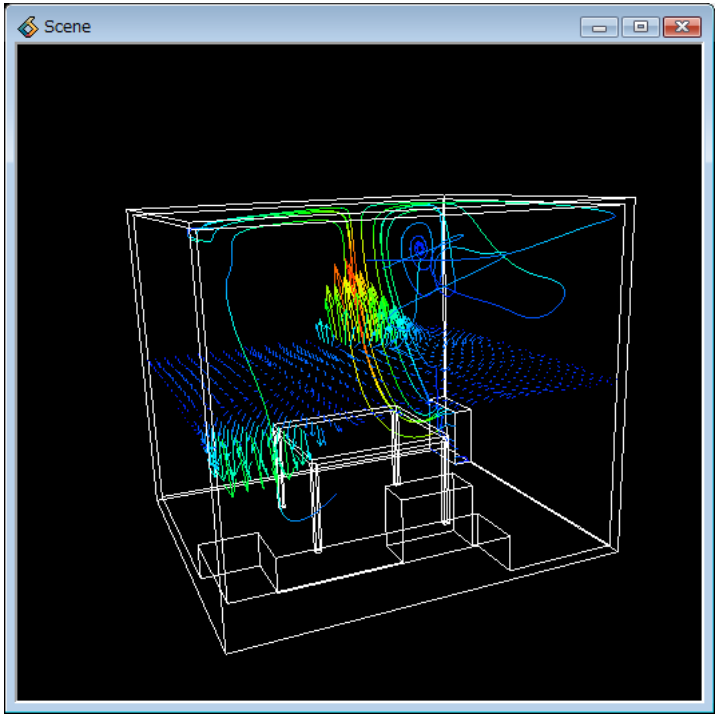
複数成分から任意の成分を抽出する処理

※この画像は成分の分布を分かり易くするため黒から緑の色分布に変更して表示。

AVS/Expressおさらい



ネットワーク図



室内空間の空調シミュレーション

対話的に機能を切り替えて表示できることは分かった。
でも、毎回新しい表示を行っている訳ではないし・・・・
複数のデータに対して同様の処理を簡単に行えたらいいけど・・・・

AVS/Expressは対話処理用の
ソフトウェアじゃないの？

いろいろ組み合わせると
バッチ処理（定型処理）にも利用できます!!

本日はこの方法を紹介します。

定型処理を行うには

以下の仕組みを利用する。

- V言語（アプリケーションファイル）
- getenv()【組み込み関数】
- OutputField+Write_Image【モジュール】
- -exit オプション【起動時オプション】
- -offscreen【起動時オプション】
- exit_process【モジュール】
- parse_v【モジュール】
- $push～$pop【Vコマンド】

定型処理を行うには

以下の仕組みを利用する。

- V言語（アプリケーションファイル）
- getenv()【組み込み関数】
- OutputField+Write_Image【モジュール】
- -exit オプション【起動時オプション】
- -offscreen【起動時オプション】
- exit_process【モジュール】
- parse_v【モジュール】
- $push～$pop【Vコマンド】

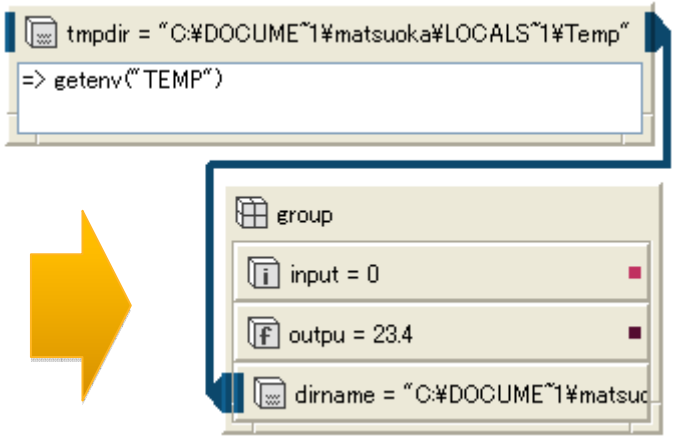
V言語　その１

- AVS/Express のカーネルであるオブジェクト・マネージャーとのインターフェースとして用いられる中間言語である。

- AVS/Express のオブジェクトはすべて V で定義されており、ネットワーク・エディター、ライブラリなどもすべて V で定義されているオブジェクトである。AVS/Express は起動時にこの V の定義を読み込み、部品やアプリケーションを構築する。

- V 言語の記述例

```
#ifdef MSDOS
string tmpdir<NEportLevels=1> => getenv("TEMP");
#else
string tmpdir<NEportLevels=1> => "/tmp";
#endif
group group {
      int input = 0;
      float outpu = 23.4;
      string dirname<NEportLevels={2,0}> => <-.tmpdir;
};
```



Windowsで読み込んだ結果

- オブジェクト保存、アプリケーション保存時のファイルもV言語で保存される。（メモ帳などで開いて確認）

V言語　その２

```
#ifdef MSDOS                ←もしWindowsマシンだったら以下を実行（Vファイル読み込み時一度だけ判断される）
string tmpdir<NEportLevels=1> => getenv("TEMP");
#else                       ←Windowsマシンではなかったら以下を実行
string tmpdir<NEportLevels=1> => "/tmp";
#endif                      ←判断文終了
group group {               ←group オブジェクトを group 名でインスタンス
  int input = 0;            ←int型（整数）オブジェクトを input 名でインスタンスし、0を設定
  float outpu = 23.4;       ←float型（単精度浮動小数点）オブジェクトをoutput名でインスタンスし、23.4を設定
  string dirname<NEportLevels={2,0}> => <-.tmpdir;      ←dirnameとtmpdirを接続
};
```

モジュールの階層と接続の記述

.	今いる階層	=	値の代入
<-	一段上位の階層	=>	値のリンク
{}	階層の区切り		

例）a=>b は、aはbと接続され値を参照していることを示す。a=bは一時的な値の代入で、bの値が変化してもaは変化しない。
（詳細はデベロッパーズガイド第1部参照）

定型処理を行うには

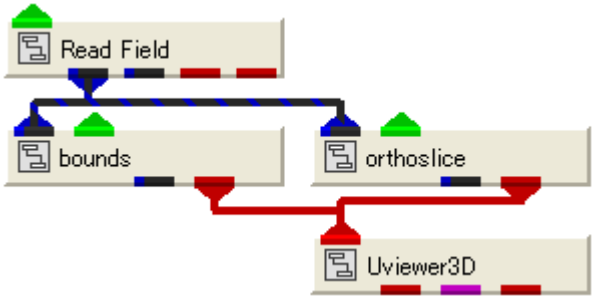
以下の仕組みを利用する。

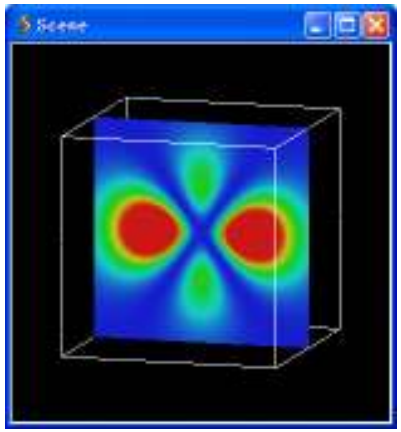
- V言語（アプリケーションファイル）
- getenv()【組み込み関数】
- OutputField+Write_Image【モジュール】
- -exit オプション【起動時オプション】
- -offscreen【起動時オプション】
- exit_process【モジュール】
- parse_v【モジュール】
- $push～$pop【Vコマンド】

getenv()【組み込み関数】その１

環境変数で設定された値を返す。
システムで設定された環境変数を取得するほか、ユーザーが設定した値も取得可能。
入出力のファイル名やパラメータ値を環境変数で指定することができる。

1. 例えば、以下のネットワークを作成して可視化後、アプリケーション保存する。





2. 保存したアプリケーションファイルを開きRead_Fieldを検索する。

```
                                                    .
MODS.Read_Field Read_Field<NEx=36.,NEy=36.> {
    read_field_ui {
      file_browser {
        ok = 1;
        x = 100;
        y = 100;
      };
      filename = "C:¥¥Express¥¥data¥¥field¥¥hydrogen.fld";
    };
    DVread_field {
                                                    .
```

3. filename 部分を以下のように修正し上書き保存する。

```
filename => getenv("MY_DATA_FILE");
```


※Vファイルの編集時は他のワードを消さないよう十分注意

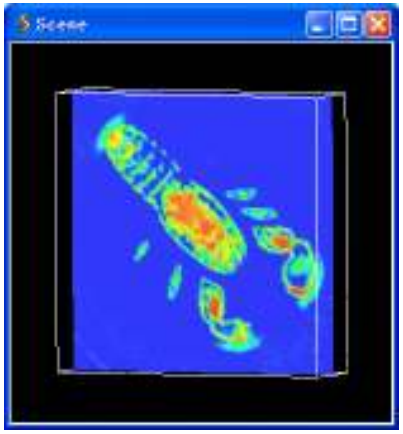
getenv()【組み込み関数】その２

4. 以下のような起動用バッチファイル利用すると、外部で読み込むファイルを指定できる。

```
REM　起動バッチファイル　exprun.bat
REM

set EXPDIR=C:¥Express
set MACHINE=pc

set MY_DATA_FILE= C:¥Express¥data¥field¥ lobster.fld
%EXPDIR%¥bin¥%MACHINE%¥express –ne sample.v
```



備考）
起動時に　-ne　オプションでアプリケーションファイルを指定すると、自動的に読み込まれる。（Viz/Express 時は –viz オプションも必要）

注意）
バッチファイル内でファイル名を指定する際は、short file name(SFN)を指定する必要あり。（dir /x で確認可能）

orthoslice のパラメータなどモジュールのパラメータも同様に設定可能。

モジュールのパラメータがどのキーワードか分からない場合は、特定の値を設定後、アプリケーション保存すると確認し易い。（値に変化の無いパラメータは出力されない）

定型処理を行うには

以下の仕組みを利用する。

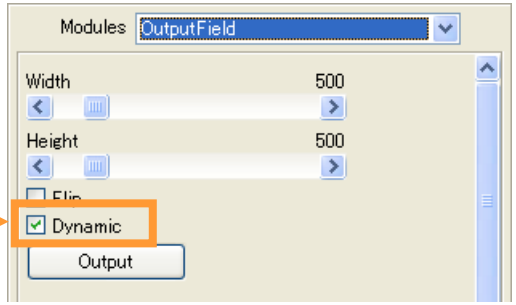
- V言語（アプリケーションファイル）
- getenv()【組み込み関数】
- OutputField+Write_Image【モジュール】
- -exit オプション【起動時オプション】
- -offscreen【起動時オプション】
- exit_process【モジュール】
- parse_v【モジュール】
- $push～$pop【Vコマンド】

OutputField+Write_Image【モジュール】その１

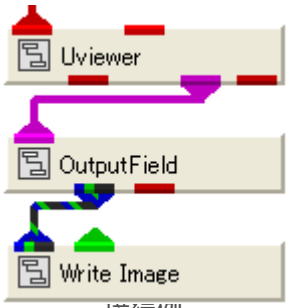
画像を出力する。

OutputFiled
ビューの内容をAVS/Expressのイメージ・フィールド・データとして出力する。（表示画面をaRGBの2次元Field データとして出力する）

DynamicをOn
（画面が描画されるたびに処理を行う）

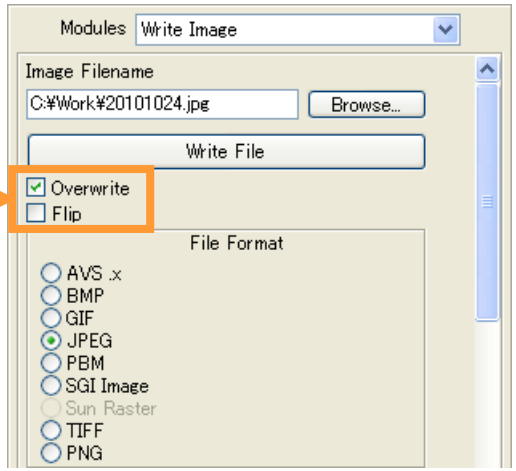


Write_Image
フィールド・データをイメージ・ファイルに出力する。



接続例

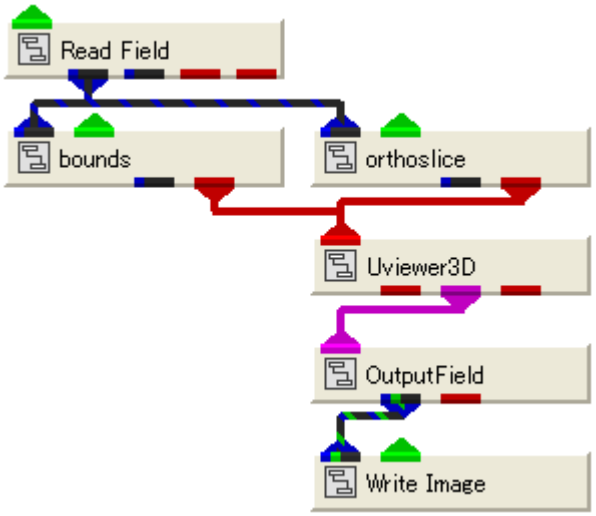
OverwriteをOn
FlipをOff
（上書き保存、画像を見た目と同じ向き）



この設定で画像が描画されたらファイルに出力される。

OutputField+Write_Image【モジュール】その２

読み込み時同様に出力もgetenv()を利用し、Write_Imageで出力するファイル名を設定する。




```
MODS.Write_Image Write_Image<NEx=189.,NEy=252.> {
    in => <-.OutputField.output_field.Output.output;
    write_image_ui {
      overwrite_toggle {
        set = 1;
      };
      file_browser {
        filename => getenv("MY_OUTPUT_FILE");
        ok = 1;
        x = 100;
        y = 100;
      };
```

アプリケーションファイルのWrite_image部分を編集

起動用バッチファイルに以下の設定を追加して実行。

```
set MY_OUTPUT_FILE= C:¥work¥orthoslice.jpg
```

定型処理を行うには

以下の仕組みを利用する。

- V言語（アプリケーションファイル）
- getenv()【組み込み関数】
- OutputField+Write_Image【モジュール】
- -exit オプション【起動時オプション】
- -offscreen【起動時オプション】
- exit_process【モジュール】
- parse_v【モジュール】
- $push～$pop【Vコマンド】

-exit【起動時オプション】

処理終了後にAVS/Express本体を終了する。

```
REM　起動バッチファイル　exprun.bat
REM

set EXPDIR=C:¥Express
set MACHINE=pc

set MY_DATA_FILE= C:¥Express¥data¥field¥hydrogen.fld
set MY_OUTPUT_FILE= C:¥work¥hydrogen.fld
%EXPDIR%¥bin¥%MACHINE%¥express –ne sample.v  -exit
```

```
set MY_DATA_FILE= C:¥Express¥data¥field¥hydrogen2.fld
set MY_OUTPUT_FILE= C:¥work¥hydrogen2.jpg
%EXPDIR%¥bin¥%MACHINE%¥express –ne sample.v  -exit

set MY_DATA_FILE= C:¥Express¥data¥field¥hydrogen3.fld
set MY_OUTPUT_FILE= C:¥work¥hydrogen3.jpg
%EXPDIR%¥bin¥%MACHINE%¥express –ne sample.v  -exit
```

参考)
AVS/Express Dev Edition でかつ、UNIX/Linux 系のマシンでは –offscreen オプションを利用できる。
-offscreenオプションは、ウィンドウを表示せずに画像出力可能。

UNIX/Linux 系以外は、通常の起動と同じようにウィンドウが表示され処理が行われる。この際、Sceneパネルが画面内に収まるようにする必要がある。

CYBERNET

ここまでの処理の確認

AVS/Express上で以下の操作を行った時と同じ

1. AVS/Expressを起動
2. アプリケーションファイルの読み込み
3. Read_Filedでデータファイルを指定
4. OutputImageでイメージファイル名を指定
5. AVS/Express終了

起動と終了以外は、アプリケーションファイルを書き換えるだけで実行可能。

CYBERNET

複数ステップのデータを扱うには

複数ステップのデータを扱うには・・・（Read_Fieldを利用して動画を作成する場合）
以下の処理が必要。

1. AVS/Expressを起動
2. アプリケーションファイルの読み込み
3. Read_Filedでデータファイルを指定
4. image_captureのModeをCapture from Viewに変更
5. Read_FiledのOne-Timeのチェックを入れる

（複数ステップのデータが連続的に読み込まれる）

6. image_captureで出力するファイル名を指定
7. image_captureのGenerate Movieボタンを押す
8. AVS/Express終了

先の仕組みだけで行うと最初のステップが終わると終了してしまう。

定型処理を行うには

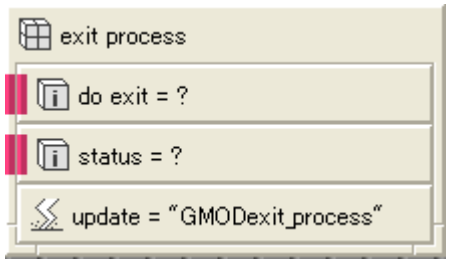
以下の仕組みを利用する。

- V言語（アプリケーションファイル）
- getenv()【組み込み関数】
- OutputField+Write_Image【モジュール】
- -exit オプション【起動時オプション】
- -offscreen【起動時オプション】
- exit_process【モジュール】
- parse_v【モジュール】
- $push～$pop【Vコマンド】

exit_process、parse_v【モジュール】

exit_process

AVS/Express本体を終了する。
ランタイムアプリケーション（販売用アプリケーション）作成時など、独自のUIからAVS/Express本体を終了する際に利用。

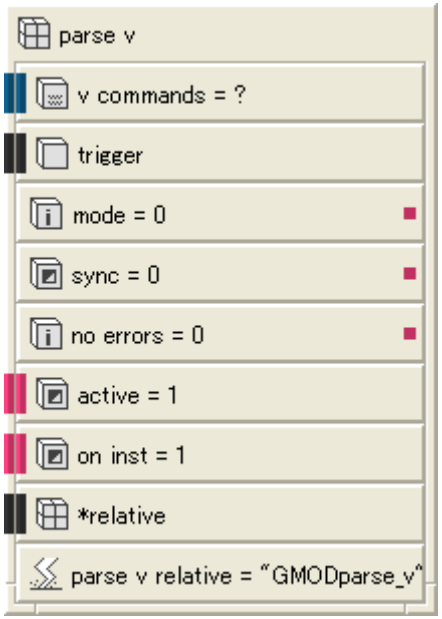


do_exit に1を設定すると即座に終了する。

動画作成時は起動時オプション –exit では対応できないため、本モジュールを利用して終了処理を行う。

parse_v

triggerに変化が生じると v_command に記述されたV言語を実行する。



v_command部分に処理内容を記述する。

$push～$pop【Vコマンド】

実行のコントロール
$popコマンドが呼び出されるまで実行の配信を遅らせる。

※ アプリケーションファイルは、上から順番に実行されるとは限らない。この為、
確実に指定した順番で実行させた場合はこの記述を利用する。

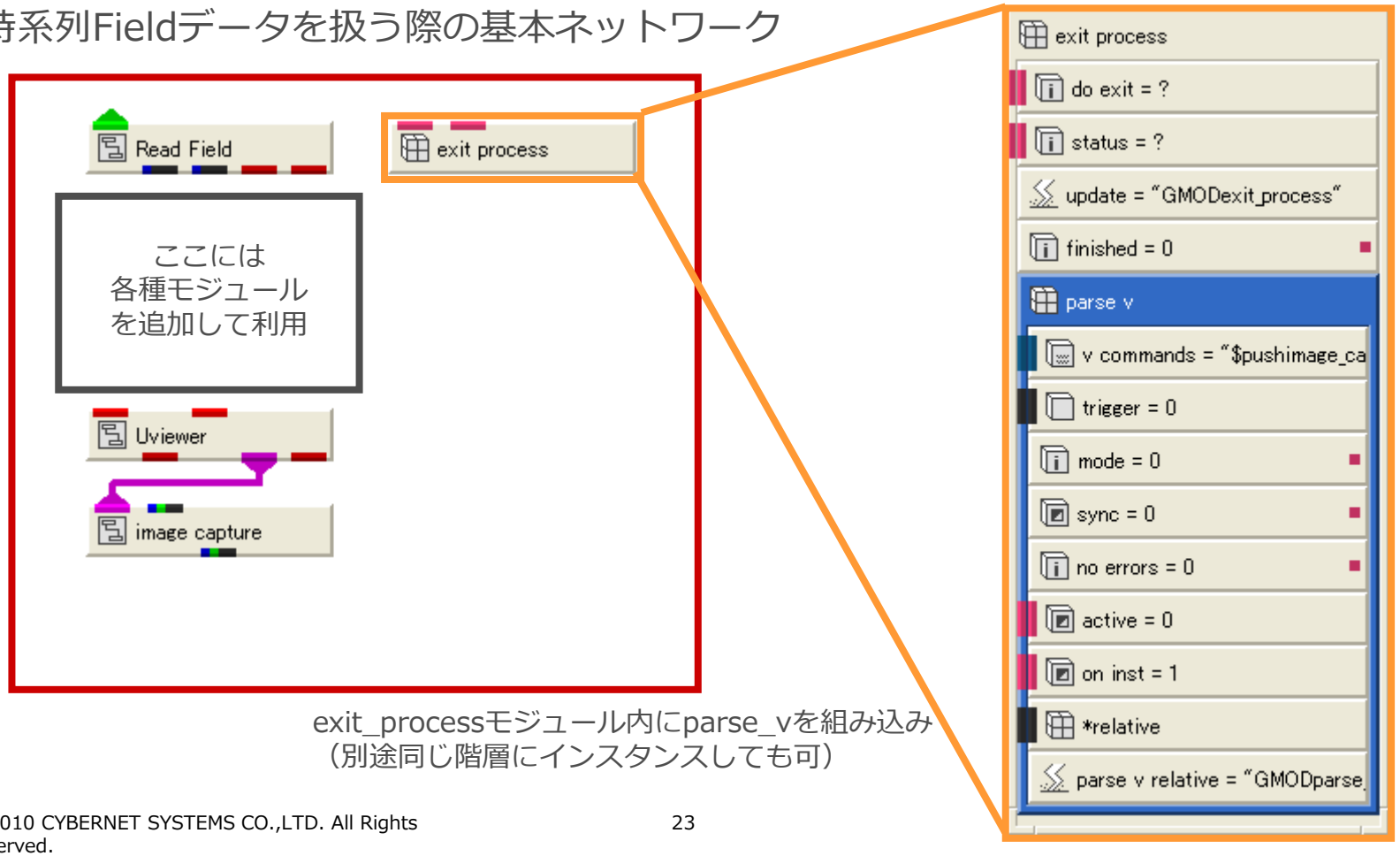
例）以下のように記述すると確実にこの順番で実行される。

```
$push
A=B;
$pop
$push
B=C;
$pop
```

複数ステップデータの処理

複数ステップを連続的に流し、image_capture で動画保存する方法

時系列Fieldデータを扱う際の基本ネットワーク



exit_processモジュール内にparse_vを組み込み
（別途同じ階層にインスタンスしても可）

動画作成用アプリケーションファイル記述例

```
$push
APPS.MultiWindowApp MultiWindowApp<NEdisplayMode="maximized"> {
  MODS.Read_Field Read_Field<NEx=27.,NEy=27.> {
    read_field_ui {
      filename => getenv(“MY_DATA_FILE”);                    ←読み込みファイル名

    };
  };
   GDM.Uviewer Uviewer<NEx=135.,NEy=189.> {
    Scene {
      Top {
        child_objs => {<-.<-.<-.xxxxxxxxxxxxxxx.out_obj};
      };
    };
  };
  ANIM_MODS.image_capture image_capture {
    imcapCompute.ImageCap {
      LAnimOutOverwrite = 1;
      LGDView => <-.<-.<-.Uviewer3D.Scene_Selector.curr_view;
    };
    imcapParam {
      capture_mode = 1;
      filename => getenv(“MY_OUTPUT_FILE”);                  ←出力ファイル名
    };
  };
```

次ページに続く

CYBERNET

動画作成用アプリケーションファイル記述例

前ページからの続き

.

```
GMOD.exit_process exit_process {
  int finished => (<-.Read_Field.Read_Field_Param.current_step >= <-
   .Read_Field.Read_Field_Param.total_steps);          ←Read_Filedのステップが最後か判断
  GMOD.parse_v parse_v {                              ←parse_vの記述
    v_commands =
    "$push¥n "
    +“image_capture.imcapParam.gen_movie=1;¥n“        ←image_captureのGenerate
    +"$pop¥n "                                          Movieボタンを押す
    +"$push¥n"
    +“exit_process.do_exit=1;¥n“                      ←exit_processのdo_exitに1を設定
    +"$pop¥n";
    active => <-.finished;                            ←finishedが真の場合実行可能
    trigger => <-.finished;                           ←finishedが真の場合実行
    relative => <-.<-;
  };
 };
};
$pop
```

.

次ページに続く

動画作成用アプリケーションファイル記述例

```
前ページからの続き
.

//以下の処理は全てのモジュールが実行された後に処理をしたいため、最後に記述する。
$push
MultiWindowApp {
   image_capture.imcapParam.mode = 1;          ← ModeをCapture from Viewに変更
   Read_Field.Read_Field_Param.one_time=1;     ← Read_FieldのOne-timeにチェックを入れる
};
$pop
```

※ここで紹介しているサンプルは、アプリケーション保存したファイルから不必要なものを削除し、複数ステップ用のルーチンを追加している。

```
$push
アプリケーションファイル（先頭データの表示）  ①

   exit_process {
      Read_Fieldが最後のステップになったらimage_captureのGenerate
      Movieboボタンを押し、動画の作成が終わったら自身を終了する。   ③
   }
$pop

$push
image_captureのmodeをCapture from Viewに変更
Read_FieldのOne-TimeのチェックOn                ②
$pop
```

①②③の順番で実行される

デモ

1. 画像出力
2. 動画出力
3. GFA出力

360ステップのデータを連続的に読み込み、動画およびGFAを作成する。（粒子数10000個）

おわりに

AVS/Expressはリアルタイムでの可視化以外にも、多少手を加えることでバッチ処理（定型処理）にも利用できることを確認いただけたのではないでしょうか？

今回紹介した方法は、定型処理を行う際の一つの方法です。今回紹介していませんが、この他ジャーナル機能を利用する方法もあります。
いろいろな利用方法がAVS/Expressには組み込まれていますので、これらを利用し業務や研究に役立てていただければ幸いです。

P.S.
ご要望やお悩みをお持ちの方は、ぜひ弊社にご相談下さい。
お客様が抱えている問題を解決する為のお手伝いをさせていただきます。

今回紹介したモジュールなどの利用方法は、英語のオンラインマニュアルの他、日本語版（PDF）マニュアルにも記載しています。（主にデベロッパーズガイド）以下のサイトからもダウンロードいただけます。

http://kgt.cybernet.co.jp/feature/viz_manual/

ご利用上の注意：

サイバネットシステム株式会社
ビジュアリゼーション部
〒101-0022東京都千代田区神田練塀町3
富士ソフトビル
http://www.cybernet.co.jp/