2013/9/25

AVS/Express Viz 版と Dev 版の違いについて

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
AVS サポートセンター

AVS/Express の Viz 版（Visualization Edition）と Dev 版（Developer Edition）の代表的な違いを示します。

1. Dev 版でのみサポートしている機能（機能の違い）

以下の機能は、Dev 版のみでサポートしています。

・オフスクリーンによるバッチ処理

Viz 版では、オフスクリーンによるバッチ処理はサポートされていません。

・グラフモジュール（AG kit）

Viz 版では、予め使い方が規定されたグラフモジュールのみがサポートされています。

Dev 版では、軸やチックマークなど、部品レベルでのモジュールがサポートされています。

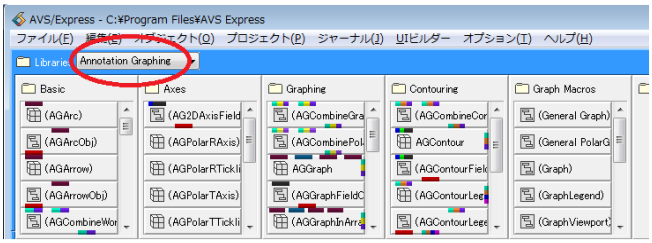

図 1 Dev 版でサポートされている AG Kit

・既存モジュールの改変

Dev 版では、既存モジュールの中を開き、その中で行われている処理を改変できます。

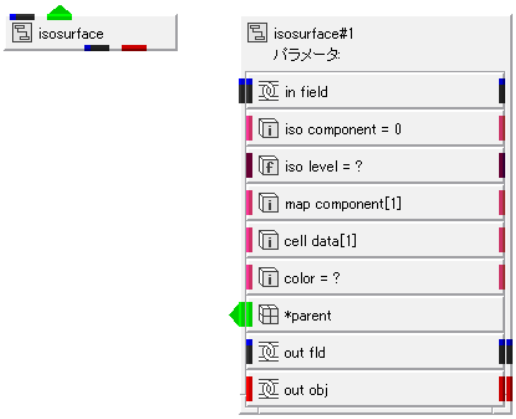

図 2 Viz 版でモジュールをダブルクリックして開いた様子

Viz 版では、モジュールの中身で見ることができるのは、公開されている一部のパラメーターだけです。

Dev 版では、下図のように、モジュールを構成しているさらに内部のモジュールまで見ることができます。

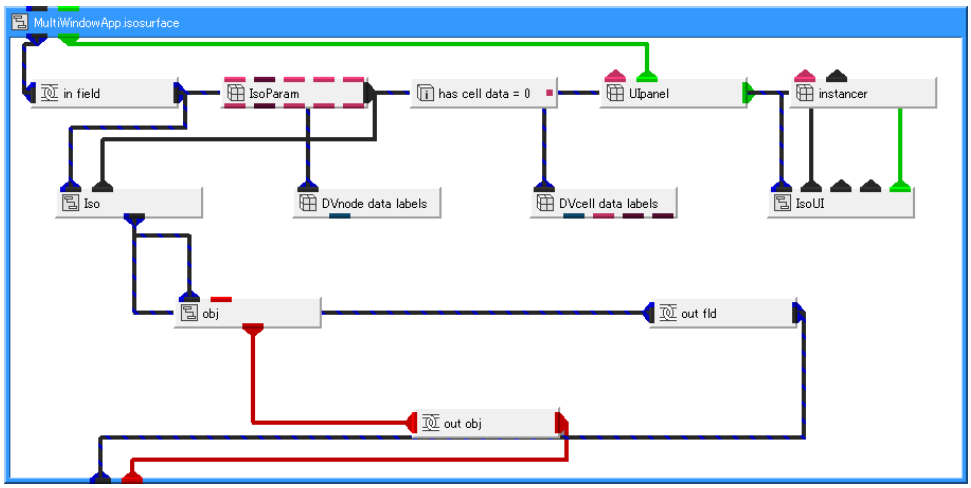

図 3 Dev 版でモジュールをダブルクリックして開いた様子

Dev 版では、このように内部を見ることができますので、その中の構成を改変することができます。

・ユーザーインターフェースの改良

また、同様に、スライダーや数値入力などのユーザーインターフェースも、モジュールの中では、
個々の UI モジュールで構成されています。
Dev 版では、その中身を開き、ユーザーインターフェースを変更することもできます。

・モジュールの基本機能の利用

各可視化のモジュールは、その処理関数が組み込まれた基本モジュールとパラメーターなどを集めた
グループやマクロモジュール、また、ユーザーインターフェースなどの各種モジュールで構成されています。
Dev 版では、この基本モジュールだけを使うこともできます。
例えば、等値面を作成する isosurface モジュールの中の基本モジュールは DViso モジュールです。
Dev 版では、このユーザーインターフェース等を持たない基本モジュールのみを使うこともできます。

図 4 可視化モジュールと基本モジュール

・ランタイムアプリケーションの作成

Dev 版では、AVS/Express 上で可視化した状態をコンパイルし、ランタイムアプリケーション化できます。
（ただし、実行にはライセンスが必要です）

2.　Dev 版のメリット（実例）

Viz 版と Dev 版の一番の違いは、Viz 版では、各モジュールが規定、許可している使い方以上の利用方法はできないという点です。

Dev 版では、以下のように、既存機能を改良することができます（以下にいくつかの実例を示します）。

・モジュール定義の改変

Dev 版では、モジュールの中の定義を改変することができます。

例えば、Arrow3 モジュールは、流れのベクトル図の可視化などで利用できる矢印を作成するモジュールです。

図のように、矢尻に 3 本のラインを持っています。

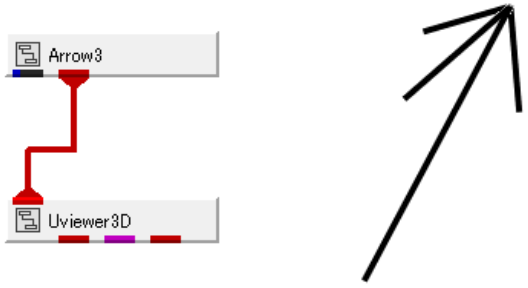

図 5 Arrow3 モジュール

Dev 版では、例えば、その座標情報を改変し、その矢尻を 4 本に変更するようなことが可能です。

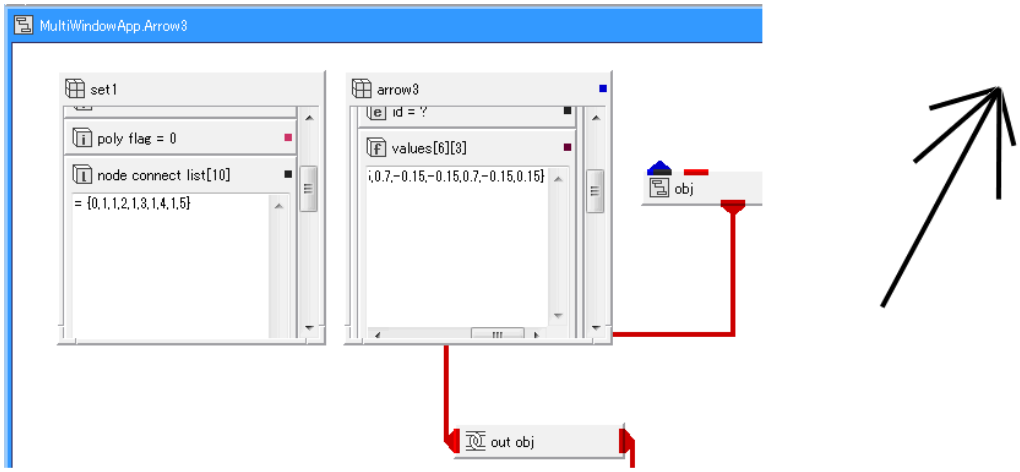

図 6 Dev 版でモジュールの定義を修正し、矢尻の数を増やした例

（図左は Arrow3 モジュールの内部を開いた様子）

・可視化モジュールの拡張

各可視化のモジュールも同様に内部を改良することができます。

下図の例では、slice_plane モジュールを利用し、任意断面を 3 枚、作成しています。

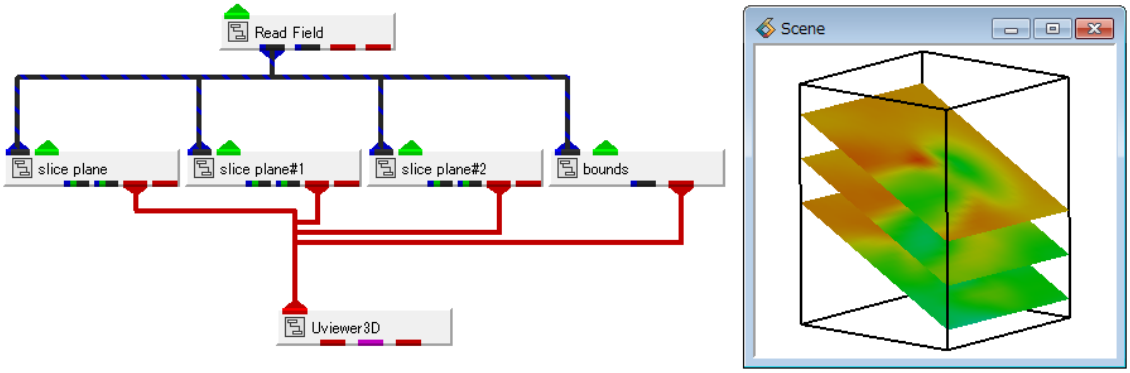

図 7　3 枚の任意断面の表示

AVS/Express では、同じ処理を複数回行うには、そのモジュールを複数個利用することで、実現できます。
ただ、このように複数のモジュールを利用した場合、パラメーターをそれぞれ設定しなければなりません。
例えば、ある間隔で並んだ複数枚の任意断面を作るには、モジュール内部にある基本モジュールをコピーして、
３枚の断面を作るように拡張すると便利です。

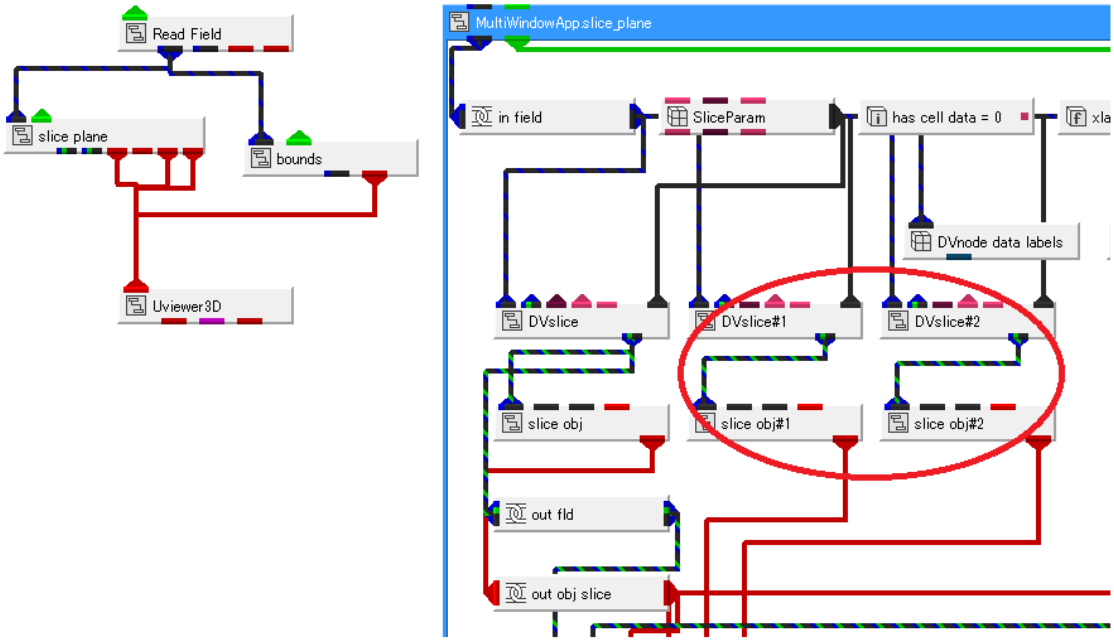

図 8 任意断面を 3 枚出力するように改良した slice_plane モジュール

・モジュール間のパラメーターの授受

あるモジュールで設定された値を、別のモジュールの入力パラメーターとして利用したいことがあります。
Dev 版では、このような動作を自由に設定できます。
下図の例では、Read_Field モジュールで読み込んだ time 文字列をそのまま表示するのではなく、
別の文字列を追加（前方に Current = を追加）して表示しています。
Read_Field モジュールの内部を改良することで、このような改変が可能です。

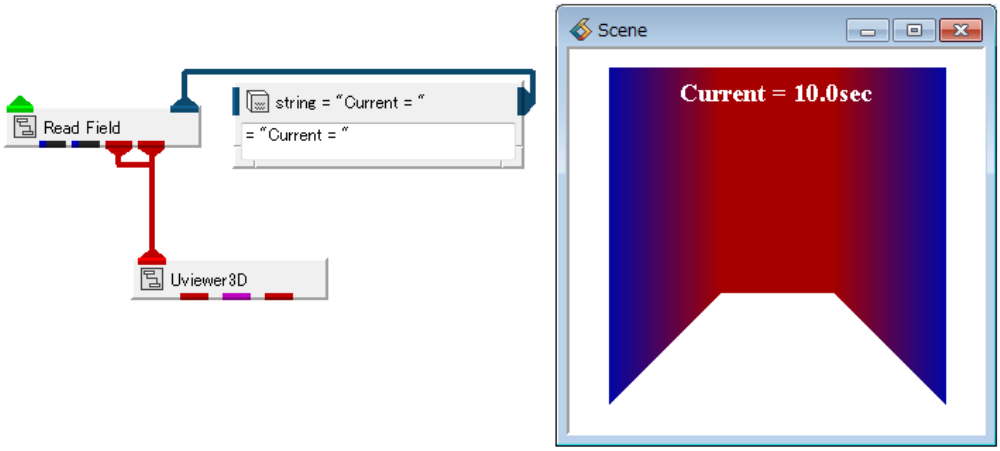

図 7 Read_Field モジュールに外部から文字列を与え、ラベルの表示方法を改良した例

・ビューワーサイズの指定

ビューワーモジュールも、その内部を構成するモジュールを見ることができます。

例えば、ビューワーのウィンドウサイズも外部から与えることができます。

下図の例では、断面コンター図のサイズと同じ比率のビューワーを作成し、そこに、

コンター図表示を行っています。

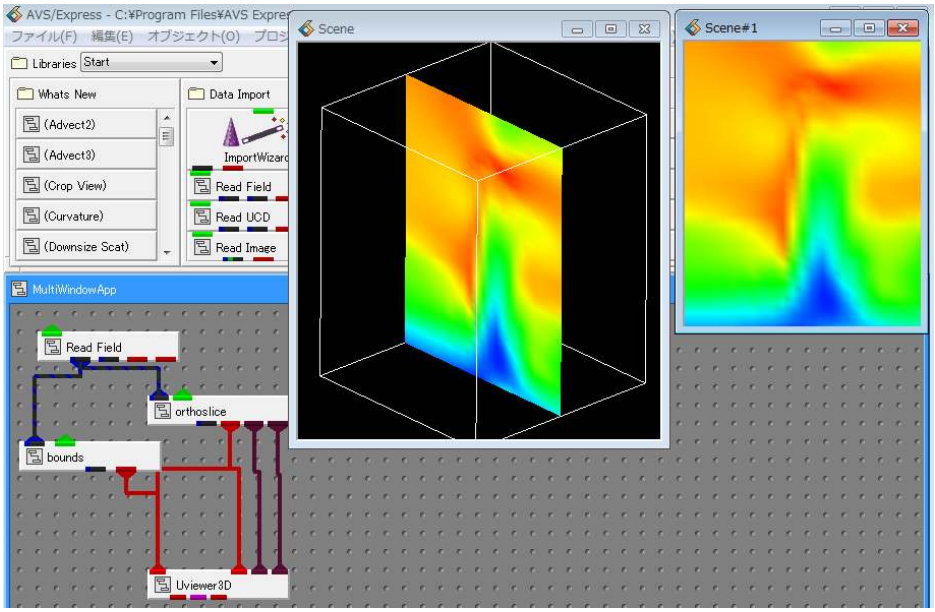

図 8 格子断面と同じ比率のビューワーに断面コンターを表示した例

・モジュール処理の自動化

また、一部モジュールには、ボタンをクリックすることで実行する仕様となっている

モジュールもあります。

Dev 版では、このボタンをクリックするというトリガーを別のモジュールから受け渡して

自動化することも可能です。

例えば下図の Rd_HDF5_Field モジュールは、Read_File ボタンをクリックすることで

データを読み込む仕様となっています。

このトリガーを別のモジュール（下図の例では Loop のカウント）から与えて、

データの読み込みを行うように、改良して利用しています。

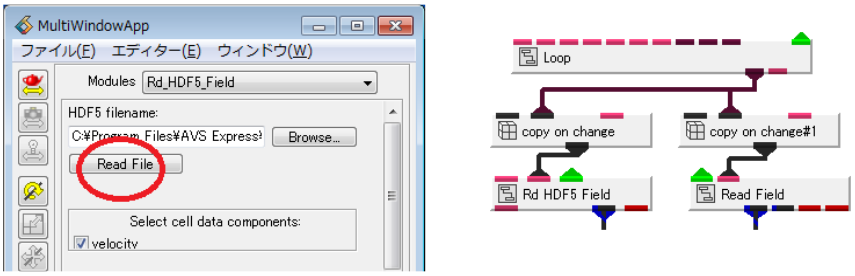

図 8 トリガーの自動化

・V コマンドによる自動化

処理の自動化は、どのモジュールのどのパラメーターを操作するかについて、以下のような V コマンドを作成することで行います。

MultiWindowApp.isosurface.IsoParam.iso_level = 12.5;

（等値面のレベル値を設定する例）

このドットでつながる文字は、モジュールの階層を意味しています。

Viz 版でもこのコマンドを知ることはできますが、簡単ではありません。

Dev 版では、実際にモジュールを開き、階層を確認しながら見ることができますので、

自動化の作業工数が削減できます。

以上