汎用可視化ツール Explorer の使い方

大阪大学大型計算機センター研究開発部
出口　弘
deguchi@harmony.center.osaka-u.ac.jp

スーパーコンピュータの出現によって、大規模計算を必要とする計算機シミュレーションという方法が物理学や化学をはじめとする様々な分野で使われています。そして、その数値計算の結果である膨大なデータを視覚的にとらえること、いわゆる Scientific Visualization（計算結果の可視化）も必要不可欠となっています。本センターでもこのような可視化の必要性に答えるため、グラフィックワークステーション IRIS 4D/310VGX を導入しました。

可視化のために提供されているツールには２つのレベル、ライブラリとパッケージアプリケーションがあります。グラフィックライブラリを使ってプログラミングすれば、ほぼ思い通りの可視化ができますが、膨大な時間と労力と専門的な技術を必要とするプログラミングやデバッグの作業が伴います。一方、特定の分野に合わせて開発されたパッケージアプリケーションは、使い易い反面、柔軟性に欠け、全ての分野の可視化のためにそれらのプログラムを揃えるには余りにも高価です。そこで登場したのがスターデント社の AVS (Application Visualization System) やシリコングラフィックス社の Explorer などです。これらのシステムでは、データを用意し、可視化のための部品（モジュール）を対話的に接続するだけで手軽に可視化を行なうことができます。本センターが導入した IRIS 4D/310VGX には IRIS Explorer がバンドルされています。そこで、今回の可視化特集となったわけです。

Explorer は、科学者や技術者に対して、可視化機能と分析機能を提供するユーザ環境を備えたアプリケーション作成システムです。ユーザは、IRIS Explorer に付属する各種モジュールを接続するだけで、専門的なプログラミング知識を使うことなく、研究や問題解決のための可視化を短時間に行なうことができます。データ入力、データ処理、データ表示、データ出力などの機能モジュールが用意されています。また、Module Builder というユーティリティを利用することにより、ユーザ定義の機能モジュールを作成することもできます。さらに、データの入出力を行なうためのモジュールは、DataScribe というユーティリティを利用することにより、プログラミングなしで作成することができます。

本稿では、Explorer と DataScribe の簡単な使い方を説明します。さらに、本センターで実際に可視化を行なった「フォームクライオターゲットの爆縮実験データの可視化」と「イオンビームプロセス・シミュレーションの可視化」の２件の事例を紹介します。

Explorer

1. はじめに

IRIS Explorer は、様々な分野の科学技術計算の結果を可視化するための汎用ポスト処理ツールです。スーパーコンピュータ等で計算した結果である数値データを Explorer で可視化するためには、まずそのデータを Explorer 型のデータタイプに変換する必要があります。この変換用のモジュールを作成するシステムが後で述べる DataScribe です。

Explorer の主なデータタイプには、

格子（Lattice）
ピラミッド（Pyramid）
幾何学（Geometry）

があります。最も一般的に使われるデータタイプである格子データタイプには、

定型（Uniform）
直交型（Perimeter）
不定型（Curvilinear）

の３種類があります。

例えば、２次元画像データの場合は、ＲＧＢ（赤緑青）の各値を要素とする２次元定型格子データ（2D-Uniform-Lattice）となります。また、ＣＴ画像やＭＲ画像のようなボクセルデータは、要素が１つの３次元ボリュームデータであり、３次元定型格子データ（3D-Uniform-Lattice）となります。定型の格子点は等間隔に整列しているため、格子点に対してその座標データを与える必要はありません。一方、ＣＦＤ（計算流体力学）のデータのように格子点が整列していない場合は、各格子点に対してその座標データと要素データを与えることができる不定型格子データ（Curvilinear-Lattice）が使えます。

ピラミッドデータタイプは階層的な格子であり、複雑な分子構造を表現したり、有限要素法のデータを表現するために使われます。

幾何学データタイプは、可視情報（色、材質感等）を持った多角形（Polygon）、線（Line）、点（Point）等の基本形状の集まりです。物体の形状を表現したい場合には、まずそれを多面体表現し、各頂点の座標と面情報を与えます。

本稿では、サンプルデータの可視化を例題に使いながら説明します。詳細は、「IRIS Explorer User's Guide」をご覧ください。

2. 起動と終了

2.1 起動の仕方

Explorer の起動は、シェルウィンドウで以下のようにタイプします。

```
% /usr/explorer/bin/explorer
```

起動すると、トップレベル（Explorer）、マップエディタ（Map Editor）、モジュールライブラリ（localhost）の３つのウィンドウが図１の様に画面に現れます。

2.2 終了の仕方

Explorer のトップレベルウィンドウの Admin メニューを開き、Quit を選択します。具体的には、それぞれマウスカーソルを Admin、Quit に合わせてマウスの左ボタンをクリックします。

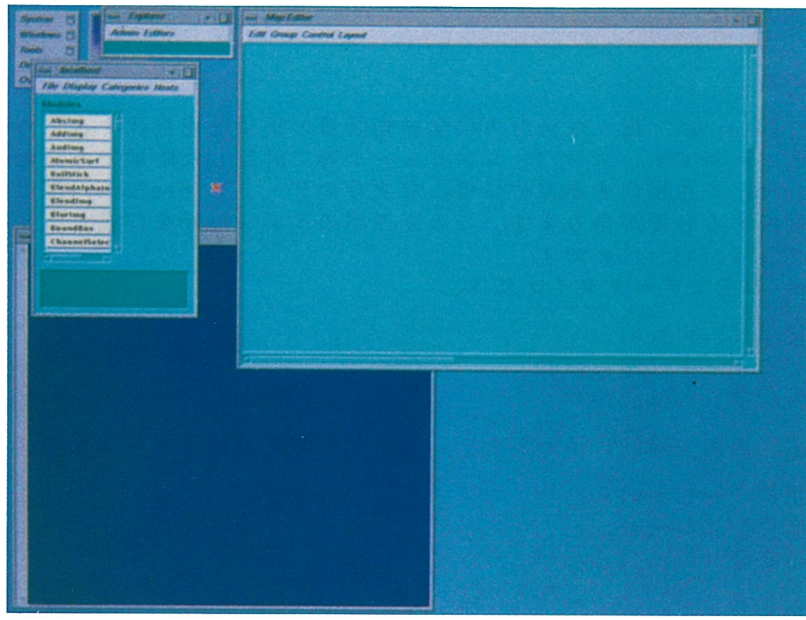


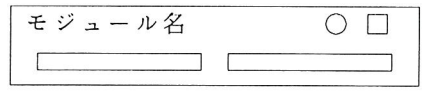
図1　Explorer の起動画面

3. モジュール

Explorer のモジュールはサブルーチンのような単一処理能力を持つユニットであり、必要なモジュールをマップエディタで接続し、マップと呼ばれる可視化ネットワークを構成します。モジュールライブラリに存在するライブラリだけで所望の機能を実現できない場合は、ファイル入力のようなデータフォーマット変換モジュールであれば DataScribe、それ以外は Module Builder というユーティリティを使って、新たに構築することができます。

3.1 モジュールの概要

各モジュールには、モジュール間の接続に使用する入力ポートパッド（左側の長方形）と出力ポートパッド（右側の長方形）があります。すなわち、データはモジュールの左側から入って、処理され、右側から出て行きます。各モジュールにおけるパラメータの値はマウスで対話的に変更することができ、リアルタイムで最終結果に反映します。パラメータ値の入力形態には、ボタン、ダイヤル、スライダー、テキスト（キーボード入力）などがあります。



3.2 モジュールのサイズ

モジュールには次の3つのサイズがあります。

マイクロサイズ　：　モジュール名と接続情報だけがわかる
ミニサイズ　：　モジュールの操作が一応可能
フルスケールサイズ　：　モジュールの詳細な表示と操作が可能

マイクロサイズとミニサイズは、モジュールの右上の丸ボタンをクリックすることによって入れ替わります。また、フルスケールサイズは、モジュールの右上の四角ボタンをクリックすることによって新たにウィンドウが表示されます。このウィンドウは他のウィンドウと同様に、大きさを変えたり（ウィンドウの縁をつかんでドラッグする）、アイコン化したり（右上の小さい丸ボタンをクリックする）、画面一杯に広げたり（右上の四角ボタンクリックする）できます。これらの操作は全てマウスの左ボタンで行ないます。

3.3 モジュールの種類

モジュールは以下の４種類に大きく分けることができます。

入力モジュール　：　データをファイルから読み込む
機能モジュール　：　データを処理する
表示モジュール　：　データを画面に表示する
出力モジュール　：　データをファイルへ書き出す

モジュールを機能別にまとめると以下のようになります。

データ読み込み、生成

GenerateColormap, GenLat, ReadGeom, ReadImg, ReadLat, ReadPDB, ReadPyr, WaveFromColormap

画像処理

AbsImg, AddImg, AndImg, BlendAlphaImg, BlendImg, BlurImg, CompassAngleImg, CompassDirImg, CompassImg, ComplementImg, DivImg, ExpImg, ForwardFFTImg, FourierConjgImg, FourierCrossCorrImg, FourierDivImg, FourierExpFltImg, FourierGaussFltImg, FourierMagnitudeImg, FourierMergeImg, FourierMultImg, FourierPhaseImg, FourierPwrImg, GaussBlurImg, GrayScaleImg, HistEqualImg, HistNormImg, HistScaleImg, InverseFFTImg, LaplaceEdgeImg, LogImg, MaxFltImg, MaxImg, MedianFltImg, MinFltImg, MinImg, MultImg, NegImg, OrImg, PowerImg, RankFltImg, RobertEdgeImg, RotZoomImg, SGIPalleteImg, SharpenImg, SobelEgdeImg, SqRootImg, SquareImg, SubtractImg, ThreshImg, XorImg

データ処理

AtomicSurf, DiffLat, DisplaceLat, CropPyr, Gradient, Interpolate, LatFunction, LattoPyr, MagnitudeLat, OrthSlice, PickLat, ProbeLat, ProbePyr, PyrToLat, SampleCrop, ScaleLatNode, Shell, Slice, SubSample

幾何学表現

BallStick, BoundBox, Contour, IsosurfaceLat, IsosurfacePyr, LatToGeom, PryToGeom, VectorGen, VolumeToGeom, WireFrame

画像生成

DisplayImg, Histgram, Graph, Render, TransformGen

データ出力

PrintLat, PrintPyr, WriteGeom, WriteImg, WriteLat, WritePyr

詳細な説明は、「IRIS Explorer Module Definitions」をご覧下さい。

4. マップエディタとモジュールライブラリ

4.1 マップエディタ

マップエディタは、モジュールライブラリから選んだモジュールを接続し、マップと呼ばれる可視化ネットワークを構築、修正するワークエリアです。マップエディタの主な機能には、そのメニューに沿って、

Edit : モジュールのカット&ペースト、コピー、消去
Group : モジュールのグループ制御
Control : モジュールの機能制御
Layout : モジュール間接続ワイヤーの属性変更

などがあります。まず、操作の対象を選択し（クリックする）、メニューからコマンドを選択します。これらの操作はマウスの左ボタンで行ない、Macintosh の操作とほとんど同様です。

4.2 モジュールライブラリ

モジュールライブラリはモジュールを供給するためのファイルブラウザーです。モジュールライブラリの主な機能には、モジュールやマップをマップエディタへ呼び出すこと、構築したマップを保存登録することなどがあります。

呼び出しには、モジュールライブラリからマップエディタにドラッグ&ドロップ（所望のモジュールの上でマウスのボタンを押してそのモジュールを選択し、ボタンを押したままマウスを移動して所望の位置まで移動し、ボタンを放す）する方法と、File メニューから Open を選び、ファイルブラウザーを呼び出して、所望のモジュールやマップを選択する方法があります。

保存登録は、File メニューから Save All を選び、ファイルブラウザーを呼び出して、ファイル名を与えます。マップを保存するときには、.map という拡張子を付けなければなりません。

/usr/explorer/maps には幾つかのサンプルマップがあります。

4.3 モジュールの接続と切断

モジュールの接続操作にはマウスの右ボタンを使います。まず、データを渡すモジュールの出力ポートパッド上で、右マウスボタンを押し、ポップアップ表示されるポートの中から接続するものを選びます。次に、そのデータを受けるモジュールの入力ポートパッド上で、右マウスボタンを押し、ポップアップ表示されるポートの中から対応するものを選びます。この受け渡しのモジュールを選ぶ順序は、逆でも構いません。また、データタイプは Lattice と Lattice、Geom と Geom のように、接続するモジュール間で一致していなければなりません。

既に接続されているモジュールのポートを選ぶと、<CONNECT> バーの下に、相手モジュールとポート名から成る、接続リストが表示されます。

モジュールの接続を切断する場合には、切断するモジュールのポートの接続リストから対応する接続（相手モジュール::ポート）を選択します。

また、2つ以上のモジュールと接続する場合には、接続するポートを選ぶ際に表示される接続リストの中から <CONNECT> を選択します。

5. 例題（サンプルデータの可視化）

5.1 マップの作成（モジュールの呼び出しと接続）

図２は、ReadLat（格子データを読み込む）、IsosurfaceLat（等値面を張る）、Render（可視化出力を表示する）の３つのモジュールをドラッグ＆ドロップによってマップエディタに呼び出した後、ReadLat の出力を IsosurfaceLat に接続しようとしているところです。すなわち、ReadLat の出力ポートパッドを右マウスボタンで押し、ポプアップメニューから Output--Lattice ポートを選んでいます。

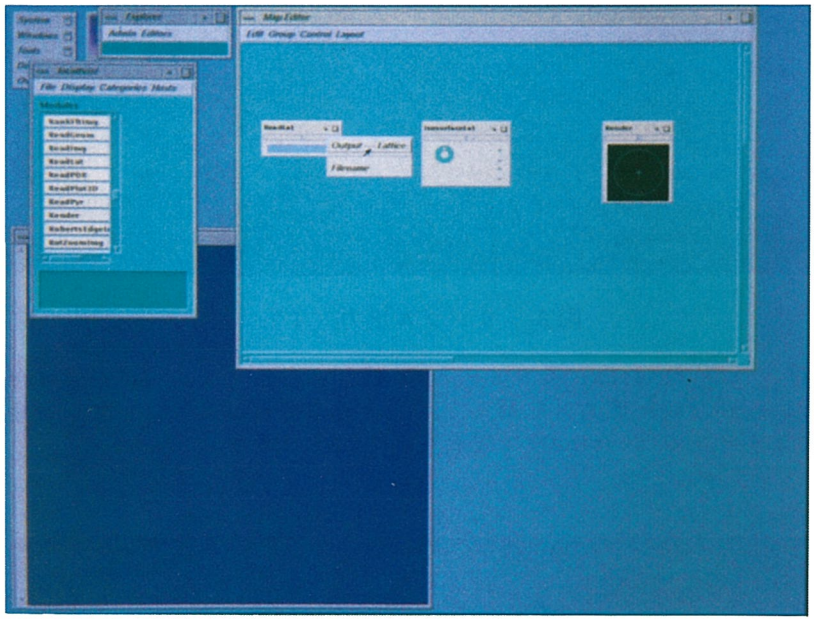


図２　モジュールの呼び出し

続いて、IsosurfaceLat の入力ポートパッドを右マウスボタンで押し、ポップアップメニューから Input--Lattice ポートを選択します。すると、ReadLat モジュールと IsosurfaceLat モジュールが接続されます。同様に、IsosurfaceLat の Surface--Geom ポートと Render の Input--Geom(Opt) ポートを接続し、この２つのモジュールを接続します。これで、格子データを読み込み、等値面を張って、表示するという可視化ネットワークのできあがりです。

5.2 モジュールパラメータの操作

図３は、各モジュールのフルスケールサイズのウィンドウを表示しているところです。ReadLat モジュールではファイルブラウザーで、入力ファイル名として、

```
/usr/explorer/data/lattice/testVol.lat
```

を与え、サンプルの格子データを読み込んでいます。
Render モジュールは、等値面を張ってできた物体を表示しています。

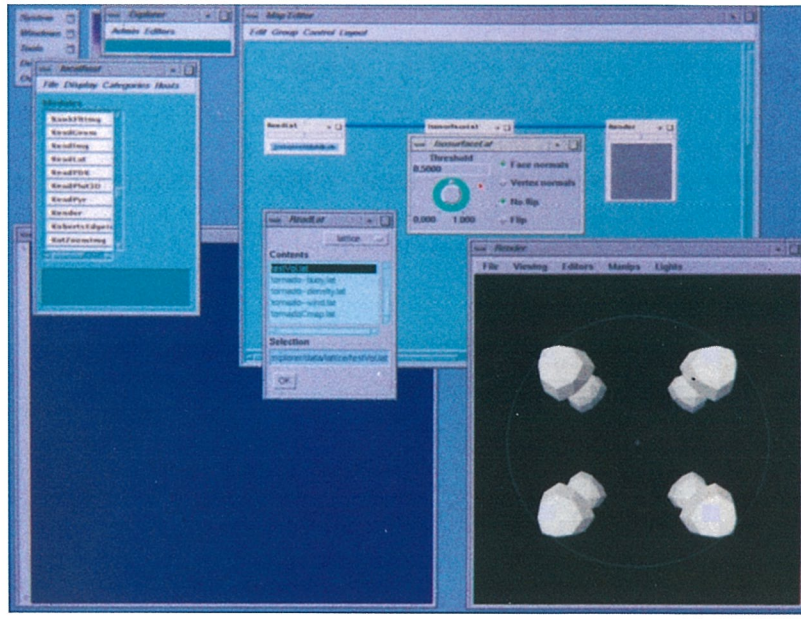


図３　サンプル出力（１）

図４では、IsosurfaceLat モジュールの Threshold パラメータをダイヤルで変更し、Vertex normals ボタンを選んで、等値面を滑らか表示にしています。また、Render モジュールの Examiner で視点を変更しています。

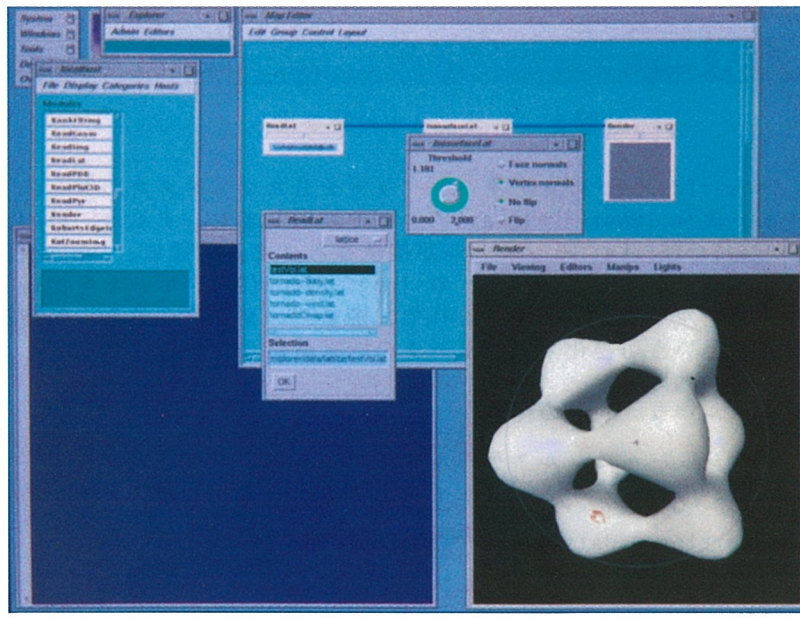


図４　サンプル出力（２）

5.3 Ｒｅｎｄｅｒモジュールの操作

図５では、Render モジュールで物体の材質感を変更しています。このために、まず、Render モジュールの Viewing メニューから Pick/Edit を選び、物体を選択します。次に、Editors メニューから Material Editor を選び、物体の透過度（Transp スライダー）を上げて等値面を半透明表示にし、Diff（散乱反射係数）の左の Edit Color ボタンを押して、カラーエディタを呼び出し、散乱反射の色（Material Diffuse Color）を変更しています。

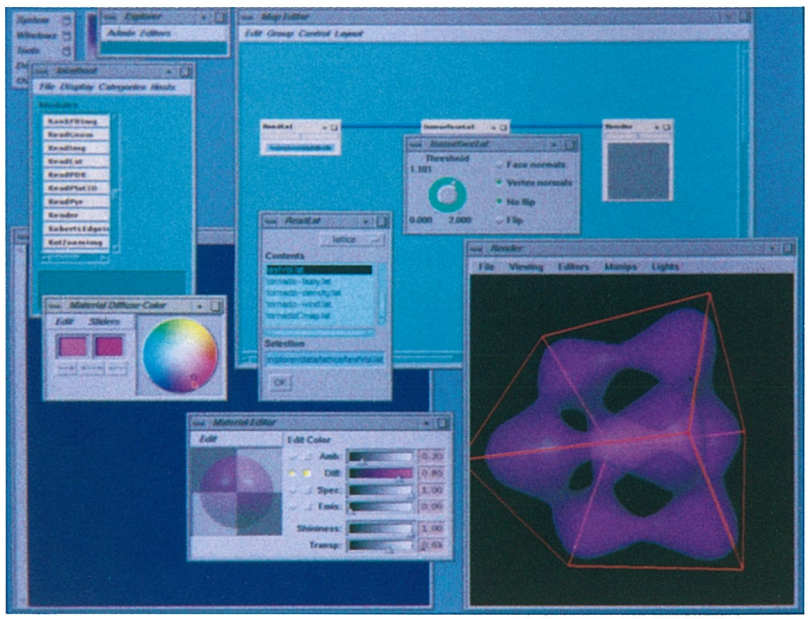


図５　サンプル出力（３）

Render モジュールには Viewing モードと Editing モードがあります。最初はデフォルトで Examiner Viewer の Viewing モードになっています。

Examiner モードでは、可視化された物体の回りを自由に移動し、任意の方向からその物体を見ることができます。可視化された物体がガラス球の中に入っていて、それを回しているような感覚です。物体が回転しているように見えますが、実は視点の方が動いています。

マウスのカーソルを球の中（画面では円の内側）、球の外（画面では円の外側）に持ってくると、カーソルがそれぞれ「手の形」、「回転矢印」になります。また、球の縁（画面では円周上）をつかんでドラッグすると、その球の半径が変化します。手の形のマウスカーソルをドラッグして動かすと、物体の回りを回転します。回転矢印の場合は、画面に垂直な軸の回りの回転となります。

マウスの中ボタンを押すと、物体がズームインして大きく表示され、<Alt>キーを押しながら中ボタンを押すと、物体がズームアウトして小さく表示されます。

Render のウィンドウの中でマウスの右ボタンを押すと、メニューがポップアップします。そのままドラッグして、View All を選ぶと、視点の位置が初期設定にもどります。これは、ズームし過ぎたりして、物体が見えなくなったときなどに便利です。

Editing モードでは、可視化された物体の位置や属性を変更できます。まず、

Viewing メニューから Pick/Edit を選び、操作対象の物体を左マウスボタンでクリックして選択します。使えるエディタ（Editors メニュー）には、

Material Editor　：　物体の材質感を変更する
Color Editor　：　物体の色（Diffuse Color）を変更する
Transform Sliders　：　物体の位置を変更（移動、回転、伸縮）する

があります。

Lights メニューを開けて操作すると、様々な光源の設定ができます。

DataScribe

1. はじめに

本稿では、Explorer のファイル入力モジュールの作り方を中心に述べます。スーパーコンピュータ等で計算した結果である数値データがファイルに入っている場合を想定しています。一般にデータフォーマットはそのプログラムに依存しているため、まずそれを Explorer 型のデータタイプに変換する必要があります。そこで、そのファイルを読み込み、Explorer 型のデータタイプに変換し、出力する Explorer モジュールを作成します。このモジュールの出力に種々のモジュールを接続することによって最終的に所望の可視化画像が得られるわけです。

詳細は、「IRIS Explorer User's Guide」の Chapter 5 をご覧ください。

2. 起動と終了

2.1 起動の仕方

DataScribe の起動は、シェルウィンドウで以下のようにタイプします。

```
% /usr/explorer/bin/dscribe
```

起動して、ウィンドウを置くと、パレット（DataScribe Palette）、データスクライブ（DataScribe）の２つのウィンドウが図１の様に画面に現れます。

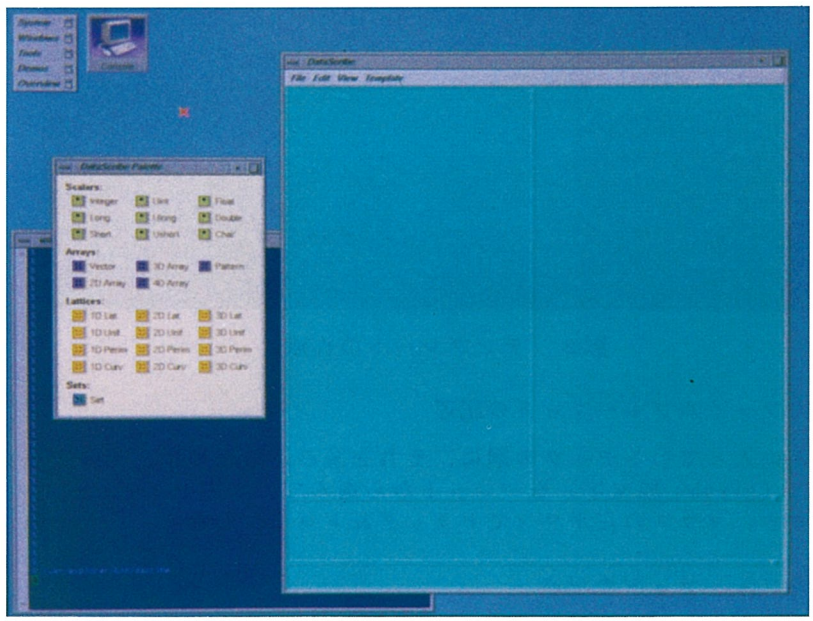


図１　Data Scribe の起動画面

2.2 終了の仕方

データスクライブウィンドウ の File メニューを開き、Quit を選択します。具体的には、それぞれマウスカーソルを File、Quit に合わせてマウスの左ボタンをクリックします。

3. ファイル入力モジュールの作り方

3.1 テンプレート作成

入力と出力のテンプレートをそれぞれ作成します。まず、データスクライブウィンドウの Template メニューから、Template を選ぶと Template Dialog ウィンドウが図2の様に表示されます。テンプレートの名前を Name の右側に設定します。そして、入力テンプレートの場合は、Direction を Input にし、ファイルフォーマットがアスキーかバイナリーに応じて Type を Ascii または Binary とします。また、出力テンプレートの場合は、Direction を Output にし、Type を Explorer にします。最後に OK をクリックします。すると、入力テンプレートは左側に、出力テンプレートは右側に、開かれた状態で置かれます。

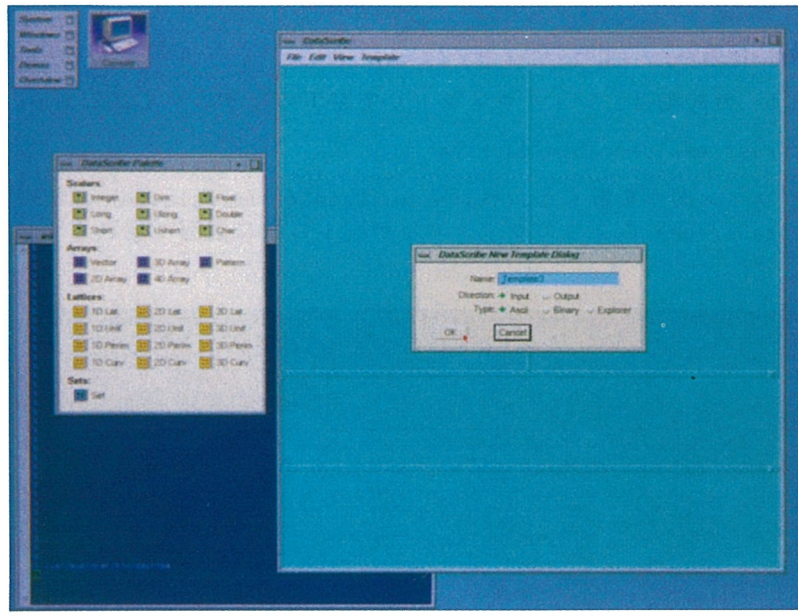


図2　テンプレートの作成

3.2 入力ファイルフォーマットの記述

ファイルに入っているデータの順に、それを読み込むために、そのデータタイプのグリフ（glyph：絵文字）をパレットから選んで、入力テンプレートの中に並べて行きます。マウスの左ボタンでドラッグ＆ドロップします。

3.3 出力フォーマットの記述

入力データの構造に応じた Explorer 型の格子データタイプを選び、そのグリフをパレットから出力テンプレートの中にマウスの左ボタンでドラッグ＆ドロップし、修正して行きます。例えば、格子点の要素データの次元（nDataVar）を設定する必要があります。

3.4 定数、パラメータの入力

定数を設定するときは定数テンプレートを使います。Template メニューから

Constants を選び、定数テンプレートを作成します。その中に、設定したいデータタイプのグリフ（スカラーまたは集合）をパレットから選んで並べ、値を設定して行きます。パラメータを設定するときはパラメータテンプレートを使います。Template メニューから Parameter を選び、パラメータテンプレートを作成します。その中に、設定したいスカラーデータタイプのグリフ（Long またはDouble）を選んで並べて行きます。できあがったモジュールにおいて、このパラメータは Control Panel Editor を用いて作ったコントロールパネルで操作することができます。

3.5 入出力の接続

入力データと出力データの対応付けを行ないます。まず、入力テンプレートの右上の長方形の上で、右マウスボタンを押し、プルダウン表示されるデータの中から今設定する入力データをドラッグして選びます。次に、出力テンプレートの左上の長方形の上でマウスボタンを押し、プルダウン表示されるデータの中から入力データに対応させるデータをドラッグして選びます。この操作を繰り返し、全ての必要なデータの対応付けを行ないます。定数テンプレートやパラメータテンプレートのデータも同様にして接続します。

3.6 保存と修正

できあがったモジュールは、File メニューから Parse を選ぶことにより、記述のチェックを行なうことができます。

完成したモジュールは File メニューから Save As を選び、ファイルブラウザーを呼び出して、所望のディレクトリに所望の名前を付けて保存します。このとき、その名前に .mres、.scribe、.help、.doc という拡張子が自動的に付加された４個のファイルが生成されます。Explorer でそのモジュールを呼び出す場合には、名前.mres を選びます。

保存されているモジュールを修正する場合は、File メニューから Open を選び、ファイルブラウザーを呼び出して、修正するモジュール（名前.scribe）を呼び出します。修正の後、Save を選び、再び保存します。

4. データタイプ

DataScribe が扱うデータは、スカラー（Scalars）、配列（Arrays）、格子（Explorer 型 Lattices）、集合（Sets）であり、グリフ（glyph：絵文字）と呼ばれるアイコンとしてパレットに整列しています。DataScribe は幾何学データやピラミッドデータなどの格子データ以外の Explorer 型データを扱うことはできません。また、条件分岐によってデータを選択したり、ファイルのデータを検索することもできません。

スカラーデータ
- 符号付き整数 ： Integer, Long, Short
- 符号無し整数 ： Uint, Ulong, Ushort
- 浮動小数点数 ： Float, Double
- 文字 ： Char

配列データ
- ベクトル ： Vector
- 配列 ： 2D Array, 3D Array, 4D Array
- パターン ： Pattern

格子データ
- 一般型 ： 1D Lat, 2D Lat, 3D Lat

定型　：　1D Uinf, 2D Unif, 3D Unif
直交型　：　1D Perim, 2D Perim, 3D Perim
不定型　：　1D Curv, 2D Curv, 3D Curv

集合データ
集合型　：　Set

それぞれのスカラーデータの値（ビット数など）は以下のようになっています。

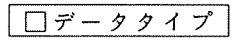
Integer, Long	：	３２ビット符号付き整数
Short	：	１６ビット符号付き整数
Uint, Ulong	：	３２ビット符号無し整数
Ushort	：	１６ビット符号無し整数
Float	：	３２ビットＩＥＥＥフォーマット
Double	：	６４ビットＩＥＥＥフォーマット
Char	：	ＡＳＣＩＩ文字（８ビット符号無し）

入力テンプレートの中でスカラーデータの値を設定することはできません。

配列データ（ベクトル、配列、パターン）は同種のスカラーや配列の集まりで、ベクトルは１次元配列に相当します。また、パターンはファイルの文字列パターンの検索に使います。配列データは、その配列の大きさを表すインデックスと要素のデータタイプを設定します。

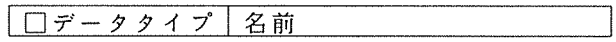
5．グリフ（glyph）

パレットに整列しているグリフは、

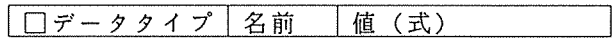


のようにデータタイプを表示しているだけですが、テンプレートに配置されたグリフは、デフォルトの名前が付き、その名前を変更することができます。

スカラーグリフの場合、

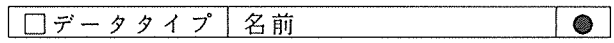


のようにデータタイプと名前が表示されます。また、定数テンプレートなどに配置されたスカラーグリフでは、



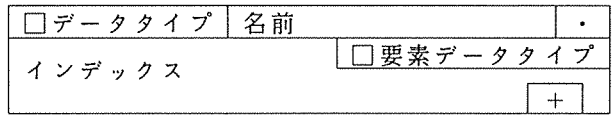
のように、その値を数値や式で与えることができます。

配列、格子、集合グリフの場合、



のように右端にボタンが付いています。この右丸ボタンをクリックすると、グリフが開き、より詳細な構造が表示され、再び右小丸ボタンをクリックすると、閉じて元に戻ります。

配列グリフの右丸をクリックすると、

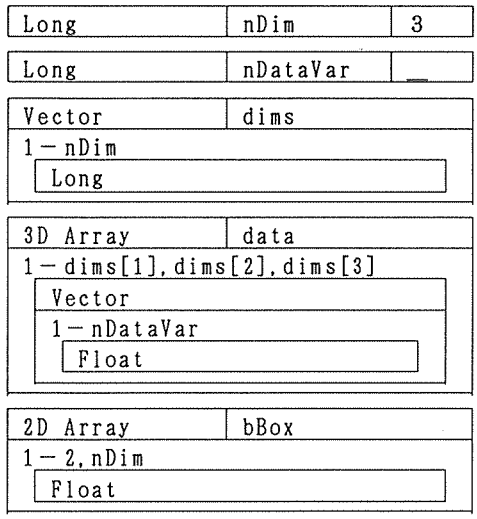


のように開きます。要素のデータタイプは、所望のデータタイプのグリフをパレットからそこにドラッグすることにより設定します。インデックスは、その配列の次元じ応じて、1－N、1－N1,N2、1－N1,N2,N3、1－N1,N2,N3,N4 とデフォルトでなっています。インデックスの設定は、各Nを定数や式で置き換えることによて行ないます。インデックスに関するさらに詳細な操作は、右下の十文字のアイコン Component Menu Button で行ないます。

5.1 Explorer 型格子

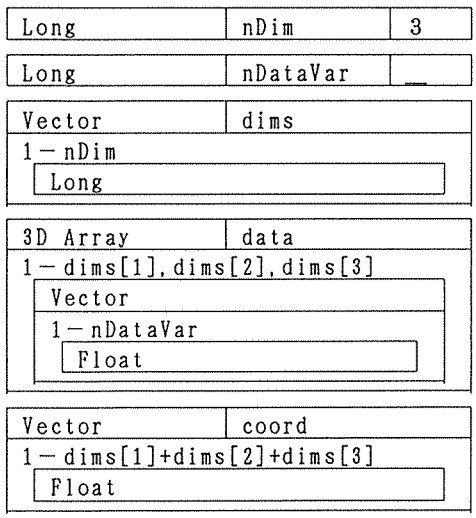
Explorer 型の格子データグリフはそれぞれ、複数のスカラーグリフや配列グリフから成り、予め定義されています。

3次元定型格子（3D Unif）では格子点が各軸方向に等間隔に並んでいます。そのグリフの中身は、



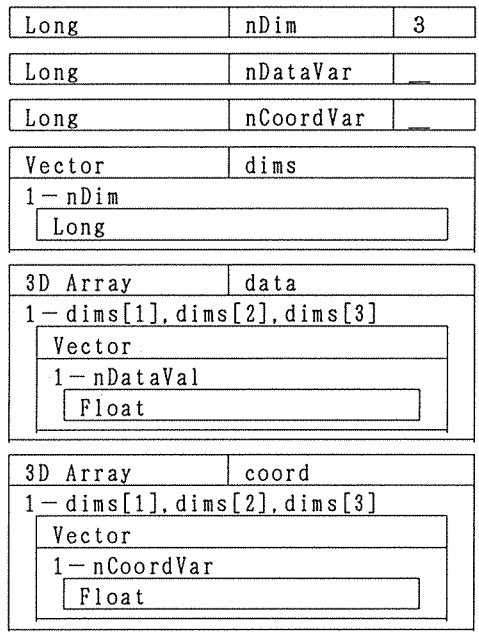
となっており、要素のデータの次元（nDataVarの値）を設定する必要があります。ファイルから読み込んだ3次元の要素データを data に渡します。

3次元直交型格子（3D Perim）では格子点の各軸方向の間隔が均等でありません。そのグリフの中身は、



となっており、要素のデータの次元（nDataVarの値）を設定する必要があります。格子点のｘ、ｙ、ｚ座標はその順番に並べて coord に渡し、３次元の要素データは data に渡します。

３次元不定型格子（3D Curvi）では格子点が整列しておらず、各格子点は座標データを持っています。そのグリフの中身は、



となっており、要素のデータの次元（nDataVarの値）を設定する必要があります。３次元の格子点の座標（nCoordVar 次元）を coord に渡し、３次元の要素データを data に渡します。

６．Ｆｏｒｔｒａｎのバイナリーデータファイルについて

Ｆｏｒｔｒａｎプログラムで作成したバイナリーデータファイルでは、Ｃで作成したバイナリーデータファイルと異なり、そのレコード構造を表現するために、各レコードデータの前後にそのレコード長を表現する（余分な）データが付加されています。DataScribe でバイナリーデータのファイル入力モジュールを作成する際には、この余分なデータを読み飛ばす必要があります。

いくつかのＷＳで調べたところ、TITAN の場合は、ワード数（４バイト単位）、SUN、EWS、IRIS の場合は、バイト数でレコード長を表現しており、互換性はありませんが、いずれも３２ビット整数で表現しています。このため、DataScribe の入力テンプレートにおいて、各レコードデータの読み込みに対応している部分の前後に Integer 型のダミーデータを読み飛ばすためのグリフを挿入すればよいことになります。

参考文献

１．IRIS Explorer Technical Report, SiliconGraphics Inc., 1992.
２．IRIS Explorer User's Guide, Silicon Graphics Inc., 1992.
３．IRIS Explorer Module Definitions, Silicon Graphics Inc., 1992.
４．IRIS Explorer Module Writer's Guide, Silicon Graphics Inc., 1992.