OCTA

ソフトマテリアルのための統合化シミュレータ

グラフィカルユーザーインターフェース GOURMET 操作マニュアル Version 4.1.3

> OCTAユーザーズグループ Aug. 08 2010

執筆者

第1章	土井正男
第2章以降	西尾裕三
改訂	西本正博、小沢拓

プログラム開発者

プログラム設計	芝野真次、西尾裕三
プログラム開発	西本正博、庄野敦子、橋本寛、乾吉治、増井郁久、
	稗島潤一郎、奥野幸洋、田子森裕香、奥田俊樹、西康之
ソフトウェアテスト	西谷英輔、西尾裕三、堀川貴子

謝辞

本プログラム開発の大半は、経済産業省の出資・補助を受け、新エネルギー・産業技術総合開発 機構(NEDO)が(財)化学技術戦略推進機構に委託した、大学連携型産業科学技術研究開発プロジェク ト「高機能材料設計プラットフォーム」通称「土井プロジェクト」の下で行われたものである。

本プログラムの改良は、独立行政法人 科学技術振興機構(旧称 特殊法人 科学技術振興事業団) の補助を受け、「多階層的バイオレオシミュレータの研究開発」における「バイオレオシミュレー タ用プラットフォーム」機能改良ソフト開発の下で行われたものである。

Copyright ©2000-2010 OCTA Licensing Committee All rights reserved.

目次

第1章 GOURMETとは	1
第2章 最初にすること	3
2.1 GOURMETを起動します	3
2.2 UDFを編集します	4
2.3 Pythonを走らせましょう	5
2.4 エンジンを動かしましょう	6
2.5 結果を見ましょう	8
2.6 ツールを使いましょう	10
第3章 起動時のオプション	12
3.1 システムの概要	12
3.2 環境変数	13
3.2.1 必須の環境変数	13
3.2.2 オプションの環境変数	13
3.3 起動用のシェルスクリプト	15
3.3.1 起動オプション	15
3.3.2 Microsoft Windows	15
3. 3. 3 Linux	16
3.4 エンジンマネージャ	16
3.4.1 起動オプション	16
3 4 2 Microsoft Windows	17
3 4 3 Linux Ellnix	17
3.5 データマネージャ	
3 5 1 Microsoft Windows	17
3.5.2 Linux & Unix	17
3.6 GOURMETの終了	
第4章 IDFを編集するには.	
4.1 編集モード	
4.1.1 編集モード	
4.2 ビュー形式の選択	
4.2.1 Treeビュー	
4.2.2 Tableビュー	
4.3 データロケーションの選択	
4.3.1 Globalロケーション	
4.3.2 Record ロケーション	
4.4 File メニュー	
4.4.1 UDFヘッダーを編集するには	
4.4.2 ファイルコンバータの使い方	
4.4.3 テキスト形式UDFとバイナリ形式UDFの使い方	
4.5 Edit メニュー	
4.5.1 Copy/Pasteモードの使い分け	
4.6 Viewメニュー	
4.6.1 エディタのPreferencesダイアログ	
4.7 単位変換	
4.8 エディタ使用のTips	
第5章 単位系の使い方	
5.1 Unit メニュー	36

5.2 単位系選択の使い方 56 5.3 単位系のクレボート方法 57 5.4 エンジン単位系の参照 38 75 4 エンジン単位系の参照 75 6.1 Python Apu / トを使うには 39 6.1 Python Panel のツールとPythonメクリプト契行 40 6.3 ビューワにおけるPythonスクリプト処理 41 6.4 エデイタにおけるアクション 42 6.5 E ビューワにおけるアクション 42 6.6 Pythonパネルの利用に関するTips 44 第7章 エンジンを実行するには 46 7.1 エンジンキメージャ 46 7.2 Engine Run'ネル 46 7.3 Engine Control.パネル 48 7.4 エンジンシー制御に関するTips 49 第8章 30オブジェクトウインドウ 50 8.2 130オブジェクトウインドウ 50 8.2 130オブジェクトのビッキング 51 8.3 ビューワのPythonウインドウ 51 8.3 ビューワのPythonウインドウ 52 8.4 ビューワのシュー 52 8.4 ビューワのメニュー 52 8.4 U ロックション 52 8.4 U ロックション 52 8.4 U ロックション 52 8.4 1 Fileメニュー 53 8.4 1 Pithonメニュー 53 8.4 2 U vexメニュー 53 8.4 2 Python ウィント		
5.3 単位系のインボート方法	5.2 単位系選択の使い方	
5.4 $z > 2 \lor 2 \lor 4 (\mathcal{R} \circ \mathcal{R} \otimes \mathbb{R})$ 38 6.6 p (thon $A \circ J \cup J \lor ket(5)$ (ct.) 39 6.1 p (thon $A \circ J \cup J \lor ket(5)$ (ct.) 40 6.2 $z = \neg U = 1 \lor 1 \lor 3 \lor J \lor 1 \lor 1 \lor 2 \lor 2 \lor 1$ 41 6.4 $z = \neg U = 1 \lor 1 \lor 3 \lor 1 \lor 1 \lor 2 \lor 2$	5.3 単位系のインポート方法	
第6章 Pythonスクリプトを使うには	5.4 エンジン単位系の参照	
6.1 Python PanelのツールとPythonメニュー 40 6.2 エディタにおけるPythonスクリプト実行 40 6.3 ビューワにおけるPythonスクリプト処理 41 6.4 エディタにおけるPythonスクリプト 43 6.5 ビューワにおけるPythonスクリプト 43 6.6 Pythonパネルの利用に関するTips 44 第7章 エンジンを実行するには 46 7.1 エンジンを実行するには 46 7.2 Engine Runパネル 46 7.3 Engine Controlパネル 48 7.4 エンジンマネージャ 46 7.3 Engine Controlパネル 48 8.2 3Dオブジェクトを表示するには 50 8.1 ビューワの起動画面 50 8.2 3Dオブジェクトウインドウ 50 8.2 3Dオブジェクトのビッキング 51 8.3 ビューワのシニュー 52 8.4 ビューワのシニュー 52 8.4 ビューワのシニュー 52 8.4 ビューワのシニュー 52 8.4 ジョンオジェクトのビュンション 52 8.4 ジョンオジェクトのアニュー 52 8.4 ジョンカージェクシーシーシーション 52 8.4 ジョンオジェクトのブェー 52 8.4 ジョンオジェクトのブェー 53 8.4 ジョンオジェクトのブェー 54 9.2 ジョンスシーシン 55 8.4.5 Optinsメニュー 57 8.4.6 Opti	第6章 Pythonスクリプトを使うには	39
6.2 エディタにおけるPythonスクリプト契行 .40 6.3 ビューワにおけるアクション .42 6.5 ビューワにおけるアクション .42 6.6 Pythonパネルの利用に関するTips .44 第7章 エンジンを実行するには .46 7.1 エンジンマネージャ .46 7.2 Engine Rumパネル .46 7.3 Engine Controlパネル .48 7.4 エンジン制算していたいパネル .46 7.3 Engine Controlパネル .48 7.4 エンジン制算していたいパネル .46 7.4 エンジン制算していたいパネル .46 7.4 エンジン制算していたいパネル .48 7.4 エンジン制算していたいたいたいたいたいたいたいたいたいたいたいたいたいたいたいたいたいたいた	6.1 Python PanelのツールとPythonメニュー	
6.3 ビューワにおけるPythonスクリプト処理	6.2 エディタにおけるPythonスクリプト実行	
6.4 $x = y + y + 2 + 3 + 3 + 5 + 5 + 5 + 5 + 5 + 5 + 5 + 5$	6.3 ビューワにおけるPythonスクリプト処理	
6.5 ビューワにおけるアクション 43 6.6 Pythonパネルの利用に関するTips 44 7.1 エンジンを実行するには 46 7.1 エンジンマネージャ 46 7.2 Engine Runパネル 48 7.4 エンジンを実行するには 46 7.1 エンジンマネージャ 46 7.2 Engine Controlパネル 48 7.4 エンジン制御に関するTips 49 第8 第 3Dオブジェクトを表示するには 50 8.1 ビューワの起動画面 50 8.2.1 3Dオブジェクトのセットウ 50 8.2.1 3Dオブジェクトのアンドウ 51 8.3 ビューワのPythonウインドウ 51 8.3.1 ジェクトのアニメーション 52 8.4.1 File/sュー 52 8.4.2 Viewメニュー 52 8.4.3 Displayメニュー 53 8.4.4 Pito/シュュー 55 8.4.4 Pito/sメニュー 57 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Pito/y・ク 63 9.1 Plot ケリール 61<	6.4 エディタにおけるアクション	
6.6 Pythonパネルの利用に関するTips 44 第7章 エンジンを実行するには 46 7.1 エンジンマネージャ 46 7.2 Engine Runパネル 46 7.3 Engine Controlパネル 48 7.4 エンジン制御に関するTips 49 第8章 3Dオブジェクトを表示するには 50 8.1 ビューワの起動画面 50 8.2 3Dオブジェクトのとジャキング 50 8.3 ビューワのPythonウインドウ 50 8.3 ビューワのPythonウインドウ 51 8.3 エクロのPythonウインドウ 52 8.4 ビューワのPythonウインドウ 52 8.4 ビューワのPythonウインドウ 52 8.4 ビューワのPythonウインドウ 52 8.4 1 Fileメニュー 52 8.4 2 Niewメニュー 52 8.4.1 Fileメニュー 52 8.4.2 Niewメニュー 53 8.4.3 Displayメニュー 56 8.4.4 Pickingメニュー 57 8.9章 プロットを行うには 61 9.1 Plotサイブランクト 62 9.2 Graph sheetオブジェクト 62 9.2 Graph sheetオブジェクト 62 9.2 I GraphSheet操作を行うPythonスクリブトの例 62 9.3 Plotスクリブティング 63 9.4 Pothon マール 66 10 章 ツールを使うには	6.5 ビューワにおけるアクション	
第7章 エンジンを実行するには 46 7.1 エンジンマネージャ 46 7.2 Engine Runパネル 46 7.3 Engine Controlパネル 48 7.4 エンジン制御に関するTips 49 第8章 3Dオブジェクトを表示するには 50 8.1 ビューワの短動画面 50 8.2 3Dオブジェクトウセンドウ 50 8.3 ビューワの欠けのウインドウ 51 8.3 エコワのアけのウインドウ 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4.2 Viewメニュー 52 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 55 8.4.4 Pickingメニュー 56 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Opthon ジェュー 57 8.4.8 Displayメニュー 56 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Opthon ジェクト 62 9.2 Graph sheetオブジェクト 62 9.2.1 GraphSheet操作を行うPythonスクリブトの例 62 9.2 Opthon ツール 66 10.1 フェノル転送ツール 66 10.2 Python ツール 68	6.6 Pythonパネルの利用に関するTips	
7.1 エンジンマネージャ 46 7.2 Engine Run ³ ネル 46 7.3 Engine Controlパネル 48 7.4 エンジン制御に関するTips 49 第 8 章 3Dオブジェクトを表示するには 50 8.1 ビューワの起動画面 50 8.2 3Dオブジェクトウインドウ 50 8.3 ビューワのPthonウインドウ 51 8.3 ビューワのPthonウインドウ 51 8.3.1 3Dオブジェクトのアニメーション 52 8.4 ビューワのスニュー 52 8.4 ビューワのスニュー 52 8.4.1 Fileメニュー 52 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 55 8.4.4 Pickingメニュー 55 8.4.5 Pythonメニュー 56 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 57 9.9 プレットを行うには 61 9.1 Plotツール 62 9.3 1 Plotライブジェクト 62 9.3 1 Plotライブラリの使使っには 66 10.1 ファイル転送ツール 67 10.2 Pythonツール 68 10.5 分子ビルグーツール 68 10.5 分子ビルグーツール 68 10.5 分子ビルグーツール 68 10.5 分	第7章 エンジンを実行するには	
7.2 Engine Run/ 2 μ . 46 7.3 Engine Control 1 2 μ μ . 48 7.4 x y y h for 2 2 μ b 2 2 2 2 49 $\hat{\mathbf{x}}$ 8 $\hat{\mathbf{x}}$ 3D 2 2 y b 2 2 2 2 2 50 8.1 U_{2} $ ^{0}$ D 2 2 2 2 2 2 2 2 50 8.1 U_{2} $ ^{0}$ 2 2 2 2 2 2 2 2 50 8.2 1 30 2 2 2 2 2 2 2 2 2 51 8.3 U_{2} $ ^{0}$ 2 2 2 2 2 2 2 51 8.3 U_{2} $ ^{0}$ 2 2 2 2 2 2 2 2 51 8.3 U_{2} $ ^{0}$ 2 2 2 2 2 2 2 52 8.4 U_{1} V_{1} 2 2 2 2 2 2 2 2 52 8.4.1 V_{1} V_{1}^{2} 2 52 8.4.2 V_{1} V_{2}^{2} 52 8.4.3 U_{1} V_{1} V_{2}^{2} 53 8.4.4 V_{1} V_{1} V_{2}^{2} 55 8.4.4 V_{1} V_{1} V_{1} 2 2 55 8.4.5 V_{1} V_{1} V_{1} 2 2 57 8.4.6 V_{1} V_{1} V_{1} 2 2 57 8.4.6 V_{1} V_{1} V_{1} 2 57 9.2 U_{1} V_{1} $V_$	7.1 エンジンマネージャ	
7.3 Engine Control $\sqrt[3]{2}\mu$ 48 7.4 $x = \sqrt[3]{2}\sqrt[3]{2}x \int k_{2} \sqrt[3]{2}x \int k$	7.2 Engine Runパネル	
7.4 エンジン制御に関するTips. 49 第8章 30オブジェクトを表示するには 50 8.1 ビューワの起動面面 50 8.2 30オブジェクトウセンドウ 50 8.2.1 3Dオブジェクトのビッキング 51 8.3 ビューワのPythonウインドウ 51 8.3 ビューワのPythonウインドウ 52 8.4.1 Fileメニュー 52 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 53 8.4.4 Pickingメニュー 56 8.4.4 Pickingメニュー 56 8.4.4 Pickingメニュー 56 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Dispheret操作を行うPythonスクリプトの例 62 9.2 Graph sheetオブジェクト 62 9.2.1 GraphSheet操作を行うPythonスクリプトの例 62 9.3 Plotスクリプティング 63 9.3 Plotスクリプティング 63 9.3 Plotスクリプティング 63 9.3 Plotスクリプティング 63 9.3 Plotスクリプティング 64 第10 章 ツールを使うこは 66 10.5 分子ビル・ダー	7.3 Engine Controlパネル	
第8章 3Dオブジェクトを表示するには 50 8.1 ビューワの起動画面 50 8.2 3Dオブジェクトウインドウ 50 8.3 ビューワのPytonウインドウ 51 8.3 ビューワのPytonウインドウ 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4.1 Fileメニュー 52 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 56 8.4.3 Displayメニュー 56 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Pythonメニュー 57 8.4.8 Spthonメニュー 57 8.9 デ Tロットを行うには 61 9.1 Plotマイン 62 9.2.1 GraphSheet操作を行うりやれの 62 9.2 TaphSheet操作を行うやしゃの 57 63 64 <td>7.4 エンジン制御に関するTips</td> <td></td>	7.4 エンジン制御に関するTips	
8.1 ビューワの起動画面 50 8.2 30オブジェクトウインドウ 50 8.2 1 3Dオブジェクトウビッキング 51 8.3 ビューワのPytonウインドウ 51 8.3 ビューワのサニュー 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4.1 Fileメニュー 52 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 55 8.4.4 Pickingメニュー 56 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Displayビュー 61 9.1 Plotツール 61 9.2 Graph sheetオブジェクト 62 9.3 Ilotスタリブティング 63 9.3 Ilotスタリブティング 63 9.3 Ilotマイブラリの使い方 64 第10 章 ツールを使うには 66 10.1 ファイル転送ツール 67 10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルグーツール 68 10.6 起動環境設定ツール 69 10.6 超動環境設定ツール 68 10.6 超動環境設定ツール 69 竹塚A 和pplication Setupツール 68 10.6 超動環境設定ツー	第8章 3Dオブジェクトを表示するには	50
8.2 3Dオブジェクトウインドウ 50 8.2 1 3Dオブジェクトウピッキング 51 8.3 ビューワのPythonウインドウ 51 8.3 1 3Dオブジェクトのアニメーション 52 8.4 ビューワのメニュー 52 8.4 1 Fileメニュー 52 8.4.1 Fileメニュー 52 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 55 8.4.4 Pickingメニュー 56 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 57 第 9章 プロットを行うには 61 9.1 Plotツール 61 9.2 Graph Sheetオブジェクト 62 9.3 1 Plotライブラリの使い方 63 9.3 1 Plotライブラリの使い方 64 第10章 ツールを使うには 66 10.1 ファイル転送ツール 67 10.3 Gnuplot ツール 67 10.3 Gnuplot ツール 68 10.5 分子ビルダーツール 68 10.6 を取り院電設ツール 68 10.7 ケビルダーツール 68 10.6 を取り完全地のシール 68 10.7 ケビルダーツール 68 10.5 分子ビルダーツール 68 10.5 分子ビルダーツール 68 10.5 分子ビルダーツール <td>8.1 ビューワの起動画面</td> <td></td>	8.1 ビューワの起動画面	
8.2.1 3Dオブジェクトのビッキング 51 8.3 ビューワのPythonウインドウ 51 8.3 ビューワのPythonウインドウ 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4 ビューマのメニュー 53 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 55 8.4.4 Pickingメニュー 56 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.9章 プロットを行うには 61 9.1 Plotツール 61 9.2 Graph sheetオブジェクト 62 9.3.1 Plotライブラリの使い方 63 9.3.1 Plotライブラリの使い方 64 第10章 ツールを使うには 66 10.1 ファイル転送ツール 67 10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環環旋定ツール 69 10.7 イル転送水グリール 68 10.6 左動環環旋定ツール 69 10.7 クリングメーシー 63 10.6 左動環環旋定ツール 68 10.7 クリングメーシー 63 10.7 クリングメンジャー	8 2 3Dオブジェクトウインドウ	50
8.3 ビューワのPythonウインドウ 51 8.3 1 3Dオブジェクトのアニメーション 52 8.4 ビューワのメニュー 52 8.4 ビューワのメニュー 52 8.4.1 Fileメニュー 52 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 55 8.4.4 Pickingメニュー 55 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.7 Pythonメニュー 57 8.4.8 Display 57 8.9 プロットを行うには 61 9.2 Graph sheetオブジェクト 62 9.3.1 Plotマイング 63 9.3.1 Plotマイング 63 9.3.1 Plotマイング 66 10.5 プラビングール 66 10.5 プラビングール 66 10.5 プラビングール 68 10.6 起動環境設定ツール 68 10.6 起動環境設定ツール	821 3Dオブジェクトのピッキング	
c. b 2 エ つり (b 1) (b 1) (b 1) (b 1) (b 2) (b 2) 52 8. 3.1 3D オブジェクトのアニメーション 52 8. 4 ビューワのメニュー 52 8. 4.1 Fileメニュー 52 8. 4.2 Viewメニュー 53 8. 4.3 Displayメニュー 53 8. 4.3 Displayメニュー 55 8. 4.4 Pickingメニュー 56 8. 4.5 Pythonメニュー 57 8.4.6 Optionメニュー 57 8.4.6 Optionメニュー 57 8.4.6 Optionメニュー 57 8.4.7 Pythonメニュー 57 8.4.6 Optionメニュー 57 8.4.6 Optionメニュー 57 8.4.6 Optionメニュー 57 8.4.7 Pythonメニュー 57 8.4.6 Optionメニュー 57 9 第 プロットを行うには 61 9.1 Plotツール 62 9.3.1 Plotライブラリの使い方 63 9.3.1 Plotライブラリの使い方 64 第 10 章 ツールを使うには 66 10.1 ファイル転送ツール 66 10.2 Pythonツール 66 10.3 Gruplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69	83 ビューワのPvthonウインドウ	51
8.4 ビューワのメニュー 51 8.4 1 Fileメニュー 52 8.4.1 Fileメニュー 52 8.4.2 Viewメニュー 53 8.4.3 Displayメニュー 55 8.4.4 Pickingメニュー 55 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 61 9.1 Plotツール 61 9.2 Graph Sheetオブジェクト 62 9.3.1 Plotフリプティング 63 9.3.1 Plotライブラリの使い方 64 第 10 章 ツールを使うには 66 10.1 ファイル転送ツール 66 10.2 Pythonツール 67 10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 68 10.6 超動環境設定ツール 69 付録A 稼働環境 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 72 A.4 Python System 73 74 Fide: アクションの定義 76	8313D	
0.7 4 CL 000000000000000000000000000000000000	$8.4 \text{ H}_{3} - 70 \text{ H}_{3} - 7$	
0.4.1 Price レーン 52 8.4.2 View メニュー 53 8.4.3 Display メニュー 55 8.4.4 Picking メニュー 56 8.4.5 Python メニュー 57 8.4.6 Options メニュー 57 8.4.7 Prove を行うには 61 9.1 Plot ツール 61 9.2 Graph Sheet オブジェクト 62 9.2.1 GraphSheet操作を行うPython スクリプトの例 62 9.3 Plot スクリプティング 63 9.3.1 Plot ライブラリの使い方 63 9.3.1 Plot ライブラリの使い方 64 第 10 章 ツールを使うには 66 10.1 ファイル転送ツール 66 10.2 Pythonツール 67 10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69 付縁A 稼働環境 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 73 イ 4 Python System 73 <td>8 4 1 Filox</td> <td></td>	8 4 1 Filox	
0.4.2 VIEW / ーユ 55 8.4.3 Display メニュー 55 8.4.4 Picking メニュー 56 8.4.5 Python メニュー 57 8.4.6 Options メニュー 57 9章 プロットを行うには 61 9.1 Plotツール 61 9.2 Graph Sheet オブジェクト 62 9.3 Plot スクリプティング 62 9.3.1 Plotライブラリの使い方 63 9.3.1 Plotライブラリの使い方 64 第 10 章 ツールを使うには 66 10.1 ファイル転送ツール 66 10.2 Pythonツール 67 10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 68 10.6 起動環境設定ツール 72 A.1 Operating System 72 A.3 Graphics System 72 A.3 Graphics System 72 A.4 Python System 73 fd録B. エンジンマネージャの定義・ので対・ロックー 72	$\begin{array}{c} 0.4.1 \text{ File} \\ \overline{} \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ $	
8.4.3 Display/-ユニー 56 8.4.4 Pickingメニュー 56 8.4.5 Pythonメニュー 57 8.4.6 Optionsメニュー 57 8.4.6 Optionsメニュー 61 9.1 Plotツール 61 9.2 Graph sheetオブジェクト 62 9.2.1 GraphSheet操作を行うPythonスクリプトの例 62 9.3 Plotスクリプティング 63 9.3.1 Plotライブラリの使い方 63 9.3.1 Plotライブラリの使い方 64 第 10章 ツールを使うには 66 10.1 ファイル転送ツール 66 10.2 Pythonツール 66 10.3 Gnuplot ツール 67 10.3 Gnuplot ツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69 付録A 稼働環境 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 76	$8.4.2 \text{ View} = 2^{-1} \dots = 2^{-1}$	
8. 4.4 Pitcking > - ユ ⁻ 50 8. 4.5 Python × ニュー 57 8. 4.6 Options × ニュー 57 9 章 プロットを行うには 61 9.1 Plot ツール 61 9.2 Graph sheet オブジェクト 62 9.3 Plot スクリプティング 63 9.3.1 Plot ライブラリの使い方 63 9.3.1 Plot ライブラリの使い方 64 第 10 章 ツールを使うには 66 10.1 ファイル転送ツール 66 10.2 Python ツール 66 10.3 Gnuplot ツール 67 10.3 Gnuplot ツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69 付録A 稼働環境 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 72	0. 4. 5 DIsplay > - 2	
8.4.5 Python $\lambda - 2^{-1}$ 57 8.4.6 Options $\lambda = 2^{-1}$ 57 第 9 章 プロットを行うには. 61 9.1 Plot $\mathcal{V} - \mathcal{N}$ 61 9.2 Graph Sheet $\lambda \vec{\tau} \vec{y} x \rho$ ト 62 9.2.1 GraphSheet操作を行うPython $\lambda \rho$ リプトの例. 62 9.3 Plot $\lambda \rho$ リプティング. 63 9.3.1 Plot $\vec{\tau} \vec{\tau} \vec{\tau} \rho$ 64 第 10 章 ツールを使うには. 66 10.1 ファイル転送ツール 66 10.2 Python $\mathcal{Y} - \mathcal{N}$ 66 10.3 Gnuplot $\mathcal{Y} - \mathcal{N}$ 68 10.4 Application Setup $\mathcal{Y} - \mathcal{N}$ 68 10.5 $\beta \vec{\tau} \vec{\tau} \mathcal{N} \vec{y} - \mathcal{Y} - \mathcal{N}$ 68 10.6 起動環境設定 $\mathcal{Y} - \mathcal{N}$ 69 付録A 稼働環境. 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 72 A.4 Python System 73 $\vec{\tau} \vec{x} \cdot \vec{y} \cdot \vec{y} - \vec{y} \cdot \vec{y} - \vec{z} + \vec{y} \cdot \vec{z} + \vec{y} - \vec{z} + \vec{y} \cdot \vec{z} + \vec{y} - \vec{z} + \vec{y} + \vec{z} $	8. 4. 4 Picking >	
8.4.6 0pt10nsメニュー 57 第9章 プロットを行うには. 61 9.1 Plotツール 61 9.2 Graph Sheetオブジェクト 62 9.2.1 GraphSheet操作を行うPythonスクリプトの例. 62 9.3 Plotスクリプティング. 63 9.3.1 Plotライブラリの使い方. 64 第 10章 ツールを使うには. 66 10.1 ファイル転送ツール 66 10.2 Pythonツール 67 10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 68 10.6 起動環境設に 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 75	8.4.5 Python >	
第9章 プロットを行うには	8.4.6 Uptionsメニュー	
9.1 Plot ツール 61 9.2 Graph sheet オブジェクト 62 9.2.1 GraphSheet操作を行うPythonスクリプトの例 62 9.3 Plot スクリプティング 63 9.3.1 Plotライブラリの使い方 63 9.3.1 Plotライブラリの使い方 64 第 10 章 ツールを使うには 66 10.1 ファイル転送ツール 66 10.2 Pythonツール 67 10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69 付録A 稼働環境 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 76	男9早 ノロツトを行りには	
9.2 Graph sheetオフシェクト	9.1 Plot $\mathcal{Y} - \mathcal{V}$	
9.2.1 GraphSheet操作を行うPythonスクリプトの例	9.2 Graph sheetオフジェクト	
9.3 Plotスクリフティング639.3.1 Plotライブラリの使い方64第 10 章 ツールを使うには6610.1 ファイル転送ツール6610.2 Pythonツール6710.3 Gnuplot ツール6710.3 Gnuplot ツール6810.5 分子ビルダーツール6810.6 起動環境設定ツール69付録A 稼働環境72A.1 Operating System72A.2 Java System72A.3 Graphics System72A.4 Python System73付録B. エンジンマネージャのセキュリティーポリシーについて74付録C. アクションの定義76	9.2.1 GraphSheet操作を行うPythonスクリプトの例	
9.3.1 Plot ライブラリの使い方64第 10 章 ツールを使うには.6610.1 ファイル転送ツール6610.2 Pythonツール6710.3 Gnuplot ツール6810.4 Application Setupツール6810.5 分子ビルダーツール6810.6 起動環境設定ツール69付録A 稼働環境.72A.1 Operating System72A.2 Java System72A.3 Graphics System72A.4 Python System73付録B. エンジンマネージャのセキュリティーポリシーについて74付録C. アクションの定義.76	9.3 Plotスクリプティング	63
第 10 章 ツールを使うには	9.3.1 Plotライブラリの使い方	64
10.1 ファイル転送ツール6610.2 Pythonツール6710.3 Gnuplot ツール6810.4 Application Setupツール6810.5 分子ビルダーツール6810.6 起動環境設定ツール69付録A 稼働環境72A.1 Operating System72A.2 Java System72A.3 Graphics System72A.4 Python System73付録B. エンジンマネージャのセキュリティーポリシーについて74付録C. アクションの定義760.1 スカンシンマネーグ76	第10章 ツールを使うには	
10.2 Pythonツール 67 10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69 付録A 稼働環境 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 76	10.1 ファイル転送ツール	
10.3 Gnuplot ツール 68 10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69 付録A 稼働環境 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 76	10.2 Pythonツール	67
10.4 Application Setupツール 68 10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69 付録A 稼働環境. 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義. 76	10.3 Gnuplot ツール	
10.5 分子ビルダーツール 68 10.6 起動環境設定ツール 69 付録A 稼働環境. 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義. 76 0.1 スカンシンマネージャのセキュリティーポリシーについて 76	10.4 Application Setupツール	
10.6 起動環境設定ツール 69 付録A 稼働環境. 72 A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義. 76 0.1 スクションの定義. 76	10.5 分子ビルダーツール	
付録A 稼働環境. 72 A. 1 Operating System 72 A. 2 Java System 72 A. 3 Graphics System 72 A. 4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義. 76 77 77 78 76 79 77 74 74 75 74 75 74	10.6 起動環境設定ツール	
A.1 Operating System 72 A.2 Java System 72 A.3 Graphics System 72 A.4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 76 0.1 スカンシンマネージャのセキュリティーポリシーについて 76	付録A 稼働環境	
A. 2 Java System 72 A. 3 Graphics System 72 A. 4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 76 71 76	A.1 Operating System	
A. 3 Graphics System	A.2 Java System	
A. 4 Python System 73 付録B. エンジンマネージャのセキュリティーポリシーについて 74 付録C. アクションの定義 76 76 76	A.3 Graphics System	
付録B. エンジンマネージャのセキュリティーポリシーについて	A.4 Python System	
付録C. アクションの定義	付録B. エンジンマネージャのセキュリティーポリシーについて	
	付録C. アクションの定義	
- 6.1 ノクンヨンノアイルの又伝	C.1 アクションファイルの文法	

C.2 Python文の特別語	
C.3 アクションファイルのサンプル	
C.4 UDFファイルとの関連付け	
付録D 汎用ファイルコンバーター	
D.1 ファイルコンバーターとは	
D.2 フィルタールールの文法	
D.3 ファイルフィルターの例	
付録E 専用コンバーターツール	
E.1 NASTRANデータコンバーターツール	
使用方法	
UDF解説	
付録F 描画対象の3次元移動(平行移動・回転移動)機能	
F.1 準備するもの	
F.2 操作手順	
平行移動	
回転移動	
付録G 環境変数生成ツール	
付録H トラブルシューティング	
H.1 Pythonスクリプティング	
H.2 3Dオブジェクトの描画	
H.3 プロットの作成	
H.4 データの編集	

図目次

図 1: GOURMET の起動画面	3
図 2: エディタのTree ビュー	4
図 3: エディタのTable ビュー	5
図 4: Engine Run パネル	7
図 5: Engine Control パネル	8
図 6: Viewer 画面	9
図 7: Plot アクション	. 10
図 8:2D プロット	. 11
図 9: GOURMETシステムの概要	. 13
図 10: エディタのTree ビューでUDF ファイルを開いた状態	. 19
図 11: エディタのTreeビューで配列を展開した状態	.21
図 12: 「…」を右クリックした時のポップアップ	.21
図 13: 「EndOfArray」マーク	. 22
図 14: エディタのTableビューでUDFファイルを開いた状態	. 22
図 15: データロケーションによるUDFオブジェクトの色表示	. 23
図 16: エディターのFile メニュー	. 25
図 17: UDF ヘッダーダイアログ	. 26
図 18: ファイルコンバータダイアログ	. 27
図 19: エディタのEdit メニュー	. 28
図 20: エディタのView メニュー	. 30
図 21: Preference/TreeView タブ	. 30
図 22: Preference/TableView タブ	. 31
図 23: Unit Conversionダイアログ	. 32
図 24: UDF Path フィールドによるフィルタリング	. 33
図 25: Table ビューにおけるKEY 値の表示	. 33
図 26: Unit メニュー	. 36
図 27: Select Unit Set ダイアログ	. 37
図 28: Browse Default Unit ダイアログ	. 38
図 29: エディタのPython Scriptingウインドウ	. 39
図 30: エディタにおけるアクション例	. 43
図 31: ビューワにおけるアクション	. 44
図 32: Engine Run パネル	.47
図 33: Engine Control パネル	. 48
図 34: ビューワの起動画面	. 50
図 35: ビューワのFile メニュー	. 52
図 36: ビューワのView メニュー	. 53
図 37: 現在のビューデータを示すテキスト画面	. 54
図 38: ビューワのDisplay メニュー	. 55
図 39: ビューワのPickingメニュー	. 56
図 40: ビューワのPythonメニュー	. 57
図 41: ビューワのOptionsメニュー	. 57
図 42: Plot タブパネル	. 61
図 43: gnuplot によるプロット表示例	. 62
図 44: ファイル転送ツール起動画面	. 66

図 45: Transmitter 接続画面	67
⊠ 46: Application Setup	68
図 47: 分子ビルダー	69
図 48: 起動環境設定ツールのPython設定タブ	71
図 49: 起動環境設定ツールのGroup設定タブ	71
図 50: NASTRANバルクファイル読み込みアクション選択	87
図 51: NASTRANバルクファイル指定画面	88
図 52: 要素の描画アクション選択	88
図 53: 部分領域の抽出アクション選択	89
図 54: 部分領域の抽出条件指定画面	89
図 55: 部分領域の描画アクション選択	90
図 56: 部分領域の描画結果(全体)	90
図 57: 部分領域の描画結果(部分選択)	91
図 58: NASTRANバルクファイルの書き出し画面	91
図 59: 要素のノード指定順序	94
図 60: 平行移動アクション選択	99
図 61: 平行移動操作ダイアログ	99
図 62: 回転アクション選択	100
図 63: 回転操作ダイアログ	100
図 64: 回転中心と回転軸ベクトル	100

表目次

表 1: 描画関数	
表 2: Operating System	
表 3: Java System	72
表 4: Graphics System	73
表 5: Python System	73
表 6: Action ファイルのBNF文法	77
表 7: フィルタールールファイルのBNF 文法	

第1章 GOURMETとは

GOURMET (Graphical Open User interface foR Multiscale analysys EnvironmenT)は、OCTAプ ロジェクトのさまざまなシミュレーションエンジンの共通ユーザインタフェースとして開発さ れました。OCTAの全てのシミュレーションエンジンは、GOURMET を使うことにより便利になりま す。GOURMETを使うと、

(1) エンジンの入力ファイルを作成し、

(2) エンジンを実行し、

(3) エンジンの出力ファイルをさまざまな形(数値、グラフ、3D アニメーション)で見て、

(4)分析し、独自のプログラムで出力ファイルを作成することができます。

いくつかのエンジンでは、実行中のエンジンの状況を見たり、計算を中止・再開したり実行中の エンジンに対して適切なタイミングでパラメータを変更するような制御を行えます。

GOURMETはスクリプト言語Pythonのインタフェースを持っており、このエレガントな言語に大 変お世話になっています。PythonによりGOURMETをあなたの好きなようにカスタマイズできるの です。Pythonプログラムを書くと、入力ファイルを自動作成したり、出力ファイルの中から選択 した特定のデータ項目を3Dグラフィックスで表示したり、解析結果をgnuplotアプリケーション でグラフ化したりするような機能を、GOURMETに追加することができます。

GOURMETの入力ファイルはUDF (User Definable Format) と呼ばれるあるフォーマットのテキ ストファイルで書かれています。UDFは、2 つの部分からできています。最初の部分はデータ構 造と単位を定義し、2つ目の部分はデータそれ自身です。UDFファイルの中では、全てのデータ(数 値および文字列) はユニークな名前を持っており、その名前でアクセスできます。このように、 UDFファイルはいつでもアクセスして変更することのできるデータストレージとみなすことがで きます。この機能は、Pythonにより強化されたものです。

UDFのどんなデータやデータ集合でもPythonの変数のように扱うことができます。

数値に名前を付けることは、他人に理解できるデータファイルを作るという点からも重要です。 GOURMETは他にもあなたのプログラムをユーザフレンドリーなものにする機能を提供しています。 それらは、

(1) 各データに単位を定義したり、

(2) あなたのプログラムのユーザを助けるためのコメントを表示したり、

(3) どんなデータにでもアクションを定義できる機能などです。

このマニュアルを読めば、このようなことをどうやって実現するかを理解できるでしょう。もっ と詳細には、他のマニュアルも見てください。

GOURMETの目的は、シミュレーション・プログラムを書いたり使ったりする様々な人々のコラ ボレーションを容易にすることです。多くのプログラムが世界中の研究所で作られていますが、 プログラムを他人に使用できるようにするためには莫大な時間が必要なため、めったにそのよう には書かれることがありません。GOURMETはこのような問題の一部を1つのインタフェースを提案 することにより解決します。(もちろん、プログラムを正しく頑丈にすることは難しい別の領域 として残っていますが)

我々は、GOURMETが手製のプログラムを他人に使えるようにし、世界の多くのプログラムのコ ラボレーションのために役立つことを望んでいます。

2001.11.30 土井正男

第2章 最初にすること

まず、GOURMET を使い始めましょう。

2.1 GOURMETを起動します

Windowsでは、スタートメニューのOCTA2010->StartGOURMETメニューを選択してください。 Linuxでは、シェルウインドウで"OCTA2010/GOURMET_2010/gourmet"を実行してください。 このコマンドは、GOURMETの起動スクリプトです。起動オプションに関する詳細は、第3章をご 覧ください。最初に編集用画面(図1)が現れます。

📥 Editor –	<new file=""></new>	<u>_ ×</u>		
<u>F</u> ile <u>E</u> dit	<u>V</u> iew <u>U</u> nit <u>Python Options Tool Window H</u> elp			
Path Histo	Dry			
< >	Solution Global Record			
UDF Path:		-		
	Name Type Value Unit			
Tree View Window				
Python	Plot			
Python:	Clear Load Save Save Lib Reload			
Run				
< undo re	Python Scripting Window			
Clear Lo	Python Log Window			

図 1: GOURMET の起動画面

・Tree View Windowでは、データ値を見たり編集することができます。

・Python Scripting Windowでは、Pythonを使ってどんなプログラムでも書くことができます。 例えば、入力データを生成したり、出力データの統計処理を行ったり、出力データを変換した りすることができます。

・Python Log Windowでは、実行したPythonスクリプトの出力結果が表示されます。

2.2 UDFを編集します

UDFファイルを開くには、File/Open... メニューを使います。

ここでは、"GOURMET_XXXX/tutorial/3dball/baseball.udf"という例題ファイルを開いてくだ さい。このファイルは、地上で投げられたボールの振る舞いをシミュレートするチュートリアル のサンプルです。

Table View Window の"Ball"をマウスの左ボタンでクリックしてください。これで、"Ball"の 属性である値と単位を見ることができます。マウスでデータを選択し、他のウインドウやアプリ ケーション間でコピー・ペーストを行うことができます。詳しい操作方法は、第4章をご覧くだ さい。

Editor - baseball.udf [D:¥OCTA2006¥GOUF	MET_2006¥tutor	ial¥3dball]		_ 🗆 X	
File Edit View Unit Python Options	<u>T</u> ool <u>W</u> ind	ow <u>H</u> elp			
Path History View	Location				
< > 🗟 Tree O Table	🔾 Global 🧕	Record			
UDF Path: Ball				•	
Name	Туре	Value	Unit		
🗋 baseball.udf		-	-		
9 🗂 Ball	struct	-	-		
-f mass	float	0.1445	[kg]		
- 📕 diameter	float	0.0715	[m]		
🔰 🗠 🗲 color[]	float array	-	-		
🛉 🗝 🚍 Environment	struct	-	-		
🕈 🗝 🚍 Parameter	struct	-	-		
🛉 🕶 🔚 Initial_condition	struct	-	-		
🗠 🚍 Solver	struct	-	-		
🛉 🕶 🔚 Goal	struct	-	-		
🗠 🚞 Calculated_results	struct	-	-		
🗠 🔚 Baseballinput	Input	-	-		
Python Plot					
Python: Clear Load	Save	Save Lib	Reload		
Python Scripting Window					
Python Log Window					
\bigtriangledown					
0 3 6 9 12 15	18 21 2	4 27 3	0 33 0	/34	

図 2: エディタの Tree ビュー

次に、Table ビューを使うために、Unit menuの下にあるViewグループから"Table"というラジ オボタンを選択してください。ビューのスタイルが、Tree View WindowからData Structure Window およびTable View Windowに変わったのが分かると思います。

- ・Data Structure Windowでは、入出力データの構造と項目を見ることができます。
- ・Table View Windowでは、データ値をテーブル形式で参照し、編集できます。

Data Structure Windowで、"Goal"をクリックし、さらに"line[]"をクリックしてください。 これらのシンボルは(UDFパスと呼ばれます)、UDFファイル中のデータの名前です。これで、 "Goal.line[]"の値を、Table View Windowで見ていることになります。これらの値をマウスドラ ッグで選択し、他のウインドウやアプリケーション間でコピー・ペーストを行うことができます。



図 3: エディタの Table ビュー

もっと詳しい操作方法は、第4章をご覧ください。

2.3 Pythonを走らせましょう

さて、開いているUDFファイルに対して、Pythonスクリプトを走らせてみましょう。Python Scripting Window に、次のテキストをタイプして、Runボタンを押してください。

print \$Ball.mass

Python Log Windowに次のような結果が得られるはずです。

0.1445

これは、Pythonスクリプトを走らせて得られた"Ball"の"mass"(質量)のデータ値です。この ようにPythonスクリプトを使えば、開いているUDFのデータにアクセスし、簡単にプログラミン グを行うことができます。詳しい操作方法は、第6章をご覧ください。

2.4 エンジンを動かしましょう

GOURMETと交信できるエンジンならば、ネットワーク上で起動し、制御することができます。 エンジンを起動したり制御するには、まずエンジンマネージャをエンジンを実行するマシンで起 動しておく必要があります。

エンジンマネージャを起動するためには、以下のようにします:

Windowsでは、スタートメニューのOCTA2010->StartEngineManagerメニューを選択してください。

Linuxでは、シェルウインドウで"OCTA2010/GOURMET_2010/eng_man"を実行してください。

エンジンを実行するマシンでエンジンマネージャが稼動していれば、GOURMETのTool/"Engine Run"メニューや"Engine Control"メニューを使えるようになります。

Tool/"Engine Run"メニューを選んで、Engine Run パネルを表示してみましょう。

図4 は、Engine Run パネルです。ここでは、エンジンを動かす条件の設定と解析の開始を行 えます。

🚔 Engine Run				
COGNAC4		New	Always Use Current UDF at Input UDF	
SUSHI4				
FORK		Remove	Display Engine Condiol Window when Run Engine	
PASTA		Duplicate 🔽	Save info & Close this Window [i.e. do OK] when Run Engine	
PhaseSeparation4_FDM				
PhaseSeparation4_FEM		The selected conten	ts	
Electrolyte4_FDM		Bun namo:	Enviro Dun	
Electrolyte4_FEM		Ruil Indille. pyoser	Kun	
MemFluid4		Server:	localbost	
Elastica4				
GelDyna4		Engine:	ython D:¥OCTA2006¥GOURMET_2006¥python¥samples¥pythonengine.py	
Turban		Morking Dir	C: ¥t own	
pyUserEngine		WORKING DIL.		
		Params:	-e 100	
		(Local Input Files)		
		(Loodinipatinoo)		
		Input UDF:		
		Parms UDF:	D:¥OCTA2006¥GOURMET_2006¥tutorial¥enginerun¥parameter.udf	
		Restart UDF:	D:¥OCTA2006¥GOURMET_2006¥tutorial¥enginerun¥resultdef.udf	
		(Local Output File	s)	
		Output UDE:		
		Output OD1.		
	_	Summary UDF:	D:¥OCTA2006¥GOURMET_2006¥tutorial¥enginerun¥summary.udf	
•		(Local Logger Ap	a)	
Cancel		Logger:		

図 4: Engine Run パネル

(補足) エンジン実行デモの設定例

Run name: pyUserEngine Server: localhost Engine: python C:¥OCTA2010¥GOURMET_2010¥python¥samples¥pythonengine.py Working Dir: C:¥temp ← (無ければ作成) Params: -e 100 Input UDF: Parms UDF: C:¥OCTA2010¥GOURMET_2010¥tutorial¥enginerun¥parameter.udf Restart UDF: C:¥OCTA2010¥GOURMET_2010¥tutorial¥enginerun¥resultdef.udf Output UDF: Summary UDF: C:¥OCTA2010¥GOURMET_2010¥tutorial¥enginerun¥summary.udf (作成される) Logger:

図5 は、Engine Controlパネルです。Engine Controlパネルは、"Summary UDF"に定義されて いる項目と、エンジンサーバから受け取った値を表示します。このパネルで、稼動中のエンジン の制御とその結果のサマリーを見ることができます。

エンジン稼動中にできることは以下の通りです。

- ・パラメータを変更するために解析を中断(Pause)する
- ・解析を中止するために終了(Stop)させる
- ・エンジンを強制終了(Kill)する

エンジンを中断(Pause)している場合にできることは以下の通りです。

- ・パラメータを編集(Edit Parameter)する
- 解析を再開 (Resume) する



図 5: Engine Control パネル

制御を変更したり結果を見たいときはいつでも、プルダウンメニュー(パネル上部の左端)から制御するエンジン名を選択できます。また、このEngine Controlパネルは、Tool/Engine Control メニューを選択すればからいつでも表示させることができます。もっと詳しい操作方法は、第7 章をご覧ください。

2.5 結果を見ましょう

次に、Window/Viewerメニューを選択して、ビューワを使ってみましょう。開かれているUDFフ ァイルに対応するビューワ画面(図6)が開きます。これは、Pythonスクリプトによって描画さ れる3Dオブジェクト描画用画面です。3D Object Windowの何も描かれていない場所を、コントロ ールキーを押しながら、マウスの左ボタンでクリックしてください。(これは、アクションスク リプトを実行する操作です)ポップアップメニューから、"show"コマンドを選択してください。 3D Object Window にボールとストライクゾーンが表示されます。



図 6: Viewer 画面

それでは、Python PanelのAnimationボタンを押して、解析結果をアニメーションしてみましょう。

ストライクッ!

"Backward [|<]"および"Forward [>|]"ボタンで、それぞれレコードを前後に移動することが できます。この例では、解析結果がUDFレコードに時系列データとして保存されているので、レ コードスライダーを使えば、どのレコードにでも移動できます。詳しい操作方法は、第8章をご 覧ください。

次に、エディタに戻って2Dプロット機能を使ってみましょう。Window/Editorメニューを選択 してください。Data Structure Windowで、"Calculated results"をマウスの右ボタンでクリッ クしてください。(これも、アクションスクリプトを実行する機能です)ポップアップメニュー から、"x-y plot"を選択してください。

Note: マウスの右クリックでアクションを実行できるUDFパスは、ボールドで表示されます

📥 Editor - baseball.udf [D:¥	#OCTA2006¥GOURMET_2006¥tutorial¥3dball]			
<u>File Edit View Unit</u>	Python Options Tool Window Help			
Path History Cree Table Cocation Coca				
UDF Path: Calculated_res	sults	•		
baseball.udf GraphSheet[] G	<pre>stime_plot xz_plot </pre>	e:Vector3D		
Python: Clear	Load Save Save Lib	Reload		
Run < > Undo redo Clear Log				
0 3 6 9	12 15 18 21 24 27 30 3:	3 0 /34		

図 7: Plot アクション

x-yプロットされたボール位置がgnuplotウインドウに表示されます。

いくつかの方法でgnuplotツールを使うことができます。もっと詳しい操作方法は、第9章をご 覧ください。

2.6 ツールを使いましょう

GOURMETのToolメニューとFileメニューには、いろいろなツールが準備されています。

- ・UDFファイルを他のマシンに転送するためのFile Transmitterツール
- ・純粋なPython環境を起動するためのPythonツール
- ・gnuplotアプリケーションを起動するためのGnuplotツール
- ・COGNACのUDFファイルにmolfileとPDBフォーマットのデータをインポートするための分 子ビルダーツール
- ・他のフォーマットのテキストデータをUDFファイルにインポートするためのファイル変換ツ



詳しい操作方法は、第10章をご覧ください。

ール

第3章 起動時のオプション

3.1 システムの概要

GOURMET (Graphical Open UseR interface for Material design EnvironmenT) はUDFファイ ルを表示・編集・描画する機能があり、ユーザーフレンドリにUDFファイルを操作することを目 的としたGUIシステムです。

GOURMETのGUI部分はJavaで開発されており、UDFファイルの読み書きはC/C++で開発されたライ ブラリが担当しています。また、3次元グラフィクスはJOGL経由でOpenGLライブラリを利用して 描画しています。

UDFファイルはインタープリタ型の言語であるPythonにより操作され、グラフのプロット機能はgnuplotを使用しています。

GOURMETはJava、OpenGLおよびC++で構築されており、Pythonとgnuplotシステムで強化されて います。

GOURMETはクライアント・サーバシステムとしても機能します。解析エンジン実行時およびリ モートファイル転送時にはGOURMETはクライアントアプリケーションとなり、エンジンマネージ ャとデータマネージャは、それぞれサーバープログラムになります。エンジンマネージャはリモ ートおよびローカルのエンジンを制御し、データマネージャはローカル (GOURMET 側) とリモー ト (エンジン側) の間でUDFファイルを転送します。

そのため、GOURMETからエンジンの制御を行うには、エンジンを稼動させるマシン上でエンジ ンマネージャを起動する必要があります。

他のマシンへUDFファイルを転送するためには、転送先マシンでデータマネージャを起動する 必要があります。

これらのマネージャの起動用スクリプトは以下の節で説明します。 GOURMETシステムの概要を図9に示します。



図 9: GOURMET システムの概要

3.2 環境変数

3.2.1 必須の環境変数

少なくとも1つの環境変数がGOURMETの起動に必要です。この環境変数はOCTAのインストーラ により自動設定されるはずです。

• PF_FILES

PF_FILES環境変数は、GOURMETのホームディレクトリのパスです。

3.2.2 オプションの環境変数

他にオプション環境変数がいくつかあります。

• PF_ENGINE

OCTAエンジンのホームディレクトリのパスで、GOURMETがOCTAエンジンの位置を知るために必要

です。

• UDF_DEF_PATH

UDFファイルの定義ファイルのあるパスです。複数のディレクトリパスを、OS依存の環境変数 区切り文字で設定できます。(Windowsの場合は";"、Unix/Linuxは":")

• UDF_ACTION_PATH

アクションファイルがあるディレクトリパスです。(OS依存の環境変数区切り文字で複数設定 できます。)

• PYTHON_LIBDIR

GOURMETのPythonライブラリディレクトリです。

インストーラを使ってインストールすれば、PF_FILESおよびPF_ENGINEの環境変数が自動的に 設定され、その他の環境変数は起動用のスクリプトファイルによって設定されます。

これらの環境変数の使い方については、次の節も参照してください。

3.3 起動用のシェルスクリプト

Javaで実装されたシステムは使用するメモリーサイズをjava実行環境の引数で指定すること ができます。デフォルトの使用メモリー量は十分でない場合が多いので、指定をしたほうがよい でしょう。Javaで使われる最大メモリーと初期メモリーサイズを-Xmx と-Xms オプションで指定 します。GOURMETはUDFファイルの内容によっては非常に多くのメモリーを使用するため、実装メ モリーよりも少ない範囲で十分なサイズを指定すべきです。

例:

マシンが512MBのメモリーを持っている場合、-Xmx256m と-Xms128m を指定。

3.3.1 起動オプション

GOURMETでは次の起動オプションをコマンドラインで指定できます。これらのオプションは起動スクリプトファイルで使用されています(gourmet.bat あるいは gourmet.sh)。

・-DPF_FILES="GOURMETのホームディレクトリパス"

デフォルトとして使うディレクトリおよびファイルをここで指定されたディレクトリ以下から探す。

・-DUDF_ACTION_PATH="アクションファイルディレクトリパス"

ディフォルトとして使うアクションファイルをここで指定されたディレクトリから探す。

・-DPYTHON_LIBDIR="Python スクリプト保存ディレクトリパス"

PythonスクリプトをLoad/Saveするためのディフォルトディレクトリを指定する。

・-DPF_MODULES="UDFコンバーター実行モジュールディレクトリパス"

UDFコンバーター実行モジュールなどが置かれたディレクトリを指定する。

- -Djava. security. manager
- -Djava. security. policy=policy_file_path

これらの2つのオプションはセキュリティーのために準備されています。 詳細は、エンジンマネージャの説明を参照してください。

3.3.2 Microsoft Windows

GOURMETを起動するには、スタートメニューのOCTA2010->Start GOURMET メニューを選択しま す。または、コマンドプロンプトウインドウで"gourmet"を実行するか、gourmet.bat をダブル クリックします。また、開きたいUDFファイルをgourmet.batもしくはそのショートカットにド ラッグ&ドロップして起動することができます。

3.3.3 Linux

シェルターミナルウインドウで、インストーラによって作成された起動スクリプトを実行しま す。(OCTA200X/GOURMET_200X/gourmet)

3.4 エンジンマネージャ

エンジンの制御を行う場合、エンジンを稼動させるマシンでエンジンマネージャを起動する必要があります。

3.4.1 起動オプション

- -DSTAND_ALONE_MANAGER=yes
- -Djava. security. manager
- -Djava. security. policy=policy_file_path

これらの3 つのオプションはセキュリティーのために準備されています。これらの設定を行う と、そのマシンをスタンドアローンとして使うことができます。スタンドアローンとは、そのマ シンをネットワーク上の他のマシンから使えなくすることを意味しており、エンジンマネージャ はpolicy_file_path で定義されたセキュリティーポリシーを使います。

重要:

ローカルマシンでのみエンジンを動かし制御するだけであり(他のマシンのエンジンを使わない)、使用しているネットワーク環境が安全でないと思う場合、OSのシステムツールを使って、 セキュリティーポリシーの設定をすべきです。

さらに詳しい説明が必要であれば、付録B エンジンマネージャのセキュリティーポリシーを参照してください。

3.4.2 Microsoft Windows

エンジンマネージャを起動するには、スタートメニューのOCTA2010->Start Engine Manager メ ニューを選択します。

または、コマンドプロンプトで"eng_man.bat"を実行します。

3.4.3 Linux とUnix

エンジンマネージャを起動するには、シェルターミナルウインドウでインストーラによって作成された"OCTA2010/GOURMET_2010/eng_man"を実行します。

3.5 データマネージャ

他のマシンとUDFファイルを転送するには、データマネージャをそのマシン(転送先マシン) で稼動させておく必要があります。

3.5.1 Microsoft Windows

データマネージャを起動するには、コマンドプロンプトで"dat_man.bat"を実行します。

3.5.2 Linux とUnix

データマネージャを起動するには、シェルターミナルウインドウで "OCTA2010/GOURMET_2010/dat_man"を実行します。

3.6 GOURMETの終了

GOURMETを終了するには、GOURMETのFile/Exitメニューを選択するか、全てのGOURMETウインド ウを閉じます。

第4章 UDFを編集するには

UDFファイルを編集するには、File/Open... メニューを選択し、File openダイアログでその ファイルを選択します。図10は、エディタでUDFファイルを開いた状態を示しています。

Path H	istory View selec	ction	UDF	Path field
	É Editor - baseball.udf [D:¥OCT∂ 2006¥G File Edit View Unit Pytho Opt	OURMET_2006¥tutoria	I¥3dball] v Help	
N	Path History View			
	< > 🗟 Tree O Ta	able 🔾 Global 🖲	Record	
	UDF Path: Goal.line[0]			-
	Name	Туре	Value	Unit
	baseball.udf	-		^
	Ball	struct		
		struct		
	🗣 🗂 Initial_condition	struct		
	🗠 🗂 Solver aerodynamic coefficia	ents struct		
	P-F-P-car	struct		
Holp non un	Γ Υ− [_] line[]	Vector3D arr		
neib hob-nb	$-\mathbf{f} \mathbf{x}$	float	0.0 [m]	
	— ́f у	float	0.0 [m]	
	f z	float	0.0 [m]	
	Line[1]	Vector3D		
		Vectorad		Ť
	Python Plot			
	Python: Clear Lo	ad Save	Save Lib	Reload
	Run			
	undo redo			
	Clear Log			
	J			
	0 3 6 9 12 15	18 21 24	27 30	33 34 /34

図 10: エディタの Tree ビューで UDF ファイルを開いた状態

- ・Path Historyを用いて、操作履歴からUDFデータパスを選択したり、履歴上の移動を行うこ とができます。
- ・Viewラジオボタンを用いて、TreeまたはTableの表示形式を選択します。
- Locationラジオボタンを用いて、GlobalまたはRecordのUDFのデータロケーションを選択します。特にデータ値を編集するときには、どのレコードにあるデータに対して編集しているかは重要です。
- ・UDF Pathフィールドに任意のUDFパスを入力してUDFデータを選択することができます。
- ・ヘルプ定義されたUDFデータ変数の上にマウスカーソルを重ねると、ヘルプ内容がポップア

ップされます。

ヘルプの内容にインターネットやローカルディスク上のファイルへハイパーリンクが定義されている場合、[Control]+マウス右クリックでポップアップされるメニューで選択したファイル内容がシステムディフォルトブラウザで表示されます。

4.1 編集モード

4.1.1 編集モード

UDFオブジェクトの値を変更するには、編集モードを用います。編集モードでは値の存在して いるものと値を作成することのできるもの全てのオブジェクト(構造)がリストされます。(非 編集モードおよび定義モードは0CTA2003以降削除されましたので、デフォルトで編集モードにな ります。)

4.2 ビュー形式の選択

4.2.1 Treeビュー

UDFデータの内容を見たいならば、Treeビューを選択します。Treeビュー形式はデータの構造 および値をツリー形式で表示します。

Treeビューで多くの配列数を持つデータを表示しようとするとツリーの展開に時間がかかる ため、指定した数だけデータ項目が表示されようになりました。配列の最初だけ表示されている 例を図11に示します。例では配列value[]が10個まで表示されており、以降は「...」で示され ています。「...」部分をダブルクリックするか「...」のツリー枝分かれ部分をマウスで左クリ ックすると次の10個の配列が表示されます。この例では配列が展開される数は10個づつであ るとしていますが、配列の展開数を変更することができます。後章の「4.6.1 エディタの Preferencesダイアログ」のTreeタブにある"Number of expanding:"で設定します。デフォルト では配列の展開数は1000になっています。

また、多少時間がかかっても一度に全ての配列が見たい時や配列の一部を見たいときは、「...」 部分をマウスで右クリックすると図12に示すポップアップが表示されます。このポップアップに は現在表示されているデータに続く次の配列のインデックスが示されます。OKボタンを押すとポ ップアップに表示されている配列インデックスに対応するデータがTreeビューに表示されます。 データ配列数が設定されているツリーの展開数よりも多い時に、上記の方法で全配列を表示させた場合、図13に示すように配列の最後に「EndOfArray」マークが表示されます。このマークを 右クリックすると図12と同様のポップアップが表示されますので、閉じるボタンを押すと最初の ツリー展開状態に戻ります。

データ配列数が設定されているツリー展開数よりも少ない時には、「...」や「EndOfArray」 マークは表示されません。

Editor - cylinder3D_a_uot.udf [D:¥OCTA200	5¥PF_ENGINE_2	005¥SUSHI5¥Su	isi¥sample¥Output] 📃			
File Edit View Unit Python Options	<u>T</u> ool <u>W</u> ind	ow <u>H</u> elp				
Path History View	Location					
	🗌 🔾 Global 🧃	Decord				
		record				
UDF Path: SUSHIOutput.phi.value]				•		
Name	Туре	Value	Unit			
👇 🚍 SUSHIOutput	struct	-	-	-		
🔷 🗂 volume_fractions	VolumeFract	-	-			
🚽 🚽 flag_of_dynamics	int	0				
🛉 👇 🚍 phi	ScalarField	-	-			
🗕 🕂 🗶 name	KEY	phi		_		
🚽 🚽 🖌 🚽 🚽 🚽 🚽	int	2		_		
📍 🛄 value[]	ScalarArray	-	-	_		
🗠 🚍 value[0]	ScalarArray	-	-	_		
🗠 🚍 value[1]	ScalarArray	-	-	_		
🗠 🚍 value[2]	ScalarArray	-	-	_		
🗠 🚍 value[3]	ScalarArray	-	-	_		
🗠 🚍 value[4]	ScalarArray	-	-	_		
🗣 🚍 value[5]	ScalarArray	-	-	_		
🗣 🚍 value[6]	ScalarArray	-	-	_		
🗣 🔚 value[7]	ScalarArray	-	-	=		
🔶 🦳 value[8]	ScalarArray	-	-	_		
🔶 🦳 value[9]	ScalarArray	-	-			
► □	ScalarArray	-	-			
	ScalarField	-	-			
🚽 🖌 free_energy	double	0.0				
🚽 🖌 excess_free_energy	double	0.0				
🔷 🗂 optional_output	OptionalOut	-	-			
i flag_of_convergence	int	0		-		
Y			Step 0			
0			1 0	/1		

図 11: エディタの Tree ビューで配列を展開した状態

💡 🗂 value[]	ScalarArray	-	-
🔶 🗂 value[0]	ScalarArray	-	-
🔶 🗂 value[1]	ScalarArray	-	-
← 🔚 value[2]	ScalarArray	-	-
🔶 🔚 value[3]	ScalarArray	-	-
🔶 🔚 value[4]	ScalarArray	-	-
← 🗂 value[5]	ScalarArray	-	-
← 🗂 value[6]	ScalarArray	-	-
← 🔚 value[7]	ScalarArray	-	-
← 🔚 value[8]	ScalarArray	-	-
← 🔚 value[9]	ScalarArray	-	-
► <u></u>	O a standard		-
	10 🕂 to	32.767 🛨	-
d free_enero			
d excess_fre Close	ок	Cancel	
- 🗂 optional_output	optionalout	-	-
🚽 👔 flag_of_convergence	int	0	

図 12: 「…」を右クリックした時のポップアップ

📔 🛛 🔶 🔚 value[32763]	ScalarArray	-	-
🔶 🗂 value[32764]	ScalarArray	-	-
🔶 📑 value[32765]	ScalarArray	-	-
🔶 🗂 value[32766]	ScalarArray	-	-
🔶 🗂 value[32767]	ScalarArray	-	-
🕨 🔶 🔶 .EndOfArray.		-	-
► □ ∨	ScalarField	-	-
🚽 🖌 free_energy	double	0.0)
- d excess_free_energy	double	0.0)
🕨 🔶 🔚 optional_output	OptionalOut	-	-
📕 🗕 🖸 flag_of_convergence	int	1)

図 13: 「EndOfArray」マーク

4.2.2 Tableビュー

UDFデータの構造を見たいならば、Tableビューを選択します。Table ビュー形式はリレーショ ナルデータベースの表示形式であり、他のアプリケーションとの間で簡単なコピー・ペースト ([Ctrl]+C、[Ctrl]+V) 操作でデータを交換することができます。(注¹)

図14 は、UDFファイルをTableビューで見た例を示しています。



図 14: エディタの Table ビューで UDF ファイルを開いた状態

¹ Table ビューで複数のセルを選択する場合、Java のバージョンによって選択方法が異なります。 Java 1.5 以降では違うセルを一旦選択しておいてから、最初の選択から選択終了までの間、Esc キーを押しながらセル範囲をマウスでドラッグします。

4.3 データロケーションの選択

UDFファイルは同じデータ名の異なる値を初期値データとゼロ個以上のレコードデータに持つ ことができます。ロケーションタイプとは初期値にあるデータを参照するのか、レコードにある データを参照するのかの区別です。GOURMETはUDFデータ名の表示色でロケーションタイプとデー タ値の有無を区別します。さらにUDFデータの初期値には、2つのタイプが使われることがありま す。初期値のみにデータを持つように"global_def"で定義されている場合と初期値およびレコー ドにデータを持てる"def"で定義されている場合です。"global_def"で定義されている場合、デ ータ名が青で表示されます。"def"定義のUDFデータ名は緑で表示されます。"def"定義のUDFデー タがレコードにある場合は黒で表示されます。データ値が存在しなければ、淡色で表示されます。 global_value'は、global_defとして定義されているため青で表示されます。'common_num'およ び'common_pos'は、初期値は存在しますが、このレコードには値が存在しないので、緑で表示 されています。他のデータはこのレコードに値が存在するため黒で表示されているます。

4.3.1 Globalロケーション

グローバルデータを変更するには、Globalラジオボタンを選択します。 Note:グローバルデータとは、OCTA2002以前には初期値(Initial Data)と呼ばれていました。

Name	Туре	Value	Unit
📘 gloval_data_sample.udf		-	-
🗌 🦸 common_num	int	111	
🚽 🚽 🧃 record_num	int	222	
🚽 🚽 🧃 comrec_num	int	2,222	
🛉 🕶 🗂 num_of_grid	Vector3d	-	-
🛉 🖵 position	struct	-	-
📔 👇 🚍 mol 🛛	Molecular_C	-	-
🛉 📥 mol[0]	Molecular_C	-	-
🛉 📥 atom 🛛	Vector3d array	-	-
💡 🗖 atom[0]	Vector3d	-	-
- ď x	double	10.0	[sigma]
– ď y	double	11.0	[sigma]
- d z	double	12.0	[sigma]
🖕 🔂 📥 📥 📥	Vector3d	-	-
🛉 🗠 🗂 common_pos[]	Vector3d array	-	-
🛉 🗠 🗂 record_pos[]	Vector3d array	-	-
🛉 🗠 🗂 comrec_pos[]	Vector3d array	-	-
🛉 📥 Unit_Parameter	struct	-	-
🖕 🗂 global_value	Vector3d	-	-

図 15: データロケーションによる UDF オブジェクトの色表示

4.3.2 Record ロケーション

レコード値を変更したり追加するには、Recordsラジオボタンを選択します。

Recordsを選択している場合、レコードデータが存在すれば、Records Sliderがエディターの 下部に表示されます。Record Sliderをマウスで動かして、レコード位置を移動することができ ます。

4.4 File メニュー

図16にエディターのFile メニューを示します。

<u>F</u> ile	<u>E</u> dit	⊻iew	<u>U</u> nit	Pyth	on	<u>O</u> ptions	Tool
<u>N</u> ev	w Wine	dow	С	trl-N			
Ope	en		С	trl-O			
<u>S</u> av	/e		С	trl-S			
Sav	/e As						
<u>C</u> lo	se		С	trl-W			
<u>R</u> el	oad		С	trl-R			
Rel	oad A	ction					
<u>H</u> ea	ader						
Say	<u>/</u> e Clip	board A	\s				
Cor	wert			►	UD)F -> Binar	у
File	Туре			•	Bir	nary -> UD	F
E <u>x</u> it	t		С	trl-Q	Co	nvertFile	

図 16: エディターの File メニュー

以下の作業を行うための操作をFileメニューから選択します。

- ・New Window:他のUDFファイルを編集したり参照するための画面を表示します
- ・Open: UDFファイルを開きます(Autorunアクションが定義されていれば実行します)
- ・Save:現在開いているUDFファイルに対する変更を保管します
- ・Save As...:現在開いているUDF ファイルに対する変更を別名で保管します
- ・Close:現在開いているUDFファイルを閉じます
- ・Reload:現在開いているUDFファイルの全てのデータをリロードします(Autorunアクション が定義されていれば実行します)
- Reload Action:現在のUDFファイルに使われているアクションファイルをリロードします (Autorunアクションが定義されていれば実行します)
- ・Header...:現在開いているUDFファイルのUDFヘッダ情報を編集または参照する画面を表示 します
- Save Clipboad As...: 現在のクリップボードの内容をテキストファイルに保管します
- Convert:テキスト形式のUDFファイルとバイナリ形式のUDFファイルを変換します。またフ ァイルフィルターを使って外部ファイルをインポートする画面を表示します。
- ・File Type:開いているUDFファイルの形式を表示します。

・Exit:GOURMETを終了します

4.4.1 UDFヘッダーを編集するには

File/Header...メニュー選択により、UDF ヘッダーを編集することができます。図 17 に UDF ヘッダーダイアログの例を示します。

≜ UDF Header	×
UDF Name:	D:\OCTA2006\GOURMET_2006\tutorial\3dball\baseb
Engine Type:	GOURMET python
Engine Version:	V1.0
IO Type:	IN/OUT
Project Name:	tutorial
Action File:	prial/action.txt;tutorial/inputaction.txt;tutorial/plotaction.txt
Comment:	
you open these lir	iks, push [ctrl]+right mouse button, and select menu.
	Cancel OK

図 17: UDF ヘッダーダイアログ

4.4.2 ファイルコンバータの使い方

File/Convert/ConvertFile... メニュー選択により、外部固定フォーマットのテキストデータ
を開いている UDF ファイルに取り込むことができます。
図 18 に ConvertFile ダイアログの例を示します。

•Data File

変換元のデータファイル

タブ区切りデータやもっと複雑な NASTRAN の bulk ファイルなど

\cdot Rule File

変換のためのフィルタルールファイル

いくつかのフィルタの例が\$PF_FILES/filter/ディレクトリにあります。

以下の例は、データファイルの第2, 3, 4列データを、それぞれ atoms[].Position.x atoms[].Position.y atoms[].Position.z に代入します。

フィルタルールファイル例:

simple example for convert molecule data from tab delimitted text file LABEL . atoms[].Position.x atoms[].Position.y atoms[].Position.z

• Input UDF

変換 UDF 定義ファイル 前述の例では、以下に示す定義を持つ入力 UDF を使用します。

入力 UDF ファイルの例:

 E^{def}

class Coodinate:{x:double, y:double, z:double}

atoms[]: {

Position: Coodinate

}

 $\text{Yend}\left\{ \text{def} \right\}$

• Output UDF

変換結果出力 UDF ファイル

変換ルールの記述は単純なので、新たなルールを簡単に作ることができます。ファイルフ ィルタの詳細は、付録 D を参照してください。

ConvertFile	×
Data File: Rule File:	D:¥OCTA2006¥nastran_convert¥cylinder_cell1.dat D:¥OCTA2006¥nastran_convert¥nastranbulkrule.txt
Input UDF:	D:¥0CTA2006¥nastran_convert¥cylinder_cell1.udf
Output ODF:	U: #ULIA2UU5#hastran_convert#cylinder_celli.udf

図 18: ファイルコンバータダイアログ

4.4.3 テキスト形式UDFとバイナリ形式UDFの使い方

GOURMET は十分大きなテキスト形式 UDF ファイルを扱えますが、100MB を超えるような多数の レコードデータを持つ UDF ファイルになると、読み込みに時間がかかるようになります。これを 解決するのがバイナリ形式 UDF です。

バイナリ形式のUDFファイルは、拡張子として.bdfを用います。バイナリUDFは文字通りデー タをバイナリ形式で持つ他、各レコードデータの先頭にレコード内の全データサイズ情報を持っ ているため、多数のレコードを持つUDFファイルの読み込みが高速に行えます。

テキスト形式UDFからバイナリ形式UDFへの変換はメニューからFile/Convert/UDF->Binary... を選択することにより実行することができます。バイナリ形式UDFからテキスト形式UDFへの変 換にはFile/Convert/Binary->UDF...から行えます。

4.5 Edit メニュー

図 19 はエディタの Edit メニューを示しています。

Edit			
Shallow Copy			
○ D <u>e</u> ep Copy			
<u>C</u> opy	Ctrl-C		
Paste	Ctrl-V		
Inse <u>r</u> t Array Elements			
Insert an Array Element	Ctrl-I		
Add Array Elements			
<u>A</u> dd an Array Element	Ctrl-E		
Delete Array Elements	Ctrl-D		
Append Record			
Insert Record			
Delete Current Record			
Delete Record			
Edit Current Record Label			

図 19: エディタの Edit メニュー

Edit メニューを使用して、以下の作業を行えます。

・Shallow Copy mode: データを表形式でコピーするモードに設定する
・Deep Copy mode: データを構造をもつオブジェクト形式でコピーするモードに設定する

・Copy: 選択されたデータを現在のコピーモードで、クリップボードにコピーする

・Paste: クリップボードから選択された場所にデータを貼り付ける

- ・Insert Array Elements...: 選択されている配列データに複数の新しい要素を挿入する
- ・Insert an Array Element: 選択されている配列要素の前に1つの新しい要素を挿入する
- ・Add Array Elements...: 選択されている配列要素の後に複数の新しい要素を追加する
- ・Add an Element: 選択されている配列要素の後に1つの新しい要素を追加する
- ・Delete Array Elements: 選択されている複数の配列要素を削除する
- ・Append Record:存在するレコードの最後に新しいレコードを追加する
- ・Insert Record...:任意の位置に複数の新しいレコードを挿入する
- ・Delete Current Record:現在のレコードを削除する
- Delete Record...: 任意の位置の複数レコードを削除する
- ・Edit Current Record Label:現在のレコードの名称を編集する

4.5.1 Copy/Pasteモードの使い分け

コピー([Ctrl]+C) とペースト([Ctrl]+V) 操作は他のアプリケーションと同様です。さらに、 2つの方式 (Shallow または Deep) からコピー・ペーストモードを選択できます。

·Shallow コピー・ペーストモード

Shallow コピー・ペーストモードは他のアプリケーションと同じく、見えている通りに表形 式でコピー・ペーストを行います。GOURMET は、このモードの時に、他のアプリケーション との間でクリップボードにあるデータを交換できます。

・Deep コピー・ペーストモード

UDF ファイルは構造化されたデータを持つことができるため、GOURMET には、選択されたデー タ全体を1つのオブジェクトとして扱う Deep コピー・ペーストモードが準備されています。 Deep コピー・ペーストモードでは、クリップボードデータは Python のリスト形式で表現さ れます。

4.6 Viewメニュー

図 20 はエディタの View メニューを示しています。 View メニューを選択して、以下の作業を行えます。

- ・Show Global: Global ロケーションのデータを表示します
- ・Show Record: Record ロケーションのデータを表示します
- ・**Preferences...**: Tree ビューと Table ビューの表示オプションを設定します

⊻iew	
O Sho	ow <u>G</u> lobal
Shows the second sec	ow <u>R</u> ecord
<u>P</u> refe	erences

図 20: エディタの View メニュー

4.6.1 エディタのPreferencesダイアログ

エディタの表示形式を Preferences メニューで変更することができます。

・TreeView Tab および TableView Tab

図 21 および図 22 は、それぞれ TreeView タブおよび TableView タブを選択した Preferences ダ イアログを示しています。

🚔 Preferences	×
Tree Table	
Initial column width: 80	Number of expanding: 1000
Show lines from parent node	
Leaf Icons	
O Use swing defaults	
Use variable type Icons	
O No Icons	
	Cancel OK

図 21: Preference/TreeView タブ

Preferences	×
Tree Table	
✓ Expand leaf scalar Show lines from parent node	☐ always show Unit ☑ Show all elements
Leaf Icons Use swing defaults Use variable type Icons No Icons	Initial column width: 80
	Cancel OK

図 22: Preference/TableView タブ

4.7 単位変換

開いている UDF が単位の定義を持つならば、Tree/Table ビューのいずれにおいても、単位変換 を行うことができます。

Table ビューでは、変換したいデータフィールドのフィールド名をマウスで右クリックします。 Tree ビューでは、変換したいデータフィールドのUnitフィールドをマウスで右クリックします。 図 23 に Unit 変換ダイアログの例を示します。

Editor – baseball.udf [D:¥OCTA2006¥GOUI	IRMET_2006¥tutorial¥3dball]								
File Edit View Unit Python Options	ns <u>T</u> ool <u>W</u> indow <u>H</u> elp								
Path History View	Location								
< > 🗟 Tree O Table	e 🔾 Global 🖲 Record								
UDF Path: Ball									
Name	Type Value Unit								
baseball.udf	· ·								
P Ball	struct								
f mass	10at U.1445[Kg]								
	float array								
Environment	Noaranav - P								
← 🗂 Initial condition	Ball.mass								
🗠 🗂 Solver 🛛 🙀	_								
🔶 🗖 Goal 🛛 🛄 🦉									
🖕 🚍 Calculated_resi									
🔶 🗂 Baseballinput	Default CANCEL OK								
A.T.									
Python Plot									
Python: Clear Load	. Save Save Lib Reload								
Rup									
i dan									
< >									
undo redo									
Clear Log									
0 3 6 9 12 15	18 21 24 27 30 33 0 /34								

図 23: Unit Conversion ダイアログ

4.8 エディタ使用のTips

・UDF Path フィールドによるフィルタリング

UDF Path フィールドはフィルタリング機能を持っています。Table ビューで配列インデッ クスを持つ UDF シンボルを入力すれば、図 24 に示すようなフィルター結果が得られます。

套 Editor - baseball.udf [D:¥00	DTA2	006¥GOURMET	2006¥tutorial¥3	dball]	<u>_ 🗆 ×</u>					
<u>File Edit View Unit Py</u>	thon	Options <u>T</u> o	ol <u>W</u> indow	<u>H</u> elp						
Path History View Location										
Tree Tree Global Record										
UDF Path: Goal.line[4]					-					
🗋 baseball.udf		line[]:(index)	x:float	y:float	z:float					
🗠 📰 GraphSheet[]		[4]	18.44	1.3	0.217					
🕈 🚍 Ball										
- f mass										
- f diameter										
J colorij										
	=									
$ -\hat{f} _{v}$										
$ - f_z $	•									

図 24: UDF Path フィールドによるフィルタリング

・Table ビューにおける KEY 値の表示

配列オブジェクトに KEY が定義されている場合、Table ビューでは図 25 のように、配列の インデックスと KEY 値が表示されます。("[0]"と"AGE")

Editor - editor_key_sample.udf Eile Edit View Unit Pytho Path History View Tree UDF Path: parameters].value[[D¥OCTA2006¥G n <u>O</u> ptions <u>I</u> o : • Table O	OURMET_2006¥ ol <u>W</u> indow cation Global © Re	itutorial¥sample) Help cord	X
editor_key_sample.udf ← ∰ GraphSheet] ♀─ ☐ parameters[] ₩ name 𝔅 name 𝔅 value]	parameters[] [0]:AGE [0]:AGE [0]:AGE [0]:AGE [0]:AGE [0]:AGE [1]:PHONE [1]:PHONE	value[] (index) [0] [1] [2] [3] [4] [0] [1]	value[]:string 1 2 3 4 5 5 123-4567 234-5678	

図 25: Table ビューにおける KEY 値の表示

・Select 値によるデータ選択機能

Select 型として定義されているデータは、入力項目リストから選択するだけでデータ入力 をすることができます。また入力項目を選択すると、同じ階層に定義されている選択した 値と同じ名称のデータ構造だけが表示され、他のデータ構造は隠れます。

・配列要素の挿入と削除

Table ビューでは、同じノード内の配列要素だけを挿入・削除することができます。これらの変更可能な要素は、インデックス番号が太字体で表示されてます。

・アクションの実行

アクションに関連付けられた UDF データ名は、太字で表示されます。すなわち、太字のデ ータ名部分を右クリックすることにより、関連付けられたアクションが実行できます。

・UDF 変数の説明文表示

UDF 変数名の上にマウスポインタを止めておくと、その内容に関する説明文が UDF ファイル のhelpとして定義されていればポップアップ表示されます。その内容に下線表示された URL リンクが含まれている場合、[Ctrl]+マウス右ボタンでメニューが表示され、選択したリン ク項目を開くことができます。

この機能はUDFファイル名のコメントをポップアップ表示した場合も同様に操作できます。 具体例は、Tutorial/3dball/baseball.udfを開き、左上のファイル名(baseball.udf)に マウスポインタを当てて上記の操作を行います。

・多次元配列へのデータの挿入

多次元配列型のデータ要素を追加する場合、上位の要素から挿入していく必要があります。 例えば、空の array2[][]に 10x10の要素を GUI メニューを使って挿入するには、array2[] に 10 個の要素を挿入した後、array2[0][]に 10 個を挿入し、各要素に値を設定していく。。。 といった操作を行うことになります。以下のような python スクリプトを使って要素の追加 と同時に値を設定した方が簡単です。

for i in range (10):

for j in range(10):

\$array2[i][j] = i*10 + j

・2次元配列の表形式表示

Table ビューを使用している時、表示対象データが array[][] のような2次元配列型なら

ば、その全(i, j)要素が縦横に配列された表形式で表示することができます。UDF Path フィ ールドで表形式表示を指定するには、例えば b[1].a[][]のように2次元配列の UDF Path がフィールドに表示された状態で Enter/Return キーを押します。Table ウインドウでは、 構造体の中に2次元配列型のデータがある場合にマウスで[...]をクリックした時点で表形 式表示されます。表形式表示を元の1次元表示に戻すには Tree ウインドウで、そのデータ をクリックします。Tree ウインドウでは表形式表示はできません。

第5章 単位系の使い方

5.1 Unit メニュー

図 26 は Unit メニューを示しています。

Unit Select Unit Set... Browse Default Unit... Unit File Import...

図 26: Unit メニュー

開いた UDF ファイルに単位系が定義されている場合、エディタ画面で Unit メニューを使うこ とができます。

- ・Select Unit Set...:変換可能な単位系を選択し、その単位系で値を表示する
- ・Browse Default Unit...: 定義されているデフォルト単位リストを表示する
- ・Unit File Import...: Unit ファイルに定義された別の単位系をインポートする

重要:

Unit 変換は GOURMET の中だけで、定義済み単位系の間の変換を行うものであり、UDF ファイル のデータ値自体は常にエンジンで定義されている単位系のままです。

5.2 単位系選択の使い方

図 27 は、Select Unit Set ダイアログを示しています。変換先単位系を選択し、全てのデータ 値をその単位系で表示することができます。デフォルト単位系は、エンジンが定義している単位 系です。



図 27: Select Unit Set ダイアログ

5.3 単位系のインポート方法

Unit File import... メニューを使って、単位系定義を GOURMET にインポートし、変換先単位 系として使用できます。

単位系ファイルの定義方法については、UDF 文法リファレンスを参照してください。 単位系定義ファイル例を以下に示します。

```
¥begin{unit_system} {"MYSI"}
CONSTANT=9.9999
[kg]
[myLength] = [10*m]
[myTime]=[ms]
[A]
[myTemp] = [mK]
[mo1]
[cd]
[rad]
[sr]
[Hz]=[1/s] // frequency
[N]=[kg*m/s^2] // force
[Pa]=[N/m^2] // pressure
[J]=[N*m] // energy
[W]=[J/s] // power
[V]=[W/A] // voltage
[Wb]=[V*s] // magnetic flux
[T]=[Wb/m<sup>2</sup>] // magnetic flux density
¥end {unit_system}
```

5.4 エンジン単位系の参照

"Browse Default Unit..."メニューで、図 28 に示すように、解析エンジン用 UDF ファイルで 定義している単位系を参照することができます。

🚔 Browse Default Unit								
Unit System : default								
Unit Name	Definition							
PI	3.141592							
[rps]	2.0*PI[rad/s]							
[rpm]	1.0/60.0[rps]							
[yard]	0.9144[m]							
	ОК							

図 28: Browse Default Unit ダイアログ

第6章 Pythonスクリプトを使うには

GOURMET は Python スクリプト言語を用いてUDFに記述されているデータに様々な操作をす ることができます。データの値を参照、編集、単位変更、レコード追加、3次元図形描画および グラフ表示などができます。GOURMET の全てのスクリプト処理は Python スクリプト処理システ ムにより実行されます。そのため、エディタとビューワはそれぞれ Python Scripting ウインド ウを備えています。



図 29: エディタの Python Scripting ウインドウ

以下の節では、GOURMET における次のような4 つのタイプのスクリプトについて説明します。

- ・エディタの Python Scripting ウインドウで直接スクリプトを実行する
- ・ビューワの Python Scripting ウインドウで直接スクリプトを実行する
- ・エディタでアクションを実行する
- ・ビューワでピッキングアクションを実行する

6.1 Python PanelのツールとPythonメニュー

エディタ画面のPythonメニューまたはエディタ画面の下にあるPythonタブでは以下のスクリプトに関する操作を実行できます。

- Run: Python Scripting ウインドウのスクリプトを実行する。実行結果は(エラーメッセージを含め)Python Log ウインドウに出力される。スクリプトが正常に実行できた場合、GOURMET はそのスクリプトを履歴に記憶する。記憶されているスクリプト履歴を再利用することができる。エディタ画面の下にある Python タブの"Run"ボタンを押すことによりスクリプトを実行することができる。
- ・Clear: Python Scripting ウインドウの内容をクリアする。
- ・Load...: Python Scripting ウインドウに外部ファイルの内容を取込む。
- ・Save...: Python Scripting ウインドウの内容を外部ファイルに保存する。
- ・Save Lib...: Python Scripting ウインドウの内容を Python preparser が変換した結果を表示し、編集結果を外部ファイルに保存する。この機能で保存された Python スクリプトは、純粋な Python 環境でも実行できる。
- Save History...: Python Scripting ウインドウの履歴を外部ファイルに保存する。保存したスクリプト履歴は、将来のセッションで再利用できる。
- ・Load History...: Python Scripting ウインドウの履歴を外部ファイルから取り込む。過去 に保存されたスクリプト履歴を再使用することができる。
- Previous History (<): Python Scripting ウインドウに履歴の直前の内容を表示する。
- ・Next History (>): Python Scripting ウインドウに履歴の次の内容を表示する。
- ・Clear Log: Python Log ウインドウをクリアする。
- ・AutoRun mode: on にすると、画面下部のスライダーで表示レコードを変更時には、自動的に「Run」が実行される。ビューワはスクリプト実行により描画されているので、常に AutoRun モードとなっている。

6.2 エディタにおけるPythonスクリプト実行

Python Scripting ウインドウでは、Python システムに用意されている全てのスクリプト処理 を実行できます。最も簡単な例では、次のスクリプトをタイプするかロードして、実行します。

例: print "hello GOURMET"

同じ方法で GOURMET Python スクリプト拡張機能も実行できます。ここで、スクリプト拡張機能とは GOURMET で UDF データ階層にアクセスするための Python 拡張機能を意味します。

Python 拡張機能には UDF ファイルおよびデータを扱う便利な機能が豊富に準備されています。 1 つの例を示すと、"\$"が前置された UDF 名を使用して UDF データにアクセスすることができま す。

例:

print \$Ball.mass

\$Environment.gravity = 9.8

6.3 ビューワにおけるPythonスクリプト処理

ビューワにおいても、エディタと同じスクリプトを実行できます。これに加えて、ビューワで は3次元描画機能を使用することができます。3次元描画機能には、sphereのような基本的な 形状描画関数および molecule や mesh のような特殊な構造を描画する関数が準備されています。

例:

pos = \$Initial_condition.position
attr = [1.0, 0.0, 0.0, 1.0, \$Ball.diameter]
sphere(pos, attr)

表1 に GOURMET で使用できる基本描画関数の一覧を示します。より詳細には、Python スクリ プトマニュアルを参照してください。

関数名	備考
<pre>line(coordinate1, coordinate2, [r, g, b, a]))</pre>	指定された RBG と透明度で描画する
line(coordinate1, coordinate2, attribute_id)	指定された描画属性 ID で描画する
<pre>point(coordinate, [r, g, b, a])</pre>	以下同様
polygon(coordinate list,[r,g,b,a])	
polyline(coordinate list, [r, g, b, a])	
disk(coordinatel,[r,g,b,t,radius,vx,vy,vz])	
ellipsel(coordinatel,[r,g,b,t,a,b,vx,vy,vz])	
ellipse2(coordinate1, coordinate2, [r, g, b, t, a, vx, vy,	
vz])	
cylinder(coordinate1, coordinate2, [r, g, b, t, radius])	
<pre>sphere(coordinate1, [r, g, b, t, radius])</pre>	

ellipsoid1(coordinate1, [r, g, b, t, a, b, c, vx, vy, vz])	
tetra(coordinate1, coordinate2, coordinate3,	
coordinate4,[r,g,b,t])	
cube(coordinatel,length,[r,g,b,t])	
cone(coordinate1, coordinate2, [r, g, b, a, radius])	
arrow(coordinate1,coordinate2,	
<pre>[r, g, b, t, radius, height])</pre>	
text(coordinate, message, [r, g, b, t, size])	
clearDraw()	

表 1: 描画関数

6.4 エディタにおけるアクション

エディタで太字表示されたUDFオブジェクトをマウスで右クリックすると表示されるポップア ップメニューでアクションを選択できます。アクションの内容はアクションファイルに記述しま す。GOURMET は、UDF ヘッダ情報に記述されたアクションファイルを、UDF ファイルのあるディ レクトリおよび環境変数 UDF_ACTION_PATH で設定されたディレクトリから探してロードします。

```
アクションファイルの例: (例題 : GOURMET_XXXX/tutorial/3dball/baseball.udf)
action Ball: setColor(BallColor="0|1|2|3|4|5") : color = eval(BallColor)
action Ball: show() : ¥begin
color=[1.0, 0.0, 0.0, 1.0, $Ball.diameter]
def drawGoal():
    lines = $Goal.line[]
    polyline(lines, 0)
drawGoal()
if $Calculated_results.position.y > 0:
    pos = $Calculated_results.position
    sphere(pos, color)
¥end
```

この例では、エディタで'Ball'をマウスで右クリックすると、(図 30 のような) ポップアップ メニューが表示されます。ここで setColor アクションを選択すると、ボールの色情報が設定さ れます。

	IRMET_2006¥tutor I s <u>T</u>ool <u>W</u>ind	ial¥3dball] ow <u>H</u> elp		<u>- 🗆 ×</u>					
Path History View	Location-								
< > 🚮 🖲 Tree 🔾 Table	e 🔾 🔾 Global 🤅	Record							
UDF Path: Ball									
Name	Туре	Value	Unit						
🗋 baseball.udf		-	-						
P 🗖 Ball	struct	-	-						
- f mass setColor	float	0.1445	[kg]						
– 🗲 diameter	float	0.0715	[m]						
🖕 🔶 🗲 color[]	float array	-	-						
👇 🗂 Environment	struct	-	-						
👇 🛄 Parameter	struct	-	-						
👇 🛄 Initial_condition	struct	-	-						
Prime Solver	struct	struct							
🕈 🗖 Goal	struct	-	-						
Calculated_results	struct	struct							
er 🔄 Baseballinput	Input	-	-						
Python Plot									
D dham			1						
Python: Clear Load.	Save	Save Lib	Reload						
Run									
K >									
undo redo									
▽									
				(34					
0 3 6 9 12 15	18 21 2	24 27 3	iu 33 <u> </u>	/J4					

図 30: エディタにおけるアクション例

アクションファイル定義の詳細については、付録Cを参照してください。

6.5 ビューワにおけるアクション

ビューワでは、特別な描画関数(CognacAtom & CognacBond あるいは setDrawRelation & resetDrawRelation)で描画された UDF オブジェクトに関連付けられたアクションのリストを表示・選択するためのポップアップメニューを表示することができます。

例えば、図 31 に示すように、[Ctrl]キー を押しながら、マウスの左クリックで分子構造をピックすると、アクション選択のためのポップアップメニューが表示され、選択したアクションを 実行することができます。

アクションファイル定義の詳細については、付録 Cを参照してください。



図 31: ビューワにおけるアクション

6.6 Pythonパネルの利用に関するTips

- ・キー操作[Ctr1+C]/[Ctr1+V] でテキストのコピー・ペーストを行うことができます。DeepCopy モードではクリップボードに Python のリスト形式でデータが保管されているため、これをペ ーストして Python 変数に簡単に代入できます。
- Python script ウインドウ内でのテキスト編集は、undo ボタンおよび redo ボタンにより、 元に戻したり繰り返したりすることができます。ただし、スクリプトの履歴機能によりスク リプト全体を入れ替えると、undo および redo はクリアーされます。
- Python Log ウインドウの Python のエラーメッセージは、エラー行情報を含んでいます。エ ラーメッセージが出た時、"line"を含む語をダブルクリックすると、Python script ウイン ドウのスクリプトのエラー行がハイライトします。
- ・Python script ウインドウと Python Log ウインドウの間のディバイダーによって、各ウイン ドウの大きさをマウスで変更できます。

• Python 実行の中断

Python スクリプトの実行中は、Python パネルの Run ボタンが Stop ボタンに変わり、この STOP ボタンを押すとスクリプトの実行を強制的に中断することができます。この操作により、ア クションによる Python スクリプト実行も中断できます。スクリプトの実行を中断した場合、 データ処理の途中で処理が中断しているため、一旦 UDF ファイルを閉じる必要があります。

第7章 エンジンを実行するには

エンジンが GOURMET をサポートしている場合、GOURMET からネットワークを通してエンジンを 起動したり、制御することができます。ここで「GOURMET をサポートする」という意味は、最低 限 UDF ファイルを入出力できるということです。もしも、制御(中止や再開など)を行いたけれ ば、そのエンジンは OCTA エンジンのようにプラットフォームインタフェースライブラリを使っ てプログラムされている必要があります。

また、エンジンの実行と制御を行いたければエンジンサーバ側でエンジンマネージャを起動し ておく必要があります。エンジンサーバ側でエンジンマネージャが起動している場合に、 Tool/"Engine Run"や"Engine Control"メニューを使用することができます。

7.1 エンジンマネージャ

GOURMET からエンジンの実行と制御を行う場合、エンジンを実行するマシンでエンジンマネージャを起動しておく必要があります。ローカルマシン(GOURMET を使うマシン)上でエンジンを稼動させる場合、同マシンでエンジンマネージャを起動します。

重要: ローカルマシンでのみエンジンを動かし制御するだけで(他のマシンのエンジンを使わ ない)、使用しているネットワーク環境が安全でないと思う場合、OSのシステムツールを使っ て、セキュリティーポリシーの設定をすべきです。詳細は、付録B「エンジンマネージャのセキ ュリティーポリシーについて」を参照してください。

7.2 Engine Runパネル

Tool/Engine Run メニューから Engine Run パネルを開いて、起動するエンジンに関する条件 を設定し、解析を開始することができます。図 32 に、Engine Run パネルを示します。 このパネルを用いて以下のような条件を設定することができます。

- Run name: エンジン実行名称の設定
- Server: ローカルでエンジンを実行する場合は、"localhost"、サーバーでエンジンを実行する場合はサーバーのホスト名か IP アドレスを設定
- ・Engine:エンジン実行スクリプトまたはモジュールの設定
- Working Dir: エンジン実行時の作業用 Directory の設定

• Params:エンジンに対応する起動時引数の設定

各エンジンのコマンドライン引数を設定します。

- Input UDF: 入力 UDF ファイルの設定
- ・Params UDF: 各エンジンの実行中に変更可能なパラメータファイルの設定
- •Restart UDF: リスタートファイルを設定
- Output UDF: 出力 UDF ファイルの設定
- ・Summary UDF: サマリーUDFファイルの設定
- ・Logger:任意の実行ステップにおける解析監視スクリプトまたはモジュールを設定
- Always Use... :

ON にすると Input UDF には現在 Open している UDF のファイル名とディレクトリ名が常に画面にセットされます。

• Display Engine Control Window... :

ON にすると Run 時に Engine Control Window を自動的に表示します。

• Save info & close... :

ON にすると Run 時に設定内容を保存し、この Engine Run パネルを自動的に閉じます。 実行制御の設定内容を複数設定することができます。

•New:エンジン設定項目の新規作成

ノート:エンジン実行名称がリストに表示されます

- Remove:エンジン設定項目の削除
- Duplicate:エンジン設定項目の複製

🚖 Engine Run				_ 🗆 🗙
COGNAC4		New	Always Use Current UDF at Input UDF	
SUSHI4				
FORK		Remove	✓ Display Engine Control Window when Run Engine	
PASTA		Duplicate	Save info & Close this Window [i.e. do OK] when Run Engine	
PhaseSeparation4_FDM				
PhaseSeparation4_FEM		The selected conte	ents	
Electrolyte4_FDM		Dur un un DioTh		
Electrolyte4_FEM		Run name: PASTA	Run	
MemFluid4		Server:	local bost	
Elastica4		Sciven		
GelDyna4		Engine:	"D:¥0CTA¥0CTA2005¥PF_ENGINE_2005¥bin¥win32¥pasta.exe"	
Turban		Morking Dir	C•¥temp	
pyUserEngine		Working Dir.	c.+temp	
		Params:		
		(Local Input Files	3)	
		Input UDE:	D.¥OCT42005¥PE_ENGINE_2005¥P4ST4¥P4ST4¥pasta_shear_in_udf	
		inpactori		
		Parms UDF:		
		Restart UDF		
		Tubber of the second		
		(Local Output Fil	es)	
		Output UDF:	D:¥OCTA2005¥PF_ENGINE_2005¥PASTA¥PASTA¥pasta_shear_out.udf	
		Summary UDF:	D:¥OCTA2005¥PF ENGINE 2005¥PASTA¥PASTA¥pasta sum.udf	
	•			
		(Local Logger A)	pp)	
Cancel OK		Logger:		
		1		

図 32: Engine Run パネル

7.3 Engine Controlパネル

Engine Control パネルを用いて、実行中のエンジンを制御し、結果のサマリーを見ることができます。図 33 に Engine Control パネルを示します。



図 33: Engine Control パネル

複数のエンジンプロセスがある場合は画面左上のコンボボックス([MM/DD HH:MM:SS] Run name の形式表示)より選択して画面を切り替えることができます。

画面内の Report & Graph SummaryData パネルにはサマリーUDF ファイルの内容が一定間隔で簡 易表示されます。

- 上部のテキストエリアには UDF データの report_attribute. label[]、report_data. value[] および graph_attribute. label[]のテキスト表示が行われます。
- ・下部のグラフエリアには UDF データの graph_attribute. title のタイトル表示および graph_data. item[]. value[]のグラフ表示が行われます。

エンジンプロセスの制御は以下のボタンで行います。

- ・Pause:エンジンプロセスの実行を一時停止します。
- ・Parms UDF…: PAUSE 状態のみ使用可能で、パラメータファイルを対象に Editor を起動しま

す。

- Resume:パラメータファイルをエンジンサーバに転送して、エンジンプロセスの実行を再開 します。
- Stop:エンジンプロセスの実行を停止します。
- Kill:エンジンプロセスの実行を強制終了します。
 Stop および Kill されたエンジンプロセス情報を破棄するには以下のボタンで行います。
 Remove Info 画面からエンジンプロセス情報を破棄します。このとき"also Remove Working dir"のチェックボックスが ON になっていると、エンジン実行時の作業用 Directory も同時に削除されます。
- Show Detail...: Engine Run Window より起動されたエンジンプロセスに関する情報を表示 します。
- ・Start Logger: 押した時点で、Engine Run パネルで設定した Logger を起動します。
- Set Refresh: サマリーファイル内容を Report & Graph SummaryData パネルへ簡易表示する 間隔(sec)を設定します。

7.4 エンジン制御に関するTips

・エンジン制御にリコネクトするには

実行時間が非常に長い(例えば1日以上)場合、エンジン制御パネルを閉じたり GOURMET を 終了してもエンジンは実行しつづけます。次回 GOURMET を起動したとき、Tool/Engine Control... メニューを選択すると、エンジン制御パネルに再接続できます。

・エンジン実行結果をもう一度見るには

保存されているエンジン実行結果をもう一度見るには、Engine Control パネルの左上端のプ ルダウンメニューから選択します。

第8章 3Dオブジェクトを表示するには

8.1 ビューワの起動画面

ビューワは 3D オブジェクトを描画するウィンドウで、Editor 画面の Window メニューで Viewer を選択するか、Editor 画面で描画用スクリプトを実行すれば表示されます。図 34 は、ビューワ の起動画面を示しています。

≜ ∕Vi	iewer – I	baseball.ud	f [D:¥OCTi	A2006¥GO	URMET_20	06¥tuto	rial¥3dball]			- II X
<u>F</u> ile	⊻iew	Display	Pic <u>k</u> ing	Python	<u>O</u> ptions	Tool	Window	Help		
	¥ ××		3	DO	bjec	t W	′ ind o	W		
Pyt	hon:		ear	Load.		ive	Save Lib		Reload]
	lo rec					innan an				
1		4		69	12 15	18 2	21 24 27	30 3	3 0	/ 34

図 34: ビューワの起動画面

ビューワの画面は、メニュー、図形パネルおよび Python パネルから構成されています。図形 パネルと Python パネル境界はマウスにより移動することができます。

8.2 3Dオブジェクトウインドウ

3D オブジェクトウインドウ内の以下の操作により、他の 3D アプリケーションと同様な表示変

更を行うことができます。

・回転(マウス左ボタンでドラッグ)

- ・ズーム(マウス右ボタンでドラッグ)
- ・ウォークスルー([alt] または[shift] を押しながら、マウス右ボタンでドラッグ)
- ・ウォークサイド([alt] または[shift] を押しながら、マウス左ボタンでドラッグ)
- ・視点方向の設定(View メニューで選択)
- ・背景色、効果、分割数の設定(Display メニューで選択)
- ・描画イメージのキャプチャ(Options メニューで選択)
- ・他の描画オプション設定(Options メニューで選択)
- ・アニメーション(Python パネルのボタン選択)

8.2.1 3Dオブジェクトのピッキング

特別な描画関数(CognacAtom & CognacBond あるいは setDrawRelation & resetDrawRelation) を使って描画され、アクションが準備されている UDF オブジェクトをピックしてアクション実行 ポップアップメニューを表示させることができます。

デフォルトアクション (UDF 全体に対するアクション) が定義されている場合、背景を[Ctr1] を 押しながら、マウス左ボタンでクリックするとデフォルトアクションメニューが表示されます。

3D オブジェクトが"CognacAtom & CognacBond"関数で描画されており、1 つ以上のアクション が UDF パスに関連付けされている場合、描画対象を[Ctr1]を押しながら、マウス左ボタンでクリ ックすると関連付けされた UDF のアクションメニューが表示されます。

3D オブジェクトが複数ピッキングの対象であり、1 つ以上のアクションが関連する UDF パスに 定義されている場合、Picking/Multi Picking Mode メニューあるいは Ctrl+M で、シングルピッ キングモードとマルチピッキングモードを切り替えることができます。

アクションファイルの定義に関する詳細は、付録Cを参照してください。

8.3 ビューワのPythonウインドウ

ビューワの Python ウインドウはエディタと同じ機能を持っています。さらに下部の Start/Stop/Backward/Forward ボタンにより、3D オブジェクトのアニメーションを制御すること ができます。

8.3.1 3Dオブジェクトのアニメーション

対象となる UDF が複数の Record を持っている場合は、Record 操作ボタン(Start、Stop、Backward、 Forward)、あるいは画面下部のスライダーによって、表示する Record を変更することができま す。ビューワ画面ではレコード移動時に、前に実行した Python スクリプトを再実行します。

• Start:

現 Record から最後の Record まで連続して前方向に処理する。(アニメーション)

・Stop: アニメーション処理を中止する。

• Backward:

現 Record から戻り方向に処理する。(ボタンを押している間は処理を続行し、離すと処理 を中止する。)

• Forward:

現 Record から進む方向に処理する。(ボタンを押している間は処理を続行し、離すと処理 を中止する。)

8.4 ビューワのメニュー

8.4.1 Fileメニュー

<u>F</u> ile		
<u>N</u> ev	w window	Ctrl-N
Ope	en	Ctrl-O
<u>C</u> lo:	se	Ctrl-W
<u>R</u> el	oad	Ctrl-R
Rel	o <u>a</u> d Action	
Cor	wert	•
E <u>x</u> it	t	Ctrl-Q

図 35: ビューワの File メニュー

- **Open...**: UDF ファイルを開く。
- ・Close: ビューワを閉じる。
- ・Reload: 開いている UDF ファイルをリロードし、再描画する。
- ・Reload Action:開いている UDF ファイルに関連したアクションファイルをリロードし、実行する。
- Convert: テキスト形式の UDF ファイルとバイナリ形式の UDF ファイルを変換します。また ファイルフィルターを使って外部ファイルをインポートする画面を表示します。
- ・Exit: GOURMET を終了する。

8.4.2 Viewメニュー

View メニューにより、3D オブジェクトウインドウの視点設定を変更できます。

• Standard:

3D パネル内の View をすべてのオブジェクトが画面に表示されるような標準的な状態に変更 します。Run ボタンから描画スクリプトを実行すると Standard 状態になります。

• Reset:

3D パネル内の View を 3D パネル内のマウス操作を行なう前の状態に変更します。 すなわち最後に行われた Standard 状態に戻します。

View					
<u>S</u> tandard					
Reset	t				
Direc	tion				
View	Register	۲	<u>R</u> egist	Þ	F1
Rotat	ion	۲	<u>C</u> all	Þ	F2
			<u>E</u> rase	Þ	F3
			Text		F4
		L		_	F5

図 36: ビューワの View メニュー

• Direction:

3D パネル内の View の視線方向を設定します。標準(Standard)視線方向および現在視線方向を設定することができます。

指定方法はView 設定画面で注視点から視点のベクトルで指定します。 ベクトルの大きさとは関係なく注視点から視点の距離は一定に保たれます。

• View Register:

ビューワでの描画図の見え方を登録(Register)しておき、後で呼び出す(Call)ことができます。また登録内容を消去(Erase)できます。ビューの登録数は、F1~F5までの5個です。登録および呼び出しは、図のメニューから操作できる他、コントロールキー+ファンクションキー(1~5)を選択することにより登録し、ファンクションキー(1~5)を押すことにより呼び出すことができます。(注²)

また、登録されるとメニューの F1~F5 の文字が太字になり、どのキーにビューが登録された ているかを表示します。登録したビューは Viewer 画面を閉じても記録されています。

View Register の Text を選択して、テキスト画面に現在のビューデータが表示される様子 を図 37 に示します。このテキストデータを取っておいて異なる View 画面で利用することが できます。

🛓 View information	1	×	
CenterOfRotationX:0.0			
CenterOfRotationY:0.0			
CenterOfRotationZ:0.0			
Scale:1.0			
TranslationX:0.0			
TranslationY:0.0			
TranslationZ:0.0			
QuaternionQ0:1.0			
QuaternionQ1:0.0			
QuaternionQ2:0.0			
QuaternionQ3:0.0			
EyePositionX:0.0			
EyePositionY:0.0			
EyePositionZ:12.12	20724		
CenterOfObjectsX:	0.0		
CenterOfObjectsY:	0.0		
CenterOfObjectsZ:	0.0		
UpwardDirectionX:0.0			
UpwardDirectionY:1.0			
UpwardDirectionZ:0.0			
Canaal	Change	7	
Cancel	cnange		

図 37: 現在のビューデータを示すテキスト画面

²ファンクションキーを利用するときは、マウスでビューア画面内を一度クリックするなどして、 ビューア画面をアクティブにしておく必要があります。Python スクリプト画面がアクティブに なっているとビューの登録および呼び出しをキー操作で行うことはできません。

Change ボタンを押すとテキスト画面のビューデータに従って描画図の見え方が変わります。

・Rotation:
Left:左方向に設定角度分回転
Right:右方向に設定角度分回転
Up:上方向に設定角度分回転
Down:下方向に設定角度分回転
Clockwise:時計回りに設定角度分回転
Anticlockwise:反時計回りに設定角度分回転
Angle:キー操作(矢印キーおよび[]キー)または上記メニューによる回転の角度を設定します。

8.4.3 Displayメニュー

3D オブジェクトウインドウの表示方法に関する設定を変更できます。

Display	
Back <u>G</u> round	Black
☑ <u>A</u> xis	White
☑ Perspective	Any
☑ Lighting	Image
☑ Legend	
□ <u>S</u> mooth	
☑ Wire Fra <u>m</u> e Movement	
Draw Size/Width/Divisions	
	,

図 38: ビューワの Display メニュー

• BackGround:

Black:図形パネル(3D)の背景を黒色に変更します。

White:図形パネル(3D)の背景を白色に変更します。

Any...: 図形パネル(3D)の背景をカラーダイアログで指定した色に変更します。

Image...: 図形パネル(3D)の背景にイメージを貼り付けます。イメージ形式は JPEG または PNG が利用できます。イメージの貼り付け方として以下の方法があります: Fit max:縦横比を変えずにイメージ全体が画面に入るように貼り付けます。 Fit min:縦横比を変えずに画面全体がイメージで覆われるように貼り付けます。 Raw size:元のイメージサイズで貼り付けます。

Fit canvas:イメージ全体を画面の幅・高さに合わせまて貼り付けます。

• Axis:

XYZ 軸の表示の有無を切り替えます。

Perspective:

透視法射影(Perspective projection)あるいは正射影(Orthographic projection)の表示を 切り替えます。

• Wire Frame Movement:

マウス操作(回転、移動、拡大)中に、操作を早くするために線画で表示するかどうかを 切り替えます。

• Lighting:

Lighting の効果の有無を設定します。

Lighting の効果を有にすると、立体的に見えます。

Lighting の効果を無にすると、色表示が View の方向に関係なく同じ色に見えます。

ノート: Lighting の効果を無しにすると、SPHERE 等は立体感が無くなります。

• Smooth:

画面に表示している線分を滑らかに表示します。

• Draw Size/Width/Divisions...:

Point(点)のサイズ、Line(線分)の幅、Sphere(球)、Ellipsoid(楕円体)、Cylinder(円 柱)、Cone(円錐)、Disk(円)およびEllipse(楕円)のポリゴン分割数を変更します。 ノート:ポリゴン分割数を増やすと、なめらかに物体が表示されますが、消費メモリーの増加と レスポンスの悪化を招きます。

8.4.4 Pickingメニュー

Pic <u>k</u> ing	
🗆 <u>M</u> ulti Picking Mode	Ctrl-M
<u>Clear all picking</u>	Ctrl+Shift-M
Picked Object Color	

図 39: ビューワの Picking メニュー

• Multi Picking Mode

シングルピッキングモードと複数ピッキングモードの切り替えを行います。

 \cdot Clear all picking

ピックされている状態を解除します。

• Picked Object Color...

ピックされたオブジェクトの色をカラーダイアログにから変更します。

8.4.5 Pythonメニュー

Python .
<u>R</u> un
Run with Current <u>V</u> iew
<u>C</u> lear
<u>L</u> oad
<u>S</u> ave
S <u>a</u> ve Lib
Load <u>H</u> istory
Save History
Prev History
Next History
Cl <u>e</u> ar Log

図 40: ビューワの Python メニュー

Python パネルのボタン操作をメニュー形式にしたものです。詳細については 8.3 を参照して ください。

8.4.6 Optionsメニュー

Options		
ImageCa	apture	۲
<u>D</u> raw Mi	SC	
<u>A</u> rrow Shape		
Attribute	e <u>F</u> ile Setup	
Python/F	Plot	۲

図 41: ビューワの Options メニュー

• ImageCapture:

描画領域のイメージファイル保存(自動保存を含む)についての設定をします。

- Now Capture GraphicPanel

現在の描画領域をイメージファイルに保存します。

- Auto Capture GraphicPanel while animation

Record 操作 animation 中の描画領域の自動保存の ON/OFF を切り替えます。

ノート:デフォルトの保存するディレクトリー・ファイル名は以下の通りです。

(BaseDir)/(UDFFileName)/graphic/(RecordNumber).jpg

例:

ImageCapture/blend_eq_uot.udf/graphic/999.jpg

- Save Option...:以下の設定をします

イメージファイルを保存するディレクトリの設定、イメージファイル形式 (JPEG または PNG)の選択および JPEG 形式の場合の Quality (0-100)を設定します。

• Draw Misc...:

描画に関する動作等を設定します。

- Clear Option:

Run 実行時に常に画面の描画を Clear してから描画するか、追加で描画するかを選択します。

- Cache Type for animation:

アニメーション実行時に描画するオブジェクトの取り扱い方法を選択します。

+ USE UDF-Cache

描画スクリプト実行によって生成された描画オブジェクトを描画します。ピッキ ングからのアクション実行により描画オブジェクトに関連付けられた UDF データ を扱う場合は、このモードにしておかないと正しい UDF データ値が得られません。

+ USE Viewer File-Cache

Viewer 内部で各 Record 毎の描画内容を自動的に外部ファイル (File-Cache) に 保存しておき、描画スクリプトに変化のないかぎり2回目からの描画に File-Cache を利用して描画の高速化を図ります。

+ USE Viewer Memory-Cache

Viewer 内部で各 Record 毎の描画内容を自動的にメモリ(Memory-Cache)内に保存しておき、描画スクリプトに変化のないかぎり2回目からの描画に Memory-Cache を利用して描画の高速化を図ります。

ノート:Viewer File-Cache および Viewer Memory-Cache が有効なのは以下の操作に限ります。

- + アニメーション操作ボタン(Start, Stop, Prev, Fwd)の操作
- + アニメーションスライダーの操作
- + Record 番号表示テキストエリアの入力
- + Record Label 表示テキストエリアの入力

- Animation Interval[millisecond]:

アニメーション間の休止時間をミリ秒単位で入力します。

- Animation Slider:

アニメーションスライダーをマウスでドラッグ操作中に、常にマウス位置に対応する Record の描画を行うか、最後にマウスボタンを離した時に描画を行うかを選択します。

- Auto Rewind:

アニメーションを繰り返し実行するかどうかを選択します。

• Arrow Shape...:

矢印の描画形状を設定します。

- User arrow length scale

描画スクリプト arrow([x1,y1,z1],[x2,y2,z2],[r,g,b,a,h,w,s])で描画された矢印の線分部 分の描画長さを調整するためのスケールを設定します。(通常は 1.0)

- Mesh arrow length scale

meshfield によってメッシュ格子点に設定されたベクトル値を描画する場合の矢印の 線分部分の描画長さをメッシュサイズの最大値の何倍で描画するかを指定します。(通 常は1.0)

- Arrowhead Shape

矢印の先端部分の形状を Cone(円錐) あるいは複数 Line(線分) にするかを設定しま す。Line(線分)の時、構成ライン数も設定できます。

- Auto arrowhead size

"Resize"にチェックを入れると、矢印の最大長さに対する矢じり長さの割合("scale") で矢じりが描画されます。"aspect"は矢じり長さに対する矢じり幅の割合です。

• Attribute File Setup...:

画面に表示する色等の設定ファイルを選択します。この設定ファイルの内容については、 Python スクリプトマニュアルの描画属性ファイルを参照してください。

• Python/Plot:

Python/Plot スクリプトパネルに表示する Font、History の設定をします。

- Font...

Python/Plot スクリプトパネルに表示する Font について、Font 名、Style、Size で設

定します。

- History...

Python/Plot 実行スクリプトパネルの内容を記憶する History に関して Window を Open/Close した時に自動的に Load/Save するかを設定します。また、History の記憶 最大数の設定も可能です。

第9章 プロットを行うには

解析結果の分析のために、gnuplot を使ったプロットを行うことができます。

9.1 Plotツール

Python パネルと Plot パネルは、Python/Plot タブで切り替えることができます。図 42 は Plot タブが選択されたエディタの画面を示しています。

≜ Editor - baseball.udf [D:¥OCTA	2006¥GOURMET	_2006¥tutorial¥3	dball]	<u>_0×</u>	
Eile Edit View Unit Pytho	n <u>O</u> ptions <u>T</u> o	ol <u>W</u> indow	<u>H</u> elp		
Path History View	Lo	cation			
< > 🗟 🔿 Tree	e 🖲 Table 🛛 🔾	Global 🔘 Red	cord		
UDF Path: GraphSheet[]					
🗋 baseball.udf 📃 🔺	GraphSheet[pos-x:float	pos-y:float	pos-z:float	
P □ GraphSheet[]	[0]	0.7969704	1.7965865	-8.80193E-4	
– f pos-x	[1]	1.589932	1.7886432-0	.002924135	
— f pos-y	[2]	2.3789213	1.776196 -	0.0061201	
⊢ f pos-z	[3]	3.1639738	1.7592702-0	.010456794	
🗠 🗂 Ball	[4]	3.9451246	1.7378907 -	0.01592269	
🗣 🗂 Environment	[5]	4.7224083	1.7120821-0	.022506686	
← C Parameter	[6]	5.495858	1.6818688-0	.030197768	
C Initial condition	[7]	6.265507	1.6472745-0	.038985077	
	[8]	7.0313888	1.608323-0	.048857898	
Solver	[9]	7.793534	1.5650374-0	.059805665	
9- 🛄 Goal	[10]	8.551974	1.5174408 -	0.07181796	
Ŷ- □ line[[11]	9.306741	1.4655557 -	0.08488448	
$-f_{1} \times$	[12]	10.057863	1.4094045-0	.098995075	
- f y	[13]	10.80537	1.3490093	-0.1141397	
— f z	[14]	11.549293	1.2843919	-0.1303086	
Python Plot					
Plot: Clear	Load	Save	Make	Reload	
Plot set title "Gra	phSheet [] "				
plot "plot.dat	″usin≪ 1:2 tit	le 'pos-x' wit	th lines . ¥		
"nlot dat" using 1:3 title 'nos-v' with lines ¥					
"plot dat	" using 1:4 tit	le 'pos-z' wit	th lines		
proc.dat	donis 1.4 cro	10 003 2 111			
	45 40		27 20	0 3	
U J 6 9 12	15 18	21 24	27 30	33 <u>0</u> /3	

図 42: Plot タブパネル

Plot ツールのグラフ化機能は、データを簡単に可視化することを目的としています。グラフフ オーマットを詳細に設定する必要があるならば、Plot ツールで生成されたプロットコマンドを 変更するか、gnuplot のインタプリタでパラメータを調整します。

プロットの作成は次の手順で行います。

・プロットしたいデータを Table ビューで表示します。

・Plot タブパネルの Make ボタンを押すと、プロットコマンドとデータが生成されます。この 時、プロットデータがカレントディレクトリの plot. dat に生成されます。

・プロットコマンドを編集します。

Plot ボタンを押すと、プロットコマンドとデータを渡されて gnuplot アプリケーションが起動します。この時、Plot パネルのコマンド内容がカレントディレクトリの cmd. dat に生成されます。図 43 は、gnuplot アプリケーションによるプロット表示例を示しています。



図 43: gnuplot によるプロット表示例

9.2 Graph sheetオブジェクト

GOURMET には、開いている各 UDF 毎に、データの整理領域として GraphSheet というワークシー トオブジェクトを準備しています。Table ビューを使用している時に、プロット用データを保管 するために GraphSheet オブジェクトを利用することができます。

GraphSheet オブジェクトのデータは通常、Python スクリプトによって生成されますが、他の UDF オブジェクトと同様に直接入力することもできます。

ワークシート作成のために、Python 関数として、行の追加、削除、データカラムの追加が準備 されています。

9.2.1 GraphSheet操作を行うPythonスクリプトの例

GraphSheet 操作を行う Python スクリプトの使用例を以下に示します。さらに詳しくは、 "GOURMET Python スクリプトマニュアル"を参照してください。

```
GraphSheet 操作を行う Python スクリプトの使用例:
from math import *
createSheetCol(0,'Refference')
createSheetCol(1,'Position')
n = totalRecord()
for i in range(0, n):
    jump(i)
    rsize=getSheetRowSize()
    if rsize <= i:
        insertSheetRow(rsize, 1)
        setSheetData(0, i, [-10*sin(pi*i/n)])
        setSheetData(1, i, [get("Structure.Position.mol[0].atom[0].x")])
```

9.3 Plotスクリプティング

GOURMET には、gnuplot.py という gnuplot インタフェースライブラリが準備されており、この Plot ライブラリを、例えばつぎのような Python スクリプトの中で通常の Python モジュールと 同様に用いることができます。

- ・GOURMET アクションファイルの Python スクリプト
- ・GOURMET の Python スクリプトウインドウ
- ・GOURMET のエンジン制御におけるロガースクリプト

純粋な Python 環境アクションファイルでの Plot ライブラリ使用例
action Calculated_results : xy_plot() : ¥begin
import gnuplot
x = []
y = []
for rec in range(totalRecord()):
 if \$Calculated_results.time > \$Solver.tmax:
 break
 jump(rec)
 x.append(\$Calculated_results.position.x)
 y.append(\$Calculated_results.position.y)

gnuplot.plot(data=[x, y], labels=['x', 'y'], title='xy-position')
¥end

9.3.1 Plotライブラリの使い方

Python スクリプトから gnuplot 用スクリプトライブラリを使用できます。このライブラリの概要を以下に示します。詳細な仕様については、PF_FILES/python/gnuplot.py を参照してください。

udfdata(udf, datafile='plot.dat', labels, axis='field')

この関数は、開いている UDF (UDFManager のインスタンス)を使って、datafille (プロッ トデータのファイルパス)を出力します。

labels はプロットする UDF シンボルパスのリストです。axis (field または row) は、プ ロットするデータ系列を指定します。

rowdata(datafile='plot.dat', datalist, axis='row')

この関数は、指定した datalist を使って、datafile(プロットデータのファイルパス)を 出力します。

datalist は、プロットする値のリストです。axis (field または row) は、プロットするデ ータ系列を指定します。

start(cmdfile='plot.cmd')

この関数は、cmdfile (プロットコマンドのファイルパス)を渡して gnuplot を起動します。

gnuplot(commands, cmdfile='plot.cmd', datafile='plot.dat')

この関数は、cmdfile と datafile を作成した後に gnuplot を起動します。 commands は、gnuplot コマンド行の文字列リストです。cmdfile は、gnuplot に使われるコ マンドファイルパスです。datafile は、cmdfile から参照するデータファイルパスです。

plot(data, title='', labels, attrs, udf, cmdfile='plotficmd',

datafile='plot.dat', axis='field')

この関数は、指定したパラメータで cmdfile (プロットコマンドのファイルパス)を作成し、 これを渡して gnuplot を起動します。

data は、プロットデータ値のリストです。udf が指定されていれば無視されます。title は、 プロットのタイトルです。labels は、プロットラベルもしくは UDF パスです。attrs は、
プロット属性です。

udf は、開いている UDFManager のインスタンスです。cmdfile は、gnuplot に使われるコマ ンドファイルパスです。datafile は、cmdfile から参照するデータファイルパスです。axis (field または row) は、プロットするデータ系列を指定します。

第10章 ツールを使うには

GOURMET には、Tool メニューと File メニューに以下のようなツールが準備されています。

10.1 ファイル転送ツール

ファイル転送ツール(Transmitter)を使って、UDFファイルを他のUDFサーバ(データマネージャが起動しているマシン)との間で転送することができます。ファイル転送ツールはリモートマシンで起動しているデータマネージャと連携してローカルとリモート間のファイル転送機能と各マシン内での基本的なファイル操作およびUDFのEngine Type/Version等のヘッダ情報の表示を可能にします。ローカルでのみ使用するときは、データマネージャの起動は不要です。

・リモートマシンとの接続初期状態では、左側の LocalHost パネルに表示されるディレクトリは現在 Open してる UDF のディレクトリです。UDF を Open していないときは、GOURMETの実行ディレクトリです。リモートマシンと接続するために Remote Host のテキストエリアに Host 名あるいは IP アドレスを入力して、Connect ボタンを押します。

図 44 は Transmitter パネルの起動時の状態を示し、図 45 はリモートホストと接続した状態を示しています。

-Local Host		1	Remote Host	
Dir: 006\GOURMET_2006\tut	orial\3dball\		🖾 🔒 Dir:	
script	Make Dir			Make Dir
🗋 3dball.def	Rename			Rename
🗋 baseball.udf	Delete			Delete
🗋 golf.udf	Refresh			Refresh
🗋 parabola.udf				
🗋 tabletennis.udf				
🗋 tennis.udf				
Show Header Info			Show Header Info	
[Engine name] GOURMET python	^			
[Version] V1.0	_			
[I/O] IN/OUT	_			
[Action File] tutorial/action.txt.t	utorial/-			
		1		
Connect Remote Host:				Exit

図 44: ファイル転送ツール起動画面

接続後、右側の Remote Host パネルに表示されるディレクトリは、データマネージャの引

数で指定されたディレクトリか起動ディレクトリとなります。但し、Remote Host を Local として Connect した時は、左側の Local Host のディレクトリと同じになります。

・Local と Remote 間のファイル転送機能

Local Host パネルあるいは Remote Host パネル内のファイルを選択して画面中央の矢印 ボタン (→: Send, ←: Receive) を押下する事によりファイル転送 (Send, Receive) が 可能です。

・Local、Remote での基本的なファイル操作

Local Host パネルあるいは Remote Host パネル内のファイル操作ボタンを押下する事に より基本的なファイル操作が可能です。



図 45: Transmitter 接続画面

・UDF のヘッダ情報の表示機能

Show Header Info のチェックボックスにより、ファイル選択時の UDF のヘッダ情報の表示が可能です。

10.2 Pythonツール

Tool/Python メニューを使用して、ピュアな(GOURMET 拡張を含まない) Python 環境を起動す ることができます。起動オプションは、Application Setup ツールで設定します。10.4 を参照 してください。

10.3 Gnuplot ツール

Tool/Gnuplot メニューを使用して、gnuplot アプリケーションを起動することができます。起 動オプションは、Application Setup ツールで設定します。10.4 を参照してください。

10.4 Application Setupツール

Python および gnuplot の起動アプリケーションの指定と、起動時の引数の指定を行います。 図 46 は Application Setup ツールダイアログを示しています。

重要:

起動時の引数のファイル名にブランクがあれば全体を""でくくる必要があります。 (設定例)

Python Application: "C:¥OCTA2010¥GOURMET_2010¥bin¥win32¥Python¥pythonw.exe" Python Argument: "C:¥OCTA2010¥GOURMET_2010¥bin¥win32¥Python¥Tools¥idle¥idle.pyw" Gnuplot Argument: "C:¥OCTA2010¥GOURMET_2010¥bin¥win32¥gnuplot¥wgnuplot.exe"

Application Setup		×
Python Application:	"D:¥UCIA2UU6¥GUUKMEI_2UU6¥bin¥win32¥Python¥pythonw.exe"	
Python Argument:	:¥OCTA2006¥GOURMET_2006¥bin¥win32¥Python¥Tools¥idle¥idle.pyw"	
Gnuplot Application:	"D:¥OCTA2006¥GOURMET_2006¥bin¥win32¥gnuplot¥wgnuplot.exe"	
Gnuplot Argument:		
	Cancel	ок

図 46: Application Setup

10.5 分子ビルダーツール

molfile フォーマットおよび PDB ファイルフォーマットのデータを COGNAC などの UDF ファイル

- にインポートする機能です。図 47 は分子ビルダーツールダイアログを示しています。 分子ビルダーツールは次の処理を行います。
 - ・molfileか PDB データファイルを、指定されたファイルフィルターで読み込む
 - ・結合角情報、二面角情報を自動生成する
 - ・これらのデータを入力 UDF ファイルの定義に従って変換する
 - ・変換後のデータを入力 UDF ファイルの既存データとマージする
 - ・上記の結果を指定された出力 UDF ファイルに保存する

MoleculeBuilder	×
Select File Filter	Type: molfile format Filter
Data File:	D:¥OCTA2006¥GOURMET_2006¥tutorial¥sample¥test_builder_ny6.mol
Input UDF:	D:¥OCTA2006¥GOURMET_2006¥tutorial¥sample¥test_builder.udf
OutPut UDF:	D:¥OCTA2006¥GOURMET_2006¥tutorial¥sample¥test_builder.udf
	Generate Electrostatic Site Select ES_Element Type: POINT_CHARGE Cancel OK

図 47: 分子ビルダー

10.6 起動環境設定ツール

起動環境設定ツールを使って GOURMET を起動するときの環境設定ファイル

(platform_win32.ini,platform_linux.ini)の内容を編集することができます。 設定できる環境変数は以下のとおりです:

- ・Python 本体のインストールディレクトリ (PYTHONHOME)
- ・Python スクリプトの検索ディレクトリ (PYTHONPATH)
- ・実行パス (PATH)
- ・ライブラリパス(Unix/Linux における LD_LIBRARY_PATH)
- ・エンジン等のグループ単位で以下の設定ができます:
 - アクションファイルの検索ディレクトリ(UDF_ACTION_PATH) Python スクリプトの検索ディレクトリ(PYTHONPATH)

インクルード UDF ファイル検索ディレクトリ(UDF_DEF_PATH) 実行パス(PATH)

ライブラリパス (Unix/Linux における LD_LIBRARY_PATH)

グループ単位で環境を設定する場合、グループの先頭ディレクトリ("Top directory")を1つ 指定することにより自動的に以下のディレクトリパスを起動環境に設定することができます:

アクションファイルの検索ディレクトリ(UDF_ACTION_PATH)

"Top directory"+"/action"

Python スクリプトの検索ディレクトリ (PYTHONPATH)

"Top directory"+"/python"

インクルード UDF ファイル検索ディレクトリ(UDF_DEF_PATH)

"Top directory"+"/udf"

実行パス (PATH)

"Top directory"+"/bin"

ライブラリパス(Unix/Linux における LD_LIBRARY_PATH)

"Top directory"+"/lib"

自動設定を使用せずに個別にグループ単位の環境設定をする場合は、"detail"にチェックをつけます。

また、環境変数内の下記キーワードを置き換える機能を持っています:

%ARCH%, \$ {ARCH} : アーキテクチャー名置き換え

%PF_FILES%, \${PF_FILES} : PF_FILES 環境変数に置き換え

%PF_ENGINE%, \${PF_ENGINE}: PF_ENGINE 環境変数に置き換え

%OCTA_DIRECTORY%, \${OCTA_DIRECTORY}: OCTA インストールパスに置き換え

GOURMET 起動時に環境設定ファイルを環境変数生成ツール(gourmet_init)が読み込んで起動 に必要な環境変数を生成します。環境変数生成ツールについては付録 G を参照してください。

図48および図49に起動環境設定ツールの画面例を示します。

∉ Environment	x
Python Path LibraryPath Group	
PYTHONHOME: %PF_FILES%\bin\%ARCH%\Python	•
	 =
	 •
CANCEL	

図 48: 起動環境設定ツールの Python 設定タブ

Environment		X		
Python Path Li	ibraryPath Group			
0 1				
Top directory:				
☑ detail Detail				
Action directory:	%PF_FILES%\action	%PF_FILES%\action		
Python directory:	_FILES%\python;%PF_FILES%\lib\%ARCH%			
UDF directory:	%PF_FILES%\udf			
PATH directory:	%PF_FILES%\bin\%ARCH%			
LD_LIBRARY_PATH: %PF_FILES%\lib\%ARCH%				
ADD DELETE				
	CANCEL			

図 49: 起動環境設定ツールの Group 設定タブ

A.1 Operating System

Operating System	Version	Comments	
Microsoft Windows-NT(Intel)	4.0 SP6	動作確認済	
Microsoft Windows 2000(Intel)	SP4	動作確認済	
Microsoft Windows XP(Intel)	SP1, SP2	動作確認済	
Microsoft Windows Vista		動作確認中	
RedHat Linux(x86)	9. 0J	動作確認済	
Red Hat Enterprise Linux (x86)	3	動作確認中	
Turbo Linux(x86)	10	動作確認中	
Fedora Core	3, 5	動作確認済	
Fedora Core x86_64	5	動作確認中(32 ビット動作)	
MacOS X Tiger	10. 4. 7	Intel Core Duo 2.16GHz	
		Power PC G5 Dual 2.5GHz	
		動作確認済	
MacOS X Panther	10. 3. 9	Power PC G4 1GHz	
		動作確認済	

表 2: Operating System

A.2 Java System

Java System	Version	Comments
SUN Java2 SE/RE	1. 4. 2	動作確認済
SUN Java2 SE/RE	1.5.0(5.0), 6	動作確認済

表 3: Java System

A.3 Graphics System

Graphics System	Version	Comments
OpenGL	enGL Mesa 1.3以降 Mesa 4.0.4以降	
JOGL JSR-231	1.0.0	動作確認済
JOGL JSR-231	1.1.0	Windows 動作確認中
		Linux カーネルバージョンに依存

A.4 Python System

Python System	Version	Comments
Python	1. 5. 2、 2. 1. x、 2. 2. x	動作確認済
Python	2. 3. x, 2. 4. x	動作確認済
Scientific Python	Pythonに依存	動作確認済
Numerical Python	Pythonに依存	動作確認済

表 5: Python System

Python 2.2.3 : Numerical Python 23.1, Scientific Python 2.4.5 Python 2.4.4 : Numerical Python 23.8.2, Scientific Python 2.4.11

付録B. エンジンマネージャのセキュリティーポリシーにつ いて

GourmetとEngineManagerは、クライアントとサーバの関係にあります。つまり、EngineManager はGourmetからの要求を待ち続けるサーバとして機能します。



通信経路としては、JavaのRMI (Remote Method Invocation)を使い、RPCライクなサービスとし てエンジンの起動、停止、UDFファイルの転送機能を提供します。ただし、認証機構の実装、Java からのファイルパーミッション制御については実装していません。

これらのサービスは実行速度を優先して設計されています。そのため、イントラネット内で実 行され、かつ、複数ユーザが競合して同じリソース(例:同じサーバの同じディレクトリを同時 刻に)を使わないことを想定しています。

そのため、以下のケースでリソースの競合、不具合が発生する可能性があります。

- ・同一ディレクトリで複数クライアントから実行しようとした場合
- ・実行中のエンジンの作業ディレクトリ内のファイルを削除、改変した場合
- ・その他、実行中のエンジンに影響を与える操作をサーバ上で行った場合

GourmetとEngineManagerとの接続単位でユニークなセッションキーを発行し、他に起動しているGourmetから動作中のエンジンを制御されることは防いでます。しかし、ファイルやディレクトリといったリソースにはアクセス可能であるため、上記のようなケースでは不具合が発生する可能性があります。

重要:

- ・エンジンマネージャはクライアントに対する認証を行いません。
- ・実行可能なモジュールに対して制限を設けていません。

このようなセキュリティー上の危険を避けるための方策として、以下が考えられます。

- ・システム管理者はエンジンマネージャを起動するための特別なユーザIDを準備し、必要な最 低限の権限を与えておく。
- ・もしも、OCTAシステムとGOURMETをローカルマシンでしか稼動させないならば、STAND ALONE
 MANAGERスタートアップオプションを使う。このオプションを使えば、他の誰もそのマシンを

エンジンサーバとしては使えなくなる。

 ・エンジンマネージャをMictosoft Windowsで稼動させる場合、エンジンマネージャが使うコ ミニュケーションポートのフィルタを行い、他のマシンからの接続を行えないようにする。
 これを行うためには、OS 付属のツールかPersonal Firewallなどのセキュリティー確保のための製品を使用することができる。

付録C. アクションの定義

C.1 アクションファイルの文法

アクションファイルの文法規約について以下に記述します。

・コメント

行の最初の文字が'#'ならば、その行全体がコメントとみなされます。

• import

import 文は、Python の import 文と同じです。

• action

action キーワードはアクション定義のターゲットとなる UDF データに対するアクションの 始まりに使用します。UDF データターゲットを指定しなければ、UDF ファイル全体に対する アクションとなります。

• autorun

autorun キーワードは UDF ファイルが開かれた時と、リロードされた時に実行されるアクションを定義するために使用します。autorun は複数定義でき、そのすべてが実行されますが その実行順序は規定されていません。

・ターゲット

アクションのターゲットとは、エディタやビューワでマウスによりポイントされる UDF デ ータオブジェクトです。2 つ以上のターゲットが定義された場合、ビューワにおける複数 ピッキングアクションとして呼び出され、実行されます。

・アクション名

アクション名はアクションが選択されたときにポップアップにリスト表示されます。

・アクションパラメータ

アクションパラメータはアクションに渡す引数です。アクションパラメータが定義された 場合、ユーザはアクションダイアログでパラメータの値を入力できるようになります。パ ラメータが文字列型であり'|'で区切られている場合、ユーザーはそれらの選択メニューか らひとつを選択できます。パラメータが文字列型であり、その値が[...]である場合、ユ ーザーはダイアログに表示された[...]位置をマウスで右クリックすることにより現れる ファイルオープンダイアログでファイルパスを選択できるようになります。

表 6 はアクションファイルの BNF (厳密な文法)を示しています。BoldFace の語は、キ ーワードで、UPPERCASE の語はトークンです。

```
構文規則の名前(非終端記号):構文規則の構成
statement:
        comment\_statement
        import\_statement
        actions
        ¥n
comment_statement:
        # any_statement ¥n
import_statement:
        {\tt import python\_module } {\tt {\tt Yn}}
actions:
        action target : action_statement ¥n
        autorun : action_statement ¥n
target:
        NOTHING
        UDF OBJECT
        target, UDF_OBJECT
action_statement:
        ACTION_NAME ( parameters ) : body
parameters:
        NOTHING
        parameters , PARAMETER_NAME
        parameters , PARAMETER_NAME = initial_value
initial_value:
        NUMBER
        "STRING"
        "string_selection"
        "filepath_selection"
string_selection:
        NOTHING
        STRING
        string_selection STRING
filepath_selection:
        [...]
body:
        python_function
        ¥begin python_statements ¥end
```

C.2 Python文の特別語

次の語は、アクション本体の Python 文の中で特別な意味を持ちます。

・parameter 名

Python 文の中で、アクションパラメータ名が使われた場合、アクションパーサはその語を パラメータの値 (ユーザが入力した値) に置換します。例1では、name が"time" かユー

表 6: Action ファイルの BNF 文法

ザの入力した文字列に置換されます。

 \cdot self

python 文の中で self が使用された場合、アクションパーサはその語 (self) をアクショ ンのターゲット名に置換します。例1では、self が Calculated_results.time に置換され ます。

例1: (ターゲット:Calculated_results.time、アクション名:add_trajectory)

```
action Calculated_results.time : add_trajectory(name="time") : ¥begin
```

try:

deleteSheetCol(name)

except RuntimeError: pass

createSheetCol(getSheetColSize(), name)

for rec in range(totalRecord()):

jump(rec)

```
setSheetData('time', rec, 'self')
```

¥end

2 つ以上のターゲットが定義されている場合、キーワード"self"+定義されているターゲット 数がこの置換プロセスに使用されます。ビューワの複数ピッキングモードにおいて、ユーザが最 初に

```
Set_of_Molecules.molecule[0].atom[1]
```

```
をピックし、次に
```

Set_of_Molecules.molecule[0].atom[5]

をピックした場合、

例2において、

self1 は Set_of_Molecules.molecule[0].atom[1] に置換され、

```
self2 は Set_of_Molecules.molecule[0].atom[5] に置換されるます。
```

```
例2: 複数ピッキングアクションのサンプル
```

```
action Set_of_Molecules.molecule[].atom[], Set_of_Molecules.molecule[].atom[]
: distance(option="print|draw") : ¥begin
pos1 = get('Structure.Position.mol[].atom[]', Location('self1').getIndex())
pos2 = get('Structure.Position.mol[].atom[]', Location('self2').getIndex())
ds = 0
for i in [0,1,2] : ds = ds + math.pow(pos1[i]-pos2[i], 2)
if option == "print":
    print 'Distance', $self1, $self2, ':', math.sqrt(ds)
```

```
else:
    line(pos1, pos2, 0)
    message = 'Distance:%f' % math.sqrt(ds)
    pos = [(pos1[0]+pos2[0])/2, (pos1[1]+pos2[1])/2, (pos1[2]+pos2[2])/2]
    text(pos, message, 0 )
```

¥end

重要:

アクションパーサは、通常の字句解析を用いて、"self[0-9]*"やパラメータ名にマッチする全 ての語を置換します。

したがって、これらの語("self[0-9]*")をアクション本体で使う場合には、十分に注意して ください。

C.3 アクションファイルのサンプル

```
例: チュートリアルより
```

```
# autorun
```

```
# This action will be excuted when the UDF is opened or reloaded.
```

```
autorun : initialize() : ¥begin
```

import gnuplot

```
# initialize GraphSheet
```

```
trajsize = totalRecord()
```

colname = ['time']

```
$GraphSheet[] = []
```

for i in range(getSheetColSize()):

```
deleteSheetCol(i)
```

```
i=0
```

```
for name in colname:
```

```
createSheetCol(i, name, trajsize)
```

```
i=i+1
```

¥end

```
# Non Target Action
```

```
# This action named 'clearDraw' will be executed when the user points UDF file icon in
Editor,
```

```
# or points background with [Ctrl] in Viewer.
action : clearDraw() : ¥begin
```

clearDraw()

jump(0)

¥end

Typical Action definition

This Action named ' setColor' will be executed when the user point ' Ball' in Editor. # Action Dialog requires to select one of the value for the parameter ' BallColor', # and the word 'BallColor' in the action body replaced to the selected color (say red). action Ball: setColor(BallColor="white|blue|green|red") : ¥begin

```
if BallColor == 'white':
    $Ball.color[] = [1,1,1]
elif BallColor == 'blue':
```

```
$Ball.color[] = [0,0,1]
elif BallColor == 'green':
$Ball.color[] = [0,1,0]
```

```
elif BallColor == 'red':
```

```
$Ball.color[] = [1,0,0]
```

¥end

C.4 UDFファイルとの関連付け

アクションファイルは UDF ファイルのヘッダー部の Action 項目にアクションファイル名が記述されていれば、UDF ファイル読み込み後に読み込まれます。

(例)Action 項目記述例

¥begin{header}
¥begin{header}
¥begin{def}
Action:string;
Comment:string;
¥end{def}
¥begin{data}
Action:"cognac_draw.act;cognac_info.act;cognac_plot.act;cognac_anal.act;cognac_edit.act"
Comment:"UDF definition file for COGNAC4.2"
¥end{data}
¥end{header}

GOURMET がアクションファイルを探索するディレクトリは、まず、UDF ファイルと同じディレクトリであり、次に GOURMET 起動時に引数として渡される UDF_ACTION_PATH 変数です。

UDF_ACTION_PATH 変数は、規定では以下のディレクトリになります: FP_FILES 環境変数ディレクトリ+"action" PF_ENGINE 環境変数ディレクトリ+"action"

付録D 汎用ファイルコンバーター

D.1 ファイルコンバーターとは

ファイルコンバーターは、専用の言語で書かれたフィルタルールを使って外部のソフトウェア で作成されたデータを UDF に変換するツールです。この章では、このフィルタールールの文法に ついて記述しています。

D.2 フィルタールールの文法

・コメント

行の最初の文字が'#'ならば、その行全体がコメントとみなされます。

・事前処理

行の先頭の語が'BEGIN'の時、フィルタリング処理の前に指定した Python 関数が呼ばれま す。AWK スクリプトと同様の機能です。

·事後処理

行の先頭の語が'END'の時、フィルタリング処理の後に、指定した Python 関数が呼ばれま す。AWK スクリプトと同様の機能です。

・数値

行の先頭の語が数値の時、フィルターは入力行から指定された数値の行数だけをスキップ します。

・制御ルール

行の先頭の語が'CONTROL'の時、その行はあるデータを何行読むかを指定する制御用データです。

FORTRAN で書かれた小さなプログラムがこの形式をよく使用します。制御ルールでは変数 (文字列)の並びを指定し、制御用データに記述されている数値を、その変数に読み込む 処理を行います。

・変数

行の先頭の語が上記の制御ルールで指定された変数と同じである時、フィルターは入力デ ータから変数の値と同じ行数を読み込む。

・ラベル

行の先頭の語が'LABEL'の時、その行はあるラベルを付けられたデータ行を読み込むための パターンマッチングデータです。SED や perl 等で書かれた多くの UNIX プログラムや CAD の pre/post プロセッサがこのデータ形式を使っています。2番目の語がマッチングパターンであり、SEDで使われる正規表現を用いています。

・フィールド幅

丸括弧() でくくられた数値は、データフォーマットの読み込みカラム数として使用され ます。数値が省略されるかまたは0を指定した場合、フィルターはホワイトスペースで区 切られたフリーフォーマットとして読み込みを行います。

表7 はフィルタールールファイルの BNF(厳密な文法)を示しています。BoldFace の語は、キ ーワードであり、UPPERCASE の語はトークンです。特別な変数(_NDATA)が、その時点で読み込 まれているデータ数として使われます。

line:

comment_line begin_proc skip_line control line fixed_line maching_line end_proc ¥n comment line: # any statement ¥n begin_proc: $BEGIN \ python_func_name \ \mbox{\sc y}n$ end_proc: END python_func_name ¥n skip_line: NUMBER ¥n control_line: CONTROL controls $\ensuremath{\mathtt{Yn}}$ controls: NOTHING controls VARIABLE_NAME field_width field width: NOTHING () (NUMBER)

fixed_line: VARIABLE_NAME format_rules ¥n format_rules: NOTHING format_rules UDF_PATH field_width format_rules UDF_PATH = value format_rules UDF_PATH = python_func_name matching_line: LABEL matching_pattern format_rules ¥n value: NUMBER "STRING" _NDATA 表 7: フィルタールールファイルの BNF 文法

D.3 ファイルフィルターの例

molfileファイルフィルターとPDFファイルフィルターが分子ビルダーツールで使用されますが、 これらは変換元データを、以下のUDFファイルのデータへ変換します。 以下の例で、(連続行) は次の行へ続いていることを示しています。(実際のデータではない)

```
¥begin{def}
class Vector3d:{x:float, y:float, z:float}
class Atom:{
   Atom_ID:int
   Atom_Type_Name:string
   Position:Vector3d
   Mol_ID:int
}
class Bond:{
   atom1:int
   atom2:int
}
atoms[]:Atom
bonds[]:Bond
```

 $\text{Hend}\{\text{def}\}$

```
例1: molfile フィルター
molfile フィルターは以下のルールにより、molfile 形式のデータを前述の UDF ファイル
へ読み込みます。
```

#*

#* Material Modeling Platform Package #* #* Copyright(c) 2000, The Japan Research Institute, Ltd. #* All rights reserved. #* #* \$Id: molfilerule.txt, v 1.1.1.1 2004/11/29 12:19:10 nishio Exp \$ #* #* # example rule for convert molecule data from molfile 3 CONTROL nAtom(3) nConnect(3) atoms[].Position.x(0) atoms[].Position.y() atoms[].Position.z() (連続行) nAtom atoms[].Atom_Type_Name() atoms[].Atom_ID=_NDATA atoms[].Mol_ID=-1 nConnectbonds[].atom1(3) bonds[].atom2(3) LABEL ^M. . CHG (6) LABEL ^M. . ISO(6) =END_PROC END 例2: PDB ファイルフィルター PDB ファイルフィルターは以下のルールにより、PDB ファイル形式のデータを前述の UDF フ ァイルへ読み込みます。 #* #* Material Modeling Platform Package #*

#* Copyright(c) 2000, The Japan Research Institute, Ltd. #* All rights reserved. #* #* \$Id: filter.tex,v 1.1 2002/02/21 12:56:12 nishio Exp \$ #* #* #* # example rule for convert molecule data from pdb file LABEL ^ATOM..(6) atoms[].Atom_ID(5) (1) atoms[].Atom_Type_Name(4) (6) (4) (連続行) (4) atoms[].Position.x(8) atoms[].Position.y(8) atoms[].Position.z(8) atoms[].Mol_ID=-1 LABEL ^HETATM(6) atoms[].Atom_ID(5) (1) atoms[].Atom_Type_Name(4) (6) (4) (連続行) (4) atoms[].Position.x(8) atoms[].Position.y(8) atoms[].Position.z(8) atoms[].Mol_ID=-1 LABEL ^GONECT(6) =PDB_conect END =END_PROC

例3: NASTRAN bulk ファイルフィルター NASTRAN bulk ファイルフィルターが MUFFIN で使われるますが、このフィルタは、NASTRAN bulk ファイル形式のメッシュデータをプラットフォームの共通 UDF のメッシュデータに変換するサ ンプルです。 (GRID、CTRIA3、CTETRA ラベルのみに対応する例)

```
#*
#* Material Modeling Platform Package
#*
#* Copyright(c) 2000, The Japan Research Institute, Ltd.
#* All rights reserved.
#*
#* $Id: filter.tex,v 1.1 2002/02/21 12:56:12 nishio Exp $
#*
#*
# example rule for convert mesh data from NASTRAN bulk file
LABEL GRID(8) mesh.data.vertex[].id(8) (8) mesh.data.vertex[].position.x(8) (連続行)
mesh. data. vertex[]. position. y(8) mesh. data. vertex[]. position. z(8)
LABEL CTRIA3(8) mesh.data.face[].id(8) mesh.partial_region[0].face[](8)
                                                                          (連続行)
v0(8) v1(8) v2(8) mesh.data.face[].vertex[]=[v0,v1,v2]
LABEL CTETRA(8) mesh. data.cell[].id(8) mesh.partial_region[0].cell[](8)
                                                                          (連続行)
v0(8) v1(8) v2(8) v3(8) mesh.data.cell[].vertex[]=[v0,v1,v2,v3]
```

付録E 専用コンバーターツール

E.1 NASTRANデータコンバーターツール

NASTRANバルクデータファイルを有限要素UDFファイルに読み込み、描画および表面抽出の処理 を行うユーティリティ群です。有限要素UDFファイル(fem_def.udf)、アクションファイルおよ びPythonスクリプトファイルは以下のディレクトリに収められています:

GOURMET_20XX/too1/NASTRAN

使用方法

(1) 起動方法

GOURMET を起動して、有限要素 UDF 定義ファイル (fem_def.udf) を読み込みます。データを 保存する場合は、必ず別名保存をして UDF 定義ファイルを変更しないように注意してください。

(2) NASTRAN バルクファイルのコンバート読み込み

編集画面のUDFファイル名部分を右クリックしてアクションリストを表示し、 [ReadBulk]を選択します。

≜ Editor - fem_def.udf [D:¥COMMON¥NASTRAN]						
<u>File E</u> dit <u>V</u> iew	<u>U</u> nit <u>P</u> ython <u>C</u>	ptions	<u>T</u> ool <u>W</u> ind	ow <u>H</u> elp		
Path History	View	Tabla	Location	Becord		
		Ianie		Record		
UDF Path:						-
	Name		Туре	Value	tU	nit
📘 fem_def.udf	Description			-	-	
👇 🚞 coordinat	DrawElement		Coordinate	-	-	
👇 🛄 node 🛛	ReadBulk		Node array	-	-	
element[]	WriteBulk		Element array	-	-	
property[]			Property array	-	-	
material[]			Material array	-	-	
🗢 🛄 group[]			RegionRes	-	-	

図 50: NASTRAN バルクファイル読み込みアクション選択

ReadBulk Argument Values 🛛 🗙				
Names	Values			
input_bulk_filename				
	Cancel OK			

図 51: NASTRAN バルクファイル指定画面

Values の下の入力領域を右クリックするとファイル選択ダイアログが表示されますので、読み込みたい NASTRAN バルクファイルを指定して、「OK」ボタンを押します。すでに NASTRAN バルクファイルを読み込んだ状態でさらに読み込み処理を行うと、以前のデータは全て削除されます。

(3) 要素の描画

要素を直接描画するためには、編集画面の"element[]"部分を右クリックしてアクションリストを表示し、[DrawElement]を選択します。

Editor – fem_def.udf [D:¥COMMON¥)	NASTRAN]		
<u>File Edit View Unit Python Options</u>	s <u>T</u> ool <u>W</u> indo	w <u>H</u> elp	
Path History View Tree O Table	Cocation Global	Record	
UDF Path: element[]			•
Name	Туре	Value	Unit
🗋 fem_def.udf		-	-
🗣 🚍 coordinate[]	Coordinate	-	-
👇 🚍 node 🛛	Node array	-	-
• 📑 element[]	Element array	-	-
←	Property array	-	-
🛉 👇 🚞 material[]	Material array	-	-
🖕 🚍 group[]	RegionRes	-	-

図 52: 要素の描画アクション選択

(4) 部分領域の抽出

編集画面の"group[]"部分を右クリックしてアクションリストを表示し、[CreateRegion]を選択します。

≜ Editor - fem_def.udf [D:¥COMMON¥NASTRAN]				
<u>File Edit View Unit Python Options</u>	<u>T</u> ool <u>W</u> ind	ow <u>H</u> elp		
Path History Cocation Tree O Table Global O Record				
UDF Path: group[-
Name	Туре	Value	Unit	
fem_def.udf		-	-	i
🗣 🚍 coordinate]	Coordinate	-	-	
🔶 🗂 node]	Node array	-	-	
🖕 🗂 element[]	Element array	-	-	
🔶 🗂 property[]	Property array	-	-	
🖕 🗂 material 🛛	Material array	-	-	
← 🗂 group[]	RegionRes	-	-	
CreateRegion DrawRegion				

図 53: 部分領域の抽出アクション選択

部分領域を抽出するための条件を"method"から選びます。"angle","property"および "angle_property"から選択することができます。"property"は、要素プロパティの値のみを使っ て領域分割を行います。"angle_property"の場合は、法線ベクトル角度条件と要素プロパティ両 方の条件で領域分割します。法線ベクトル角度条件のみで領域分割する"angle"の場合は法線ベ クトル角度(°)を入力してOKボタンを押します。

≜ CreateRegion Ar	gument Values 🛛 🗙
Names	Values
angle	60.0
method	angle 💌
	Cancel OK

編集画面で"group[]"データ以下を展開すると、"group[0]"が作成され、

"group[0].region[]"以下に部分領域が作成されていることが確認できます。

さらに編集画面の"group[]"部分を右クリックしてアクションリストを表示し、 [CreateRegion]を選択して部分領域抽出処理を行うと、"group[1]"が作成されて部分領域抽出デ ータが追加されます。

また、例えばインデックス付の"group[0]"部分を右クリックして部分領域抽出処理を行うと、

図 54: 部分領域の抽出条件指定画面

"group[0]"に今回行った部分領域抽出データが書き直されます。

2次元要素に対する部分領域抽出処理では、境界となる辺の集合を閉多角形となるように出力 します。

(5) 部分領域の描画

編集画面の"group[0]"部分を右クリックしてアクションリストを表示し、[DrawRegion]を選択 します。

≜ Editor - fem_def.udf [D:¥COMMON¥NASTRAN]					
<u>File E</u> dit <u>V</u> iew <u>U</u> ni	t <u>P</u> ython <u>O</u> ptions	s <u>T</u> ool <u>W</u> ind	ow <u>H</u> elp		
Path History	View • Tree O Table	Global	Record		
UDF Path: group[0]					-
Nam	1e	Туре	Value	Unit	
🗋 fem_def.udf			-	-	
🔤 📑 coordinate 🛛		Coordinate	-	-	
👇 🗂 node 🛛		Node array	-	-	
👇 🚞 element[]		Element array	-	-	
👇 🗂 property[]		Property array	-	-	
👇 🚞 material 🛛		Material array	-	-	
🛉 🕂 📑 group[]		RegionRes	-	-	
e 📑 group[0]		RegionResult	-	-	
– 🗴 commer	CreateRegion	string	ANGLE=60.0		
🔶 🚍 face 🛛	DrawRegion	RegionFace	-	-	
🕴 🛉 📑 region[]		PartialRegio	-	-	
📔 🔶 🚍 regio	n[0]	PartialRegion	-	-	
🔶 📥 regio	n[1]	PartialRegion	-	-	

図 55: 部分領域の描画アクション選択



図 56: 部分領域の描画結果(全体)

"group[1]"などを選択して描画すると、その領域分割全体が描画されます。

"group[1].region[5]"など"region[N]"を選択して描画すると、選択した部分領域のみが面で描 画され、残りは線で描画されます。



図 57: 部分領域の描画結果(部分選択)

(6) NASTRAN バルクファイルのコンバート書き出し

編集画面のUDFファイル名部分を右クリックしてアクションリストを表示して、 [WriteBulk]を選択します。

🚔 WriteBulk Argument Values 🛛 🗙		
Names	Values	
output_bulk_file		
Cancel OK		

図 58: NASTRAN バルクファイルの書き出し画面

Values の下の入力領域を右クリックするとファイル選択ダイアログが表示されますので、フ ァイル選択ダイアログで書き出すディレクトリに移動してファイル名を入力してからファイル 選択ダイアログを終了し、このダイアログで「OK」ボタンを押します。

UDF解説

トップレベルのデータ構成および内容は以下のとおりです。

coordinate[]:Coordinate	座標系データ
node[]:Node	節点データ
element[]:Element	要素データ
property[]:Property	要素プロパティデータ
material[]:Material	プロパティ物性データ
group[]:RegionResult	部分領域抽出処理データ

各下位データ構成は以下のとおりです。

应樗系	データ	'構成
上小ホル	. /	「日子ルス

class Coordinate:{	
id:ID	
system:{	
type:select{"NodeCartesian","N	IodeCylindrical","NodeSpherical",
"PointCartesian","	'PointCylindrical","PointSpherical"}
NodeCartesian:{	"節点 ID による直交座標系"
node1_id: <node,id></node,id>	
node2_id: <node,id></node,id>	
node3_id: <node,id></node,id>	
} "CORD1R"	
NodeCylindrical:{	"節点 ID による円筒座標系"
node1_id: <node,id></node,id>	
node2_id: <node,id></node,id>	
node3_id: <node,id></node,id>	
} "CORD1C"	
NodeSpherical:{	"節点 ID による球面座標系"
node1_id: <node,id></node,id>	
node2_id: <node,id></node,id>	
node3_id: <node,id></node,id>	
} "CORD1S"	
PointCartesian:{	"点座標を直接与えることによる直交座標系"
coordinate_id int	
a:{x1:double, x2:double, x3:do	puble }
b:{ x1:double, x2:double, x3:do	puble }
c:{ x1:double, x2:double, x3:do	puble }
} "CORD2R"	
PointCylindrical:{	"点座標を直接与えることによる円筒座標糸"
coordinate_id:int	
a:{x1:double, x2:double, x3:do	puble }
b?{x1:double, x2:double, x3:do	buble }
c:{ x1:double, x2:double, x3:do	buble }
} "CORD2C"	
PointSpherical	"点座標を直接与えることによる球面座標糸"
coordinate_id-int	11)
$a \cdot \{x \mid double, x 2 \cdot double, x 3 \cdot doubl$	buble }
$D \cdot \chi X I \cdot dOUDIE, X Z \cdot dOUDIE, X 3 \cdot dO$	
c.i x1.double, x2.double, x3.do	puble j
} "CORD2S"	

_}

座標系タイプの意味は以下のとおり:

- "NodeCartesian": NASTRAN の CORD1R レコードに対応し、3つの節点 ID により直交 座標系を指定する。
- "NodeCylindrical": NASTRAN の CORD1C レコードに対応し、3つの節点 ID により円 筒座標系を指定する。
- "NodeSpherical": NASTRAN の CORD1S レコードに対応し、3つの節点 ID により球面 座標系を指定する。
- "PointCartesian": NASTRAN の CORD2R レコードに対応し、3 つの点座標を直接与える ことにより直交座標系を指定する。与えた点座標に対する座標系を指定す ることができる。
- "PointCylindrical": NASTRAN の CORD2C レコードに対応し、3つの点座標を直接与えることにより円筒座標系を指定する。与えた点座標に対する座標系を指定することができる。
- "PointSpherical": NASTRAN の CORD2S レコードに対応し、3 つの点座標を直接与える ことにより球面座標系を指定する。与えた点座標に対する座標系を指定す ることができる。

ノードデータ構成

class Node:{	
id:ID	"節点 ID"
coordinate_id:int " <coordinate,id>"</coordinate,id>	"座標系 ID"
position:{	"節点座標"
x1:double	
x2:double	
x3:double	
}	
}	

要素データ構成

class Element:{	
type:select{"tri","quad","tetra","penta","hexa"}	"要素タイプ"
id:ID	"要素 ID"
pid: <property,id></property,id>	"プロパティ ID"
node[]: <node,id></node,id>	"節点 ID の配列"
}	

【解説】

要素のノード指定順序は NASTRAN に準拠して、下図のようになります:



図 59: 要素のノード指定順序

プロパティーデータ構成

class Property:{ id:ID mid:<Material,ID> dimension:int "2:shell,3:solid" values[]:string マテリアルデータ構成

```
class MaterialProperty:{
  name:string
  value:double
}
class Material:{
  id:ID
  E:double "Young modulus"
  G:double "Shear momulus"
  NU:double "Poisson ratio"
  RHO:double "Mass devsity"
  A:double "Thermal expation coefficient"
  Other[]:MaterialProperty
}
```

部分領域抽出処理データ構成

class RegionSideElement:{	
element_id: <element,id></element,id>	"面または辺の元要素 ID"
side_id:int	"面または辺の元要素に対する位置 ID【解説】参照"
}	
class RegionFace:{	
id:ID	"面または辺の ID"
node_id[]: <node,id></node,id>	"面または辺をなす節点 ID の配列"
element[]:RegionSideElement	"元要素に関するデータ"
}	
class PartialRegion:{	
color_id:int	"部分領域を最小限の色数で塗り分ける色 ID"
face_id[]: <regionface,id></regionface,id>	"部分領域をなす面または辺の ID の配列"
}	
class RegionResult:{	
comment:string	
face[]:RegionFace	"面または辺のデータ配列"
region[]:PartialRegion	"部分領域のデータ配列"
3	

【解説】

color_id は全体を最小の色数で塗り分けるための色インデックスです。 side_id は要素の辺または側面を指定するためのインデックスで、要素形状に応じてノード が下記のようになっている部分要素です。

例えば、四面体要素では side_id=0 は、要素内のノード順序{1,2,3}よりなる三角形です。

・三角形要素(サイドは辺)
side_id: [0]={1,2}, [1]={2,0}, [2]={1,2}

・四角形要素(サイドは辺)
side_id: [0]={0,1}, [1]={1,2}, [2]={2,3}, [3]={3,0}

・四面体要素(サイドは三角形面)
side_id: [0]={1,2,3}, [1]={0,3,2}, [2]={0,1,3}, [3]={0,2,1}

・五面体要素(サイドは三角形面または四角形面) side_id:[0]={0,2,1},[1]={3,4,5},[2]={0,1,4,3},[3]={1,2,5,4},[4]={0,3,5,2}

・六面体要素(サイドは四角形面)
[0]={0,4,7,3}, [1]={1,2,6,5}, [2]={0,1,5,4},
[3]={2,3,7,6}, [4]={0,3,2,1}, [5]={4,5,6,7}

付録F 描画対象の3次元移動(平行移動・回転移動)機能

3次元描画画面の特定の描画対象(Ver.4.0から点・線・球・円柱)について、直接描画座標 を変換する機能が追加されました。以下にこの機能について説明します。

F.1 準備するもの

3次元移動(平行移動・回転移動)機能を実現するために準備するものは2種類のアクションおよび UDF データと関連付けられた描画対象です。

2種類のアクションとは以下の2つです:

- (1) セレクトアクション(\$Select_\$Translate(), \$Select_\$Rotate())描画対象に関連付けられた UDF データ名から描画対象をセレクト状態にするアクション
- (2) リターンアクション(\$Translate(), \$Rotate())移動後の座標を UDF データに戻すアクション

上記2つのアクションサンプルが、GOURMET_20XX/action/cognac/cognac_transform.act にあ ります。サンプルには1アトムの移動(\$Translate_ATOM())、1モレキュールの移動 (\$Translate MOL())および回転(\$Rotate())があります。

COGNAC エンジンの描画アクションファイルにこのアクション文を挿入すれば、次に説明する3 次元移動(平行移動・回転移動)機能が操作できます。COGNAC の UDF ファイルを用いて移動ア クションを実行する時、境界条件を"NONE"にしておかないと正しい移動座標値が得られないこと があります。

F.2 操作手順

移動操作を行う前に、COGNAC エンジンの描画アクションファイル (cognac_draw.act) の最後 に3次元移動サンプルアクション (cognac_transform.act) 全体をコピーしておきます。

平行移動

COGNAC エンジンの UDF データを用いて平行移動操作手順を説明します。

- COGNAC エンジンの UDF データを開き、"show"アクションを選択して"ball-stick"で描画 します。
- (2)移動対象となる分子内の1つのアトムを選択します。
- (3) アトム選択時のアクション選択メニューで"\$Translate_MOL"を選択します。(図 60)
- (4)上記「セレクトアクション」が実行され、分子全体が選択された状態になり、平行移動操 作ダイアログが表示されます。(図 61)
- (5) 平行移動操作ダイアログには、2種類の画面があります。1つは"Direction"タブ画面で、 画面内の"Up"/"Down"/"Left"/"Right"のボタンを押すと、それぞれ現在見えているビュー の上・下・左・右方向に指定長さ("Length")分だけ平行移動します。 もう一つは、"Vector"タブ画面で、移動ベクトル量を入力する画面です。 また、平行移動操作ダイアログが表示されている間、コントロールキー+マウス左ボタン を押したまま描画画面をドラッグすると、選択対象がマウスと同じ方向に移動します。 移動量は描画画面上での1ピクセル移動量が、ボタンによる移動量の10分の1となりま す。
- (6) "Direction"タブ画面では、各方向ボタンを押すとその分だけ平行移動します。"Vector" タブ画面では Apply ボタンを押すと入力ベクトル量分平行移動します。
- (7) OK ボタンを押すと最後の描画状態で平行移動操作が確定し、移動後の座標を UDF データ に戻す「リターンアクション」が実行されます。
- また、移動操作中でも通常の視点の回転・移動・拡大・縮小を行えます。

回転移動

COGNAC エンジンの UDF データを用いて回転移動操作手順を説明します。

- (1) COGNAC エンジンの UDF データを開き、"ball-stick"で描画します。
- (2) 描画されたビューでマルチピッキングモードにします。
- (3)回転の軸となる2つのアトムを選択します。
- (4) 2つ目のアトム選択時のアクション選択メニューで"\$Rotate"を選択します。(図 62)
- (5)上記「セレクトアクション」が実行され、2つ目のアトム側のアトムおよびボンドが選択 された状態になり、回転操作ダイアログが表示されます。(図 63)
- (6)回転操作ダイアログには、回転の中心座標と回転軸ベクトルが示されます。回転の中心座標は、1つ目のアトム座標、回転軸ベクトルは1つ目のアトムから2つめのアトムへのベクトルです。(図 64)
- (7)回転の中心座標、回転軸ベクトルおよび回転量(度)を編集することができます。
- (8)回転量(度)を入力して回転矢印方向ボタンを押すと回転後の座標で図形が描画されます。

(9) OK ボタンを押すと最後の描画状態で回転操作が確定し、上記「リターンアクション」が 実行されます。





図 60: 平行移動アクション選択

b \$Translat	te_MOL	
Direction	Vector	
🔶 Left	tup → Up	➡ Right
Length		1.0
Cancel		ок

図 61: 平行移動操作ダイアログ



図 62: 回転アクション選択

👍 \$Rotate	×
ORIGIN X	19.92060089111328
ORIGIN Y	7.959270000457764
ORIGIN Z	5.228789806365967
AXIS X	1.0728988647460938
AXIS Y	-0.2032299041748047
AXIS Z	1.085890293121338
ANGLE(degree)	30.0
ANGLE 🔾 15 💿 30	O 45 Other
~	4
Cancel	ОК

図 63: 回転操作ダイアログ



図 64: 回転中心と回転軸ベクトル
付録G 環境変数生成ツール

環境変数生成ツール (gourmet_init) は、C/C++で作成された実行モジュールであ り、%PF_FILES%/bin/%ARCH%に置きます。起動シェル (バッチ) ファイルから起動されると、 PF_FILES 環境変数が示すディレクトリにある環境設定ファイル (platform_win32.ini または platform_linux.ini)を読み込んでオプション引数に応じた環境変数を生成して起動シェルに返 します。

環境変数生成ツールのオプション引数は以下のとおりです:

arch :マシンアーキテクチャー名を返します。

(win32 あるいは、"uname -s"コマンドが返す小文字名)

- python.home : Python ホームディレクトリを返します。
- python.path : Python パスを連結して返します。
- action.path : アクションパスを連結して返します。
- udf.path : UDF パスを連結して返します。
- java.home : Java ホームディレクトリを返します。
- java. java : Java 実行モジュールパスを返します。
- user.path : 実行パスを連結して返します。
- library.path : ライブラリパスを連結して返します。
- -a:マシンアーキテクチャー名をセットします。
- -f:環境設定ファイル名をセットします。("platform_win32.ini"など)
- -j: Java ホームディレクトリまたは Java ホームディレクトリの検索パスをセットします。

例えば、C:¥OCTA20XX¥GOURMET_20XX¥bin¥win32¥jre1.5 を渡すと実際に存在する C:¥OCTA20XX¥GOURMET_20XX¥bin¥win32¥jre1.5.0_08を返します。

-u:パスを渡すとディレクトリ部分を返します。

オプション引数なし:上記の-オプション以外の戻り値一覧を表示します。

環境変数生成ツールを引数なしで実行した場合の出力例を示します:

gourmet_init -f platform_win32.ini

arch=win32

python.home=C:\u00e40CTA20XX\u00e4GOURMET_20XX\u00e4bin\u00e4win32\u00e4Python

python.path=C:\u00efCourseller_20XX\u00efGourseller_20X\u00efGourseller_20X\u00e

CTA20XX¥PF_ENGINE_20XX¥python;C:¥OCTA20XX¥PF_ENGINE_20XX¥1ib¥win32

action.path=C:\u00e40CTA20XX\u00e4GOURMET_20XX\u00e4action;C:\u00e40CTA20XX\u00e4PF_ENGINE_20XX\u00e4actionudf.pa

th=C:¥OCTA20XX¥PF_ENGINE_20XX¥udf

java.home=C:\u00efCTA20XX\u00efGOURMET_20XX\u00efbin\u00efwin32\u00efjre1.5.0_08

java.java=C:\u00e40CTA20XX\u00e4G0URMET_20XX\u00e4bin\u00e4win32\u00e4jre1.5.0_08\u00e4bin\u00e4java.exe

user.path=C:\u00e40CTA20XX\u00e4G0URMET_20XX\u00e4bin\u00e4win32\u00e3C:\u00e40CTA20XX\u00e4PF_ENGINE_20XX\u00e4bin\u00e4win32\u00e4

library.path=C:\u00e40CTA20XX\u00e4GOURMET_20XX\u00e4lib\u00e4win32;C:\u00e40CTA20XX\u00e4PF_ENGINE_20XX\u00e4lib\u00e4win3
2

なお、環境変数生成ツールが返すパスは実際にディレクトリが存在するパスしか返しません。 また、環境設定ファイル内の下記キーワードを置き換えて返すことができます:

%ARCH%, \$ {ARCH} : アーキテクチャー名

(Windows の場合は win32、Unix/Linux の場合は"uname -s"コマンドが返す小文字名) %PF_FILES%, \${PF_FILES} : PF_FILES 環境変数(GOURMET のインストールディレクトリパス) %PF_ENGINE%, \${PF_ENGINE} : PF_ENGINE 環境変数(エンジン群のインストールディレクトリパス) %OCTA_HOME%, \${OCTA_HOME} : OCTA_XXXX_HOME 環境変数

(OCTA インストールパスディレクトリパス)

付録H トラブルシューティング

H.1 Pythonスクリプティング

- ・Python 環境を起動できないのですが...
 - インストールが正常に行われていれば、Python パッケージがインストールされ、そのパスが Tool/Application Setup ダイアログに表示されています。これが表示されていなかったり、インストール後に GOURMET の位置を移動して Python パッケージのパスが異なっているなどの場合、手動で正しいパスを設定してください。

H.2 3Dオブジェクトの描画

- ・3D オブジェクトを描画できません。または、描画しようとすると画面がおかしくなります。
 - 比較的新しいマシンの場合は、ディスプレイドライバを更新してみてください。
 - Java をご自身でインストールされた場合、JOGL ライブラリがインストールされていません。JOGL JSR-231 (1.0.0)をダウンロードしてインストールしてください。
 - インストールマニュアルのトラブルシューティングを参照してください。
- ・Linux x86_64上で GOURMET は起動しますが、描画できません。
 - GOURMET は x86_64 上でも 32 ビットモードで動作します。
 x86_64 Linux には 32 ビットモード用のライブラリが用意されてるため GOURMET は起動します。しかし 32 ビット用の GL ライブラリは用意されていない場合があるため描画できないことがあります。
 具体的には、/usr/lib64/lib6L.so.1 があっても、/usr/lib/lib6L.so.1 は無い場合があ

ります。この場合は、32 ビット用の GL ライブラリ(GL および GLU)をインストールして ください。

H.3 プロットの作成

- ・gnuplot アプリケーションが起動しないのですが。
 - インストールが正常に行われていれば、gnuplot パッケージがインストールされ、そのパ スが Tool/Application Setup ダイアログに表示されています。これが表示されていなか

ったり、インストール後に GOURMET の位置を移動して gnuplot パスが異なっているなどの 場合、手動で正しいパスを設定してください。

H.4 データの編集

- Linux を使っていますが、GOURMET と他のアプリケーションの間でコピーペーストできません。
 Java1.4 からサポートされるようになり、この症状は解決されました。
- ・Ver.4.0からテーブル編集画面でセルを複数選択できなくなりました。
 - Java のバージョンアップにより選択方法が変わりました。セルを複数選択するときは、 一つ目のセルを選択してから、二つ目のセルをシフトキーを押しながら選択します。する と一つ目のセルと二つ目のセルの間の矩形領域にある複数セルが選択状態になります。