In-Situ PBVR(v2.2)マニュアル

2023 年 11 月

国立研究開発法人日本原子力研究開発機構

システム計算科学センター

改訂履歴

版数	改定日	章番号	改版内容
1.0	2019.03.28	-	新規
1.1	2021.05.01		クライアント・サーバ型 PBVR と共通の GUI に修正
1.2	2022.03.30		ポリゴン合成機能の追加
2.0	2023.01,01		クライアントのソースコードを CS-PBVR と統合 伝達関数で利用できるカラーマップを変更
2.1	2023.04.20		非構造格子向けライブラリに対応要素を追加

目次

1 はじめに 4

1.1. 粒子サンプラの概要 5
1.2. デーモンの概要 5
1.3. PBVR クライアントの概要 6
1.4. 動作環境 6

2 パッケージ構成 7

2.1. ロードモジュールパッケージ 7
2.2. ソースコードパッケージ 8

3 ソースパッケージのビルド 9

3.1. デーモン、粒子サンプラ 9
3.2. PBVR クライアント 10

4 In-Situ 可視化のセットアップ 11

4.1. 環境変数の設定 11
4.2. 可視化パラメータの設定 11
4.3. ポートフォワード接続 11
4.3.1. 2点間リモート接続 12
4.3.2. 多段リモート接続 12
4.4. デーモンの起動とポートフォワード接続 12

5 粒子サンプラ 14

5.1. 構造格子向けの粒子生成関数 14
5.2. 非構造格子向けの粒子生成関数 15
5.3. 階層格子向けの粒子生成関数 16
5.4. 粒子サンプラの組込み 17
5.4.1. 領域情報の定義と粒子生成関数の追加 18

6 PBVR クライアント 19

6.1. 起動方法 19
6.2. 終了方法 19
6.2.1. 強制終了 20
6.3. GUI の構成と操作方法 21
6.3.1. ビューワ 21
6.3.2. ツールバー 23
6.3.3. 伝達関数エディタ 29
6.3.4. タイムステップ制御パネル 46
6.3.5. 粒子・ポリゴンの統合表示 47

６.３.６. 画像ファイル作成 51
６.３.７. レジェンドパネル 56
６.３.８. ビューワ制御パネル 58
７ サンプルの実行 59
７.１. プログラム実行が許可されたログインノード 59
７.２. 対話ジョブ 60
７.３. Windows 62

1 はじめに

本書は日本原子力研究開発機構システム計算科学センターで開発した In-Situ による遠隔可視化フレームワーク In-Situ PBVR のマニュアルである。昨今のスーパーコンピュータでは演算速度が I/O 速度を大幅に上回るため、計算結果の出力が困難になった。その結果、遠隔地にあるストレージ上の計算結果を手元の PC に転送して可視化する従来の可視化手法は適用が困難になった。In-Situ 可視化は、シミュレーションに結合された可視化プログラムが計算と同時に可視化処理を実行することで大規模なデータ I/O を回避し、確実に可視化画像を生成することができる。ポリゴンベースによる従来の手法では対話的な In-Situ 可視化が難しかったが、In-Situ PBVR は粒子ベースによる可視化と対話的 In-Situ 制御技術により対話的な可視化を可能にしたフレームワークである。

In-Situ PBVR は C++で開発され，京都大学小山田研究室で開発された粒子ベースのボリュームレンダリング手法である PBVR（Particle Based Volume Rnderring）法，および，可視化ライブラリ KVS を用いて構築されている。本フレームワークは粒子サンプラ、デーモン、PBVR クライアントの３つのコンポーネントで構成されている（図 1-1）。

（１）粒子サンプラ

粒子サンプラはシミュレーションコードに結合され、計算と同じ環境で粒子を生成する可視化ライブラリである。シミュレーション＋In-Situ 可視化のコードを構築するには、粒子サンプラが提供する可視化用関数に計算結果の配列を渡しシミュレーションコードに挿入する。粒子サンプラはストレージ上の可視化パラメータファイルを参照して、各タイムステップの計算結果を圧縮された可視化用粒子に変換し、ストレージ上に出力する。粒子ファイルは各プロセスから分散して出力される。

（２）デーモン

デーモンはログインノードあるいは対話ジョブ上で動作し、ストレージ上の粒子ファイルと PBVR クライアントから送信されてくる可視化パラメータを仲介する機能をもち、対話的 In-Situ 制御の要となるアプリケーションである。デーモンはストレージ上のファイルを監視して分散出力された粒子ファイルを集約し、ネットワークを介してユーザ PC に転送する。またデーモンは PBVR クライアントから送信されてくる可視化パラメータを受信し、粒子サンプラが参照する可視化パラメータファイルとして出力する。この動作はシミュレーションと非同期に実行されるため、シミュレーションを阻害せずに対話的制御が可能である。

（３）PBVR クライアント

PBVR クライアントはユーザ PC 上で動作し、可視化結果を表示するビューワと可視化パラメータを編集する GUI を提供する。PBVR クライアントはポートフォワードを使用してデーモンと通信しする。PBVR クライアントはデーモンからの粒子データを受信してビューワ上にボリュームレンダリング画像を表示し、ユーザが GUI 上で編集した可視化パラメータをデーモンに送信する。

In-Situ PBVR には、様々なシミュレーションに汎用的に対応可能な多変量可視化機能が実装されている。３次元データの可視化では計算結果のデータに色と不透明度を割り当てる。そして、その関係を記述する関数を伝達関数と呼ぶ。従来の可視化では一つの物理値に対して、色・不透明度を割り当てる

１次元伝達関数が利用されていた。しかし従来の可視化手法では多変量向けの多次元伝達関数を設計することが困難であった。In-Situ PBVR は代数式により多次元伝達関数を設計できる伝達関数エディタを提供している。伝達関数エディタでは各変量の１次元伝達関数を編集し、それらを変数として任意の代数式による多次元伝達関数を記述できる。この代数式では基本的な演算子や初等関数、微分演算子が利用可能である。

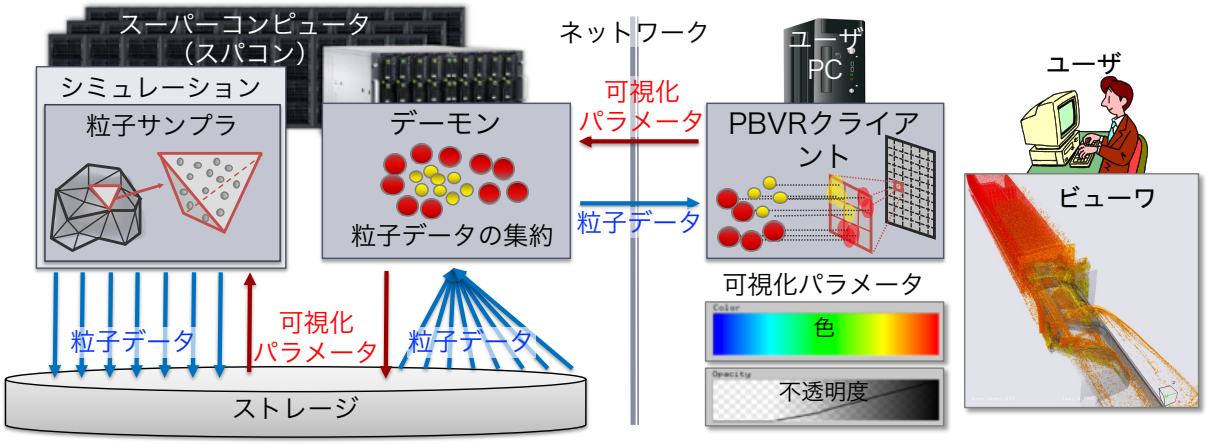


図　1-1　In-Situ PBVR フレームワークの全体構成

1.1. 粒子サンプラの概要

粒子サンプラは、MPI/OMP のハイブリッドプログラミングモデルで並列化され、更に SIMD 演算を利用することで可視化用粒子の生成を加速させている。粒子サンプラは、シミュレーションで用いる領域分割を変えることなく MPI 並列化され、各部分領域で OpenMP による要素並列で粒子を生成する。多変量可視化で必要となる物理値の合成や伝達関数の合成、そして物理値の補間計算が SIMD 演算の利用によりベクトル化されている。

粒子サンプラはシミュレーションで利用する格子の種類に合わせて、構造格子向け、非構造格子向け、そして階層格子向けの３種類が利用可能である。粒子サンプラは C／C++そして、FORTRAN で記述されたシミュレーションコードをサポートしている。In-Situ 可視化するために、粒子サンプラはシミュレーションのタイムステップループに挿入される。この時、シミュレーション結果の多変量データ、格子データ、そして領域の全体座標が粒子サンプラの引数として入力される。また、多変量データのメモリレイアウトは配列構造体（SOA）を仮定している。

1.2. デーモンの概要

デーモンはスーパーコンピュータの対話ノードあるいは対話ジョブで実行され、バッチ処理実行時の対話的可視化を実現するための要となる。デーモンはストレージ上のファイル情報を常に監視しており、粒子サンプラが出力した粒子ファイルを収集する。この収集作業は粒子サンプラ及びシミュレーションと非同期であるために、シミュレーションのパフォーマンスを阻害しない。粒子ファイルの収集は OpenMP で並列化されていて、デーモンは読み込んだ粒子データを一つの粒子データに集約して PBVR クライアントに送信する。また PBVR クライアントから送信された可視化パラメータを受信し、粒子サンプラが使用するストレージ上の可視化パラメータファイルを更新する。

In-Situ PBVR ではデーモンと PBVR クライアントはインターネットを介したソケット通信でデータを送受信するため、対話ノードとユーザ PC がポートフォワードで接続される必要がある。そのため、ポートフォワードが許可されない運用のスーパーコンピュータではデーモンが動作せず、対話的可視化が利用できない。またバッチ処理終了時までファイル出力が行われないステージングによる I/O を採用したスーパーコンピュータでもまた対話的可視化が利用できない。

1.3. PBVR クライアントの概要

PBVR クライアントはユーザ PC 上で起動され、可視化結果を表示するための画面、伝達関数を編集するための伝達関数エディタから構成されている。伝達関数エディタでは、多変量可視化を実現するための機能である“伝達関数合成器”（TFS）が実装されている。TFS は、結果データに含まれる変量を組み合わせ新しいボリュームデータを生成するボリュームデータ合成機能と、複数の伝達関数を組み合わせる伝達関数合成機能を備えている。ユーザは TFS 上で代数式により合成関数を指定する。代数式は可視化パラメータとして粒子計算モジュールに転送され、数式処理機能によりリアルタイムにボリュームデータ・伝達関数の合成が行われる。ユーザは TFS 上の代数式として、初頭的な数学関数や、物理変数の空間微分が利用可能であり、それらを組み合わせて自在に数式を構築可能である。

1.4. 動作環境

In-Situ PBVR はクロスプラットフォームのプログラムであり、富岳や機構所有の SGI8600 等様々なスーパーコンピュータで動作する。下表に動作環境を示す。

表 1-1 PBVR クライアント

プラットフォーム	OS	コンパイラ
Linux	Ubuntu18	g++
Mac	OSX12	clang
Windows	Windows10	MSVC

表 1-2 デーモン

プラットフォーム	OS	コンパイラ
Linux	RedHat	g++

表 1-3 粒子サンプラ

プラットフォーム	CPU (アーキテクチャ)	コンパイラ
富岳	A64FX (ARM)	富士通コンパイラ
Intel 系 CPU スパコン	Intel Xeon Phi (Knights Landing, KNL)	Intel コンパイラ
SGI8600	Intel Xeon Gold	Intel コンパイラ

2 パッケージ構成

CCSE の公開ページ(https://ccse.jaea.go.jp/software/In-Situ_PBVR/)から In-Situ PBVR のソースコードおよびロードモジュール（バイナリ）のパッケージが利用可能である。PBVR クライアント、デーモン、粒子サンプラはそれぞれユーザ PC、ログインノード、計算ノードで独立にビルドされ、連携して動作する。

2.1. ロードモジュールパッケージ

ロードモジュールに関して、Windows、Linux、Mac 用に構築された PBVR クライアントと、Linux サーバ用に構築されたデーモン及び粒子サンプラを提供する。粒子サンプラおよびデーモンは SGI8600 および富岳（clang-mode）上で構築された。粒子サンプラは以下の３つのライブラリから構成される。

（１） 粒子生成機能を提供する粒子生成ライブラリ
（２） 可視化機能を提供する KVS ライブラリ
（３） 代数式処理機能を提供する数式処理ライブラリ

さらに粒子生成ライブラリは、３種類のシミュレーションの格子構造（構想格子、非構造格子、階層型格子）に対応している。下表にロードモジュールの一覧を示す。

表 2-1 PBVR クライアント

機種	並列化	パッケージ名
Linux	OpenMP	pbvr_client_linux.tar.gz
Mac	OpenMP	pbvr_client_mac.tar.gz
Windows	OpenMP	pbvr_client_win.zip

表 2-2 デーモン

機種	並列化	ロードモジュール名
SGI8600	OpenMP	pbvr_daemon_s86.tar.gz
富岳	OpenMP	pbvr_daemon_fugaku.tar.gz

表 2-3 粒子サンプラ

機種	並列化	ロードモジュール名
SGI8600	MPI+OpenMP	pbvr_sampler_s86.tar.gz
富岳	MPI+OpenMP	pbvr_sampler_fugaku.tar.gz

2.2. ソースコードパッケージ

ソースコードパッケージに In-Situ PBVR の粒子サンプラ、デーモン、そして PBVR クライアントが含まれる。ソースコードパッケージを構成するソースツリーを表 2-4 に示す。ソースコードパッケージには粒子サンプラ結合済みのテストシミュレーションコードが含まれている。

表 2-4 In-Situ PBVR のソースツリー

ディレクトリ・ファイル	説明
DaemonAndSampler/	In-Situ PBVR のルートディレクトリ
|-pbvr.conf	make の設定ファイル
|-Makefile	サンプラ、デーモン、クライアントのコンパイルする
|-arch/	各環境のコンパイラの設定ファイル
|-Client/	PBVR クライアントプログラム
|-Common/	プロトコル、通信、共通ライブラリ
|-Daemon/	デーモンプログラム
|-Example/	テストシミュレーションコードのサンプル
|C/	C 版
|-Hydrogen_struct/	構造格子テストプログラム
|-Hydrogen_AMR/	階層構造格子テストプログラム
|-Hydrogen_unstruct_tetra/	四面体要素非構造格子テストプログラム
|-Hydrogen_unstruct_hex/	六面体要素非構造格子テストプログラム
|-Hydrogen_unstruct_prism/	プリズム要素非構造格子テストプログラム
|-Hydrogen_unstruct_pyramid/	ピラミッド要素非構造格子テストプログラム
|-Hydrogen_unstruct_quadtetra/	四面体二次要素非構造格子テストプログラム
|-Hydrogen_unstruct_quadhex/	六面体二次要素非構造格子テストプログラム
|-Fortran/	Fortran 版
|-Hydrogen_struct/	構造格子テストプログラム
|-Hydrogen_AMR/	階層構造格子テストプログラム
|-Hydrogen_unstruct	非構造格子テストプログラム
|-FunctionParser/	数式処理ライブラリ
|-InSituLib/	粒子生成ライブラリ
|-struct/	構造格子向け粒子生成ライブラリ
|-AMR/	階層構造格子向け粒子生成ライブラリ
|-unstruct/	非構造格子向け粒子生成ライブラリ
|-KVS/	可視化ライブラリ KVS

3 ソースパッケージのビルド

PBVR クライアントはユーザ PC 上で、デーモンと粒子サンプラはスーパーコンピュータ上でソースコードをビルドすることで生成される。

3.1. デーモン、粒子サンプラ

ソースパッケージからデーモンおよび粒子サンプラのライブラリをビルドする手順を示す。以下の手順では、ソースパッケージがスパコン上にダウンロード済みであり、ファイルのダウンロードパスを $HOME とする。

① is_pbvr_1.10.tar.gz を解凍する。

```
$ cd $HOME
$ tar xvfz is_pbvr_1.10.tar.gz
```

② コンフィグファイルを編集する。

ビルドに使用するコンパイラは pbvr.conf を環境に合わせて編集することで制御される。pbvr.conf 内で指定している変数の概要を表 3-1 に、変数の値として利用できるコンパイル設定ファイルの一覧を表 3-2 に示す。

表 3-1 pbvr.conf の変数一覧

変数	入力値	説明
PBVR_MACHINE	文字列	arch 配下のコンパイル設定ファイルを指定

表 3-2 コンパイル設定ファイル一覧

ファイル名	説明
Makefile_machine_gcc	gcc による逐次版コンパイルの設定
Makefile_machine_gcc_omp	gcc による OpenMP 版コンパイルの設定
Makefile_machine_gcc_mpi_omp	gcc による MPI+OpenMP 版コンパイルの設定
Makefile_machine_intel	intel による逐次版コンパイルの設定
Makefile_machine_intel_omp	intel による OpenMP 版コンパイルの設定
Makefile_machine_intel_mpi_omp	intel による MPI+OpenMP 版コンパイルの設定
Makefile_machine_fugaku_clang	富岳(ARM)clang モードのコンパイル設定
Makefile_machine_fugaku_trad	富岳(ARM)trad モードのコンパイル設定
Makefile_machine_s86_mpi_omp	機構 SGI8600(Intel Xeon Gold)向けのコンパイルの設定 Intel コンパイラと mpt ライブラリを使用

```
$ cd $HOME/is_pbvr
$ cat pbvr.conf
PBVR_MACHINE=Makefile_machine_gcc_mpi_omp
PBVR_MAKE_CLIENT=0
```

③　make によってデーモンと、粒子サンプラを構成するライブラリ群がビルドされる。

```
$ cd $HOME/is_pbvr
$ make all_clean
$ make
```

表 3-3　make によって生成されるファイル

ディレクトリ名	ファイル名	説明
KVS	libkvsCore.a	KVS ライブラリ。粒子データフォーマットや可視化機能を提供する。
Common	libpbvrCommon.a	通信ライブラリ。ソケット通信用のプロトコルを提供する。
Daemon	pbvr_daemon	デーモン
FuctionParser	libpbvrFunc.a	数式処理ライブラリ。数式処理の機能を提供する。
InsituLib/struct	libInSituPBVR.a	構造格子向け粒子生成ライブラリ
InsituLib/unstruct	libInSituPBVR.a	非構造格子向け粒子生成ライブラリ
InsituLib/AMR	libInSituPBVR.a	階層格子向け粒子生成ライブラリ

④　以上でデーモンおよび粒子サンプラのビルドは完了であるが、続けて Example 以下にあるテストコードのビルド方法を示す。

```
$ cd $HOME/is_pbvr/Example/C/Hydrogen_struct
$ make
```

3.2. PBVR クライアント

PBVR クライアントはクライアント・サーバ型 PBVR (https://ccse.jaea.go.jp/software/PBVR/)と共通のコードで構築されている。In-Situ 向けに PBVR クライアントをビルドするために、qtpbvr.conf を以下のように編集して PBVR_MODE=IS を有効にして下さい。

```
#PBVR_MODE - Either CS (ClientServer), or IS (Insitu) - Needed on all
platforms
#PBVR_MODE = CS
PBVR_MODE = IS
```

PBVR クライアントのビルド方法の詳細は、クライアント・サーバ型 PBVR のマニュアルを参照してください。

4 In-Situ 可視化のセットアップ

シミュレーションに結合された粒子サンプラ、対話ノード上で動作するデーモン、そしてユーザ PC 上の PBVR クライアントが連携することで、バッチ処理されるシミュレーションが対話的に可視化される。これを実現するためには、幾つかの簡単な設定とポートフォワード接続が必要になる。

4.1. 環境変数の設定

デーモンと粒子サンプラは以下の表 4-1 に示す環境変数を利用しており、実行時には export コマンドで環境変数を設定する必要がある。

表 4-1 デーモンおよび粒子サンプラで参照する環境変数

環境変数名	説 明
VIS_PARAM_DIR	伝達関数ファイル（可視化パラメータ）の配置されるディレクトリ※1
PARTICLE_DIR	疑似シュミレーションコードが粒子データを出力するディレクトリ※1
TF_NAME	伝達関数のファイル名（拡張子は含まない）※2

※1 指定が無い場合、デーモンおよび粒子サンプラは各々が実行されているカレントディレクトリを検索する。

※2 指定が無い場合、デーモンおよび粒子サンプラは伝達関数名として default.tf を採用する。

4.2. 可視化パラメータの設定

粒子サンプラがシミュレーション結果のボリュームデータを可視化用粒子データに変換する際に、伝達関数や可視化する物理値のレンジ、画面解像度等の可視化パラメータが必要とされる。可視化パラメータはその各項目が伝達関数ファイルの中にタグベースで記述される。ユーザはソースパッケージの Example の中に格納してある default.tf を初起動時の伝達関数ファイルとして利用できる。

In-Situ PBVR を実行するために、ユーザは伝達関数ファイルを表 4-1 の VIS_PARAM_DIR で指定されるディレクトリに TF_NAME で指定されるファイル名で配置する必要がある。

ユーザはデーモンと PBVR クライアントを起動することで GUI により伝達関数ファイルの内容を編集できる。環境変数で指定された伝達関数ファイルはデーモンに読み取られ、PBVR クライアント上に内容表示され、編集後上書きされる。

4.3. ポートフォワード接続

デーモンと PBVR クライアントの間で粒子データや可視化パラメータはソケット通信により送受信される。遠隔地のスーパーコンピュータ上の対話ノードと手元の PC の間でソケット通信するために、ユーザは双方のポート間を ssh ポートフォワーディングで接続する必要がある。

４.３.１．２点間リモート接続

手元の machineA と遠隔地に有る machineB の２点間でポートフォワーディングするには以下のコマンドを利用する。

machineA> ssh -L portnumA:hostnameB:portnumB username@machineB

上記のコマンドにおいて、portnumA は machineA のポート番号、hostnameB は machineB のホスト名、portnumB は machineB のポート番号である。hostnameB は machineB のターミナル上に表示されている場合が多く、また hostname コマンドでも確認できる。また、machineB のログインノードでポートフォワードが許可されている場合、hostnameB は特別なホスト名を入力する必要はなく、localhost が利用できる。

４.３.２. 多段リモート接続

遠隔地の２台のマシンでクライアントプログラム(machineA)とデーモンプログラム(machineB)を起動するが，セキュリティ上の理由等により machineC を経由して接続する例を示す。この場合も 2 点間の場合と同様に ssh ポートフォワードが確立した後の起動方法はスタンドアロンと同じである。

手順 1[ssh ポートフォワード A→C]

machineA> ssh -L 60000:localhost:60000 username@machineC

（machineA の 60000 番ポートを machineC の 60000 番ポートにフォワード）

手順 2[ssh ポートフォワード C→B]

machineC> ssh -L 60000:localhost:60000 username@machineB

（machineC の 60000 番ポートを machineB の 60000 番ポートにフォワード）

４.４. デーモンの起動とポートフォワード接続

デーモンは対話的処理が可能なノードあるいは、対話ジョブにおいて起動され、PBVR クライアントとソケット通信する。ユーザは手元の PC のターミナルから対話ノードへ ssh ポートフォワーディングし、デーモンのロードモジュールが配置されたディレクトリに移動し、以下のようにデーモンを起動する。

```
$ ./pbvr_daemon
first reading time[ms]:0
Server initialize done
Server bind done
Server listen done
Waiting for connection ...
```

上記のようにクライアントとのソケット通信の接続待ちになったら，別の端末から PBVR クライアントを起動する。デーモン起動時のポート番号の省略値は 60000 である。ポート番号は，以下のように起動時のコマンドラインオプション–p で変更できる。

```
$ ./pbvr_daemon -p 71000
```

デーモンは先述した環境変数（４.１節）を参照して粒子ファイルの集約や伝達関数ファイルの更新を行う。そして起動時に指定されたポートを通じて PBVR クライアントとデータの送受信を行う。

5 粒子サンプラ

シミュレーションコードに generate_particles 関数を挿入することで、粒子サンプラが In-Situ に可視化用粒子を生成できるようになる。この関数は kvs_wrapper.h で定義されており、粒子生成ライブラリを参照・リンクすることで使用可能となる。

シミュレーションに結合された粒子サンプラはシミュレーションのバッチ処理と共に実行され、先述した環境変数（4.1 節）と伝達関数ファイル（4.2節）を参照しながら計算結果のボリュームデータを可視化用粒子データに変換する。粒子サンプラは PARTICLE_DIR で指定されたディレクトリに粒子データファイルと、領域の最大最小値を記した t_pfi_coords_minmax.txt ファイルを出力する。粒子データファイルは、ヘッダファイル（kvsml）、座標ファイル（coord.dat）、色ファイル（color.dat）、法線ファイル（normal.dat）で構成されており、各タイムステップで１ノードにつき１組ずつ出力される。同時に、粒子サンプラは VIS_PARAM_DIR に伝達関数の変更履歴を記した$TF_NAME_タイムステップ.tf ファイルと、ヒストグラムや可視化対象の物理値のレンジを記した history_タイムステップ.txt ファイル、処理タイムステップ区間を記した state.txt を出力する。

5.1. 構造格子向けの粒子生成関数

```
#include “kvs_wrapper.h”
void generate_particles(
    int time_step, domain_patameters dom, Typs** volume_data, int num_volume_data );
```

この関数はシミュレーションのタイムステップや計算領域の情報、シミュレーション結果のボリュームデータを引数とする。

- int time_step：シミュレーションタイムステップ。
- domain_patameters dom：以下に示す計算領域を定義する構造体。

```
typedef struct
{
    float x_global_min;   //全計算領域のｘ座標の最小値
    float y_global_min;   //全計算領域の y 座標の最小値
    float z_global_min;   //全計算領域の z 座標の最小値
    float x_global_max;   //全計算領域のｘ座標の最大値
    float y_global_max;   //全計算領域の y 座標の最大値
    float z_global_max;   //全計算領域の z 座標の最大値
    float x_min;          //部分領域のｘ座標の最小値
    float y_min;          //部分領域の y 座標の最小値
    float z_min;          //部分領域の z 座標の最小値
    int*   resolution;    //格子解像度 int resolution[3]へのポインタ
    float cell_length;    //格子の単位長さ
} domain_parameters;
```

- Typs** volume_data：シミュレーション結果のボリュームデータの配列へのポインタ。Type はユーザ指定の物理値の型であり、多変数のボリュームデータは 2 次元配列として定義される。格子解像度(X,Y,Z)のボリュームデータにおいて、n 番目の変数の位置(i,j,k)の値は volume_data[n][i+j*X+k*X*Y]で参照する。
- int num_volume_data：変数の数

5.2. 非構造格子向けの粒子生成関数

```
#include “kvs_wrapper.h”
  void generate_particles(
        int time_step, domain_parameters dom, Type** values, int nvariables, float* coordinates,
        int ncoords, unsigned int* connections, int ncells,
        const pbvr::VolumeObjectBase::CellType& celltype );
```

この関数はシミュレーションのタイムステップや計算領域の情報、シミュレーション結果のボリュームデータ、非構造格子の格子情報を引数とする。

- int time_step：シミュレーションタイムステップ。
- domain_patameters dom：以下に示す計算領域を定義する構造体。

```
typedef struct
{
    float x_global_min;   //全計算領域のｘ座標の最小値
    float y_global_min;   //全計算領域の y 座標の最小値
    float z_global_min;   //全計算領域の z 座標の最小値
    float x_global_max;   //全計算領域のｘ座標の最大値
    float y_global_max;   //全計算領域の y 座標の最大値
    float z_global_max;   //全計算領域の z 座標の最大値
} domain_parameters;
```

- Typs** volume_data：シミュレーション結果のボリュームデータの配列へのポインタ。Type はユーザ指定の物理値の型であり、多変数のボリュームデータは 2 次元配列として定義される。n 番目の変数の cell 番目の頂点上の値は volume_data[n][cell]で参照する。
- float* coordinates：頂点座標の配列へのポインタ。i 番目の頂点座標(x,y,z)は(coordinates[3*i], coordinates[3*i+1], coordinates[3*i+2])で参照する。
- int ncoords：頂点の数。
- unsigned int* connections：六面体要素を構成する頂点 ID の接続リストへのポインタ。六面体要素の構成を図 5-1 に示す。i 番目の六面体要素の n 番目の頂点は connections[8*i+n]で参照する。
- int ncells：要素の数。
- pbvr::VolumeObjectBase::CellType& celltype：要素タイプ。IS-PBVR は四面体、六面体、ピラミッド、プリズム、四面体二次要素、六面体二次要素が使用可能である。各要素タイプは以下のファイルの VolumeObjectBase クラス内で定義されている。
  /DaemonAndSampler/InSituLib/unstruct/TFS/VolumeObjectBase.h

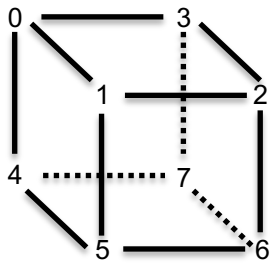


図 5-1　六面体要素の頂点接続

5.3. 階層格子向けの粒子生成関数

In-Situ PBVR ではメニーコア計算機のメモリレイアウトに最適化された階層格子構造（Block Structured AMR）をサポートしている。このタイプの格子は、N^3 の直交格子を最小の処理領域の単位（リーフ）と定義し、各階層でサイズの異なるリーフが接続されている。そのため Block Structured AMR は NxNxNxL(L はリーフ数)の四次元格子として定義される。図 5-2 に二次元の例を示す。

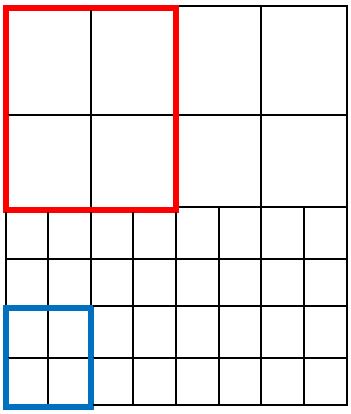


図 5-2 二次元の階層格子の例。上部が階層 Lv.1 で下部が階層 Lv.2.の格子。リーフが 2x2 の直交格子で定義され、赤が Lv.1 のリーフ、青が Lv.2 のリーフ。

```
#include “kvs_wrapper.h”
void generate_particles(
        int time_step, domain_patameters dom,
        std::vector<float>& leaf_length,
        std::vector<float>& leaf_min_coord,
        int nvariables, float** values);
```

この関数はシミュレーションのタイムステップや計算領域の情報、階層格子の構成情報、シミュレーション結果のボリュームデータを引数とする。

- int time_step：シミュレーションタイムステップ。
- domain_patameters dom：以下に示す計算領域を定義する構造体。

```
typedef struct
{
    float x_global_min;   //全計算領域のｘ座標の最小値
    float y_global_min;   //全計算領域の y 座標の最小値
    float z_global_min;   //全計算領域の z 座標の最小値
    float x_global_max;   //全計算領域のｘ座標の最大値
    float y_global_max;   //全計算領域の y 座標の最大値
    float z_global_max;   //全計算領域の z 座標の最大値
    int*   resolution;       //格子解像度 int resolution[4]へのポインタ
} domain_parameters;
```

・

・ std::vector<float>& leaf_length：リーフの長さの配列への参照。l 番目のリーフの長さは leaf_length[l]で参照する。

・ std::vector<float>& leaf_min_coord：リーフの最小位置座標の配列への参照。l 番目のリーフの座標は(leaf_min_coord[3*l], leaf_min_coord[3*l+1], leaf_min_coord[3*l+2])で参照する。

・ float** values：シミュレーション結果のボリュームデータの配列へのポインタ。多変数のボリュームデータは 2 次元配列として定義される。格子解像度(X,Y,Z,L)のボリュームデータにおいて、n 番目の変数の位置(i,j,k,l)の値は values[n][i+j*X+k*X*Y+l*X*Y*Z]で参照する。

・ int nvariables：変数の数

5.4. 粒子サンプラの組込み

ユーザのシミュレーションコードに generate_particles 関数を組み込みコンパイルすることで、In-Situ 可視化が可能になる。この章ではテストシミュレーションコードを例とした粒子サンプラの組み込み手順を示す。テストシミュレーションコードは、水素の電荷密度の式により各タイムステップで格子頂点上に電荷密度の値を計算する。そのため計算するクラス名は Hydrogen となっている。粒子サンプラの組み込みがない状態のコードは以下のようになっている。

```
#include "Hydrogen.h"
#include <iostream>
#include <mpi.h>
int main( int argc, char** argv )
{
    MPI_Init(&argc, &argv);
    Hydrogen hydro;
    int time_step = 0;
    for(;;)
    {
        hydro.values;
        time_step++;
    }
    MPI_Finalize();
    return 0 ;
}
```

上記ソースコードにおいて、Hydrogen クラスは for 文中の hydro.values で各タイムステップのボリュームデータを出力している。

5.4.1. 領域情報の定義と粒子生成関数の追加

generate_particles 関数は構造体 domain_parameters によって領域情報を取得する。ユーザはシミュレーションコードから得られる領域情報を構造体にコピーする必要がある。

```
int mpi_rank;
MPI_Comm_rank( MPI_COMM_WORLD, &(mpi_rank) );
int resol[3] = { hydro.resolution.x(), hydro.resolution.y(), hydro.resolution.z() };
domain_parameters dom = {
    hydro.global_min_coord.x(),
    hydro.global_min_coord.y(),
    hydro.global_min_coord.z(),
    hydro.global_max_coord.x(),
    hydro.global_max_coord.y(),
    hydro.global_max_coord.z(),
    hydro.global_region[mpi_rank].x(),
    hydro.global_region[mpi_rank].y(),
    0.0,
    resol,
    hydro.cell_length
};
```

上記コードでは、Hydrogen の部分領域の情報を得るために MPI のランク数を取得している。

領域情報とシミュレーション結果のボリュームデータを generate_particles 関数に入力する。generate_particles 関数はタイムステップループの内部でシミュレーション後にコールされる。Hydrogen の例の場合、以下のように挿入される。

```
int time_step = 0;
for(;;) {
    generate_particles( time_step, dom, hydro.values, hydro.nvariables )
    time_step++;
}
```

6 PBVR クライアント

PBVR クライアントはデフォルトでデーモンとから受信した最新タイムステップの粒子データを描画する。粒子データはビューワ上に OpenGL により描画される。PBVR クライアントは伝達関数を含む可視化パラメータを編集する GUI を提供し、可視化パラメータをデーモンに送信する。デーモンと PBVR クライアント間のデータの送受信はソケット通信により任意のポート番号を通して行う。

6.1. 起動方法

PBVR クライアントは以下のコマンドで実行する。

$ pbvr_client [コマンドラインオプション]

表 6-1　クライアントのコマンドラインオプション一覧

オプション	指定値	デフォルト	機能
-p	ポート番号	60000	ソケット通信ポート番号
-viewer	100 ～ 9999 × 100～9999	620×620	PBVR クライアント解像度
-shading	{L/P/B},ka,kd,ks,n	-	シェーディング方法 ※1

※1. シェーディング方法の指定は以下の通り。

L:ランバートシェーディング
効果：拡散反射を考慮したシェーディングを与える。
使用パラメータ：ka（物体の明るさに掛かる係数。0～1 の実数），kd（法線方向と光線方向から計算される拡散反射成分に掛かる係数。0～1 の実数）

P：フォンシェーディング
効果：ランバートシェーディングにハイライト効果を追加したもの。
使用パラメータ：ka,kd,ks（視線方向，法線方向，光線方向から計算される鏡面反射成分に掛かる係数。0～1 の実数），n（ハイライトの強さ。0～１００の実数）

B：ブリン-フォンシェーディング
効果：フォンシェーディングの簡略化モデル
使用パラメータ：ka,kd,ks,n

6.2. 終了方法

PBVR クライアントの終了はプログラムを起動したコンソールにおいて Ctrl+c キーを押して行う。Ctrl+c キーを押すと，時刻更新のタイミングで PBVR クライアントはデーモンと同期し、両方が終了する。ただし，タイムステップ制御パネルの Stop ボタンで通信を中断させた状態だと Ctrl+c キーの入

力は無視される。また、デーモンを Ctrl+c キーで終了させてしまうと，PBVR クライアントを Ctrl+c キーで終了させることができなくなるため注意すること。

6.2.1. 強制終了

何らかの理由により Ctrl+c キーの押下で PBVR クライアントとデーモンが終了しない場合は、以下に示すように，ps コマンドでクライアントとデーモンのプロセス番号を取得し，kill コマンドで強制終了させる。

【クライアントプロセスの強制終了】

```
$ ps -C PBVRViewer
  PID TTY          TIME CMD
19582 pts/6    00:00:00 PBVRViewer
$ kill -9 19582
```

【デーモンプロセスの強制終了】

```
$ ps -C CPUServer
  PID TTY          TIME CMD
19539 pts/5    00:00:00 CPUServer
$ kill -9 19539
```

6.3. GUI の構成と操作方法

PBVR クライアントは可視化結果を表示し、可視化パラメータを対話的に制御できる GUI を提供する。

6.3.1. ビューワ

図 6-1 に示すビューワには、粒子データの描画結果が表示される。

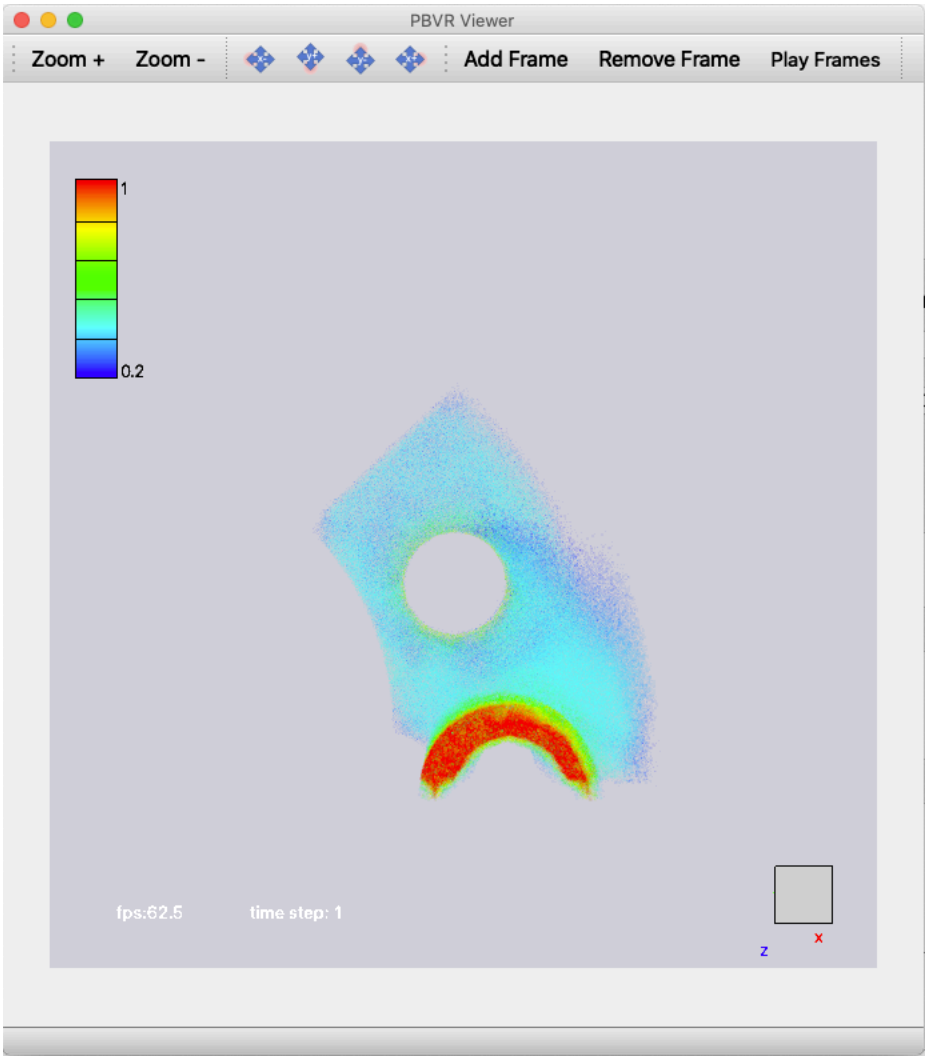


図 6-1　ビューワ

【操作方法】

回転：マウスで左ドラッグ

移動：マウスで右ドラッグ

拡大・縮小：マウスで Shift+左ドラッグ　もしくは　マウスホイール押下+ドラッグ

リセット：home ボタン（Mac では fn + left arrow）

【表示内容】

time step：表示しているデータのタイムステップ

fps：フレームレート（frame/sec）

【ツールバー】

拡大、縮小

平行移動

キーフレームの取得/削除/アニメーションの再生

6.3.2. ツールバー

クライアントプログラムの様々な機能はツールバーから選択される。“File”タブは可視化パラメータファイルや伝達関数の入出力を制御する。

可視化パラメータとは、ビューワの解像度、粒子密度、粒子数制限値、サーバ上の入力ボリュームデータファイル名（pfi ファイルまたは pfl ファイル）、伝達関数等、クライアントで設定可能なパラメータ郡のことを意味し、可視化パラメータファイルとはそれらがタグ形式で記述されたファイルのことである。

以下に“File”タブとその機能を示す。

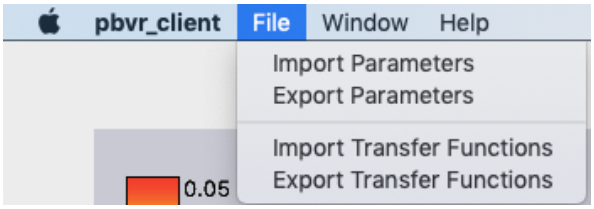


図 6-2 “File”タブ

- Import Parameters
  入力する可視化パラメータファイルを指定する。
- Export Parameters
  可視化パラメータファイルを出力する。
- Import Transfer Functions
  入力する伝達関数ファイルを指定する。
- Export Transfer Functions
  伝達関数ファイルを出力する。

以下に“Window”タブとその機能を示す。

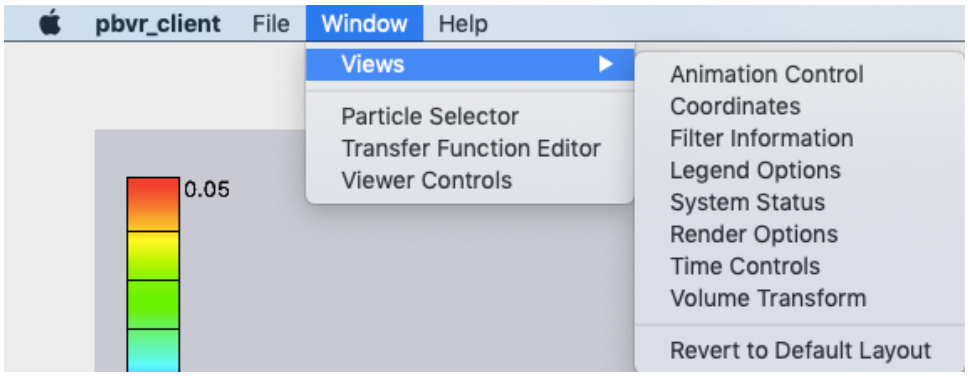


図 6-3 “Window”タブ

- Animation Control
  動画作成用パネルを表示する。詳細については後述。

• Coordinates
  座標エディタを表示する。詳細については後述。
• Filter Information
  入力ボリュームデータの情報を表示する。詳細については後述。
• Legend Options
  レジェンドパネルを表示する。詳細については後述。
• System Status
  システム情報を表示する。詳細については後述。
• Render Options
  レンダリングのオプションを制御する。詳細については後述。
• Time Controls
  タイムステップを制御する。詳細については後述。
• Volume Transform
  描画対象の幾何変換を指定する。詳細については後述。
• Revert to Default Layout
  GUI のレイアウトをクライアントプログラム起動時に戻す。
• Transfer Function Editor
  伝達関数エディタを表示する。詳細については後述。
• Viewer Controls
  ビューワ制御パネルを表示する。詳細については後述。

6.3.2.1 **FilterInfo**

FilterInfo パネルでは入力されたフィルタ処理済みのボリュームデータの情報を表示する。各項目について以下に説明する。

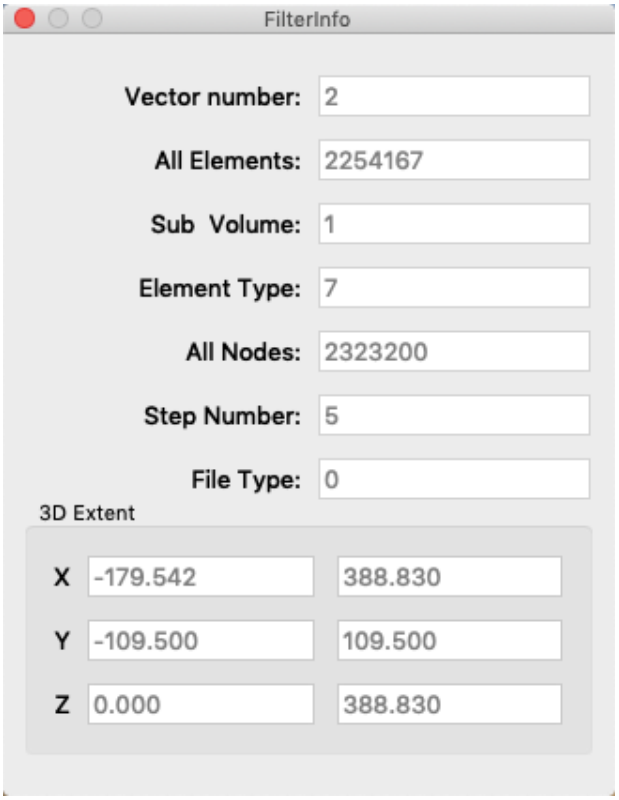


図 6-4 FilterInfo

- Vector number
  ボリュームデータに含まれる変数の数
- All Elements
  ボリュームデータの要素数
- Sub Volume
  ボリュームデータの分割数
- Element Type
  ボリュームデータを構成する要素タイプ。
- All Nodes
  ボリュームデータの頂点数
- Step Number
  ボリュームデータのタイムステップ数
- FileType
  クライアント・サーバ型 PBVR におけるフィルタの分割形式。In-Situ PBVR では使用しない。
- 3D Extent
  ボリュームデータの X-Y-Z 座標の最大最小値

6.3.2.2 **System Status**

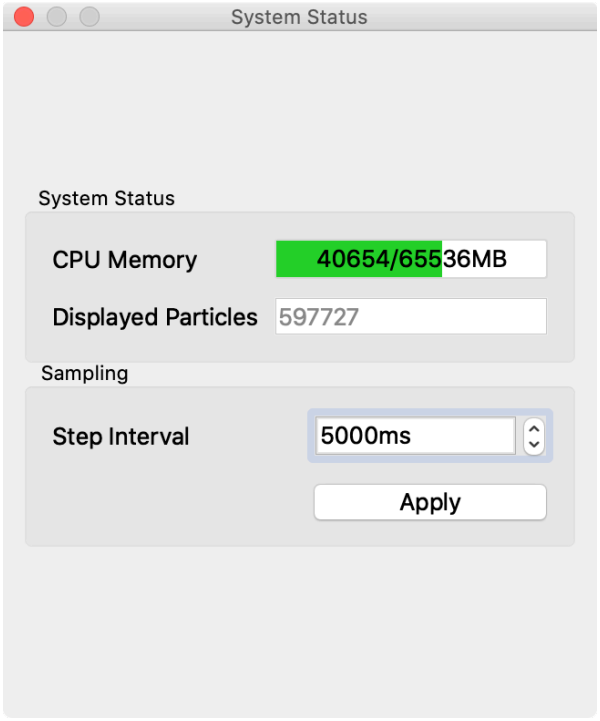


図 6-5　システム情報

- CPU Memory
  ステムメモリの使用状況（単位：MB）を表示する。
- Displayed Particles
  ビューワに表示されている粒子の数を表示する。
- Step Interval
  タイムステップの更新間隔の最低値を指定する（単位 msec）。タイムステップの更新間隔が短すぎる場合の調整に利用する。

6.3.2.3 **Render Options**

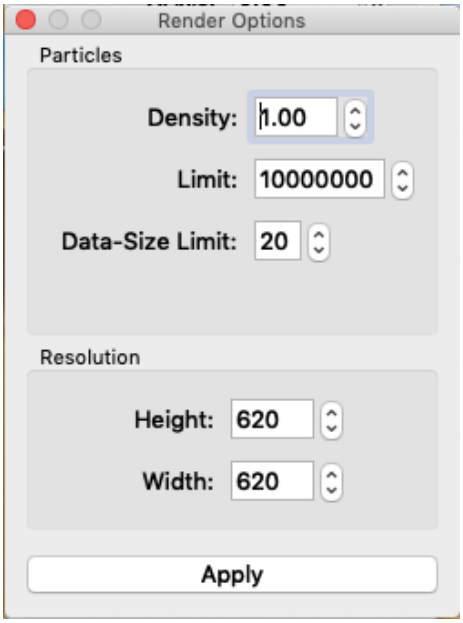


図 6-6　レンダリング用のオプション

- Density
  画像の濃さに影響する粒子密度を指定する。
- Limit
  伝達関数誤指定等による粒子数の爆発を避けるため、サーバプログラム側で生成する粒子数の上限値を指定する。粒子数がこの上限を超える場合、サーバプログラムは自動的に画質を下げて粒子数がこの上限に収まるように調整する。
- Data-Size Limit
  伝達関数誤指定等による粒子数の爆発を避けるため、サーバプログラム側で生成する粒子データサイズの上限値を指定する。単位は[GB]。粒子データサイズがこの上限を超える場合、粒子生成は強制的に停止する。
- Resolution
  レンダリング解像度を指定する。

6.3.2.4 **Volume Transform**

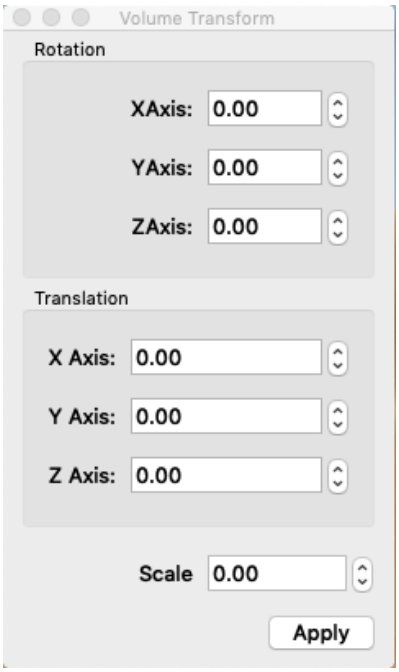


図 6-7　描画対象の幾何変換

・Rotation

物体のｘ軸、ｙ軸、ｚ軸それぞれに関する回転角（度数）を指定する。

・Translation

物体のｘ、ｙ、ｚ方向の平行移動を指定する。

・Scale

物体の拡大率を指定する。

6.3.3. 伝達関数エディタ

伝達関数エディタでは、物理値に割り当てられる色および不透明度を定義する伝達関数を作成できる。通常のボリュームレンダリングでは伝達関数は一つの物理量のみによって定義されるが、PBVR では

（1） 色と不透明度に独立な変量を割り当てる。
（2） 各変量を座標 X、Y、Z、変量 q1、q2、q3…の任意の関数式で定義。
（3） １次元伝達関数の色関数を C1、C2…、不透明度関数を O1、O2、….で定義し、さらにそれら任意の関数式で合成して多次元伝達関数を定義。

という、新たな伝達関数設計を可能としたことで、極めて自由度の高い可視化処理を実現した。図 6-8 に伝達関数エディタパネルを示す。パネル内の各項目について以下に説明する。

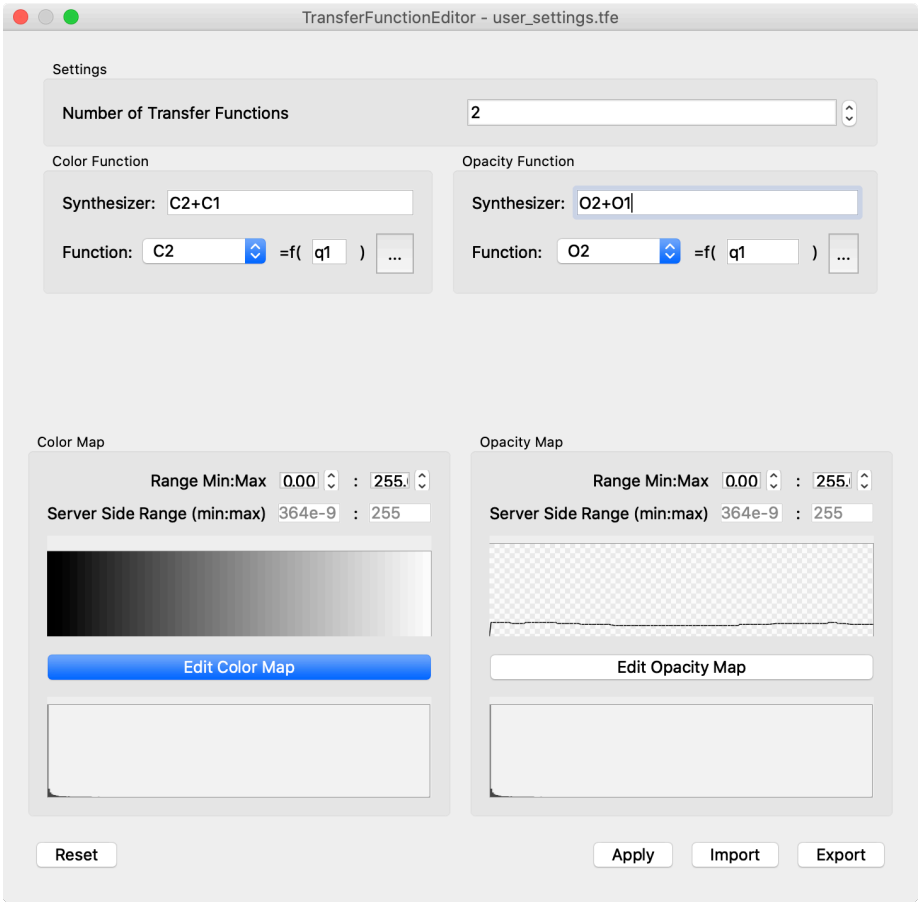


図 6-8 伝達関数エディタパネル

【操作方法】

- histogram スケール変更
  Histogram 上でマウスを上下にドラッグ
- Number of Transfer Fuctions
  作成可能な伝達関数の制限数を指定する
- Color Fuction カテゴリ

色関数とその引数となる変量の合成式を指定する。

- Opacity Fuction カテゴリ

  不透明度関数とその引数となる変量の合成式を指定する。

- Color Map カテゴリ

  色関数を編集する。

- Opacity Map カテゴリ

  不透明度関数を編集する。

- Reset ボタン

  本パネルを初期状態にリセットする。

- Apply ボタン

  本パネルで作成した伝達関数をサーバへ送信する。編集された伝達関数はタイムステップ更新のタイミングでサーバに読み込まれるため、可視化結果に反映されるまで１ステップ分のタイムラグがある。

- Export ボタン

  本パネルで作成した伝達関数を、-pa オプションで指定するパラメータファイルと同じ書式で、ファイルへ保存する。

- Import ボタン

  ファイルに保存されている伝達関数を読み込んで本パネルへ反映する。

【代数式で使用可能な演算子】

二項演算子　+, -, *, /, ^

関数　sqrt(),cbrt(), log(), log10(), exp(), abs(), floor(), ceil(), sin(), cos(), tan, asin(), sinh(), cosh(), tanh(), asinh(), acosh(), atanh(), heavi(), rectfunc(),

二項関数　sigmoid(*,*), gauss(*,*)

6.3.3.1 カラーマップ指定機能

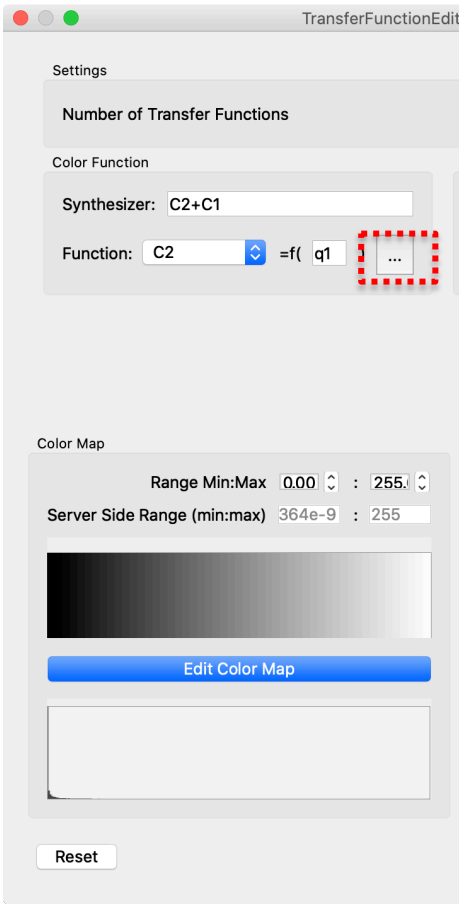


図 6-9　上部：Color Function カテゴリ、下部：Color Map カテゴリ

【Color Function カテゴリ】

- Synthesizer

  色関数 C1～C[N] ※1 による合成式を指定する。

- Function スピンボタン

  編集する色関数 C1～C[N]を選択する。

- Color Function Editor ボタン

  上図 6-9 中の赤い点線で囲まれたボタン。Color Function Editor 画面を表示して、選択された色関数 C1～C[N]の引数となる（合成）変量を作成する。

【Color Map カテゴリ】

- User Defined Range Min:Max

  Color Function Editor で指定した（合成）変量に対して、色関数を割り当てる最大最小値を指定する。

- Synth. Func. Range Min:Max

  Color Function Editor で指定した（合成）変量の最大最小値を表示する。

- Edit Color Map ボタン

選択した色関数名に対応するカラーマップを作成するための Color Map Editor パネルを開く。

・Histogram

User Defined Range Min:Max で指定した最大最小値の範囲のヒストグラムが表示される。

※1：[N]は Number Transfer Fuctions にて指定した伝達関数の制限数の値である。

6.3.3.1.1. Color Function Editor

Color Function Editor では、ユーザーが指定した代数式を使用して物理値を合成できる。合成した物理値は色関数 C1～C[N]の引数となる。代数式で使用できる変量の名称は以下に示される。

- 物理量：q1、 q2、 q3、・・・qn
- 座標値：X、 Y、 Z

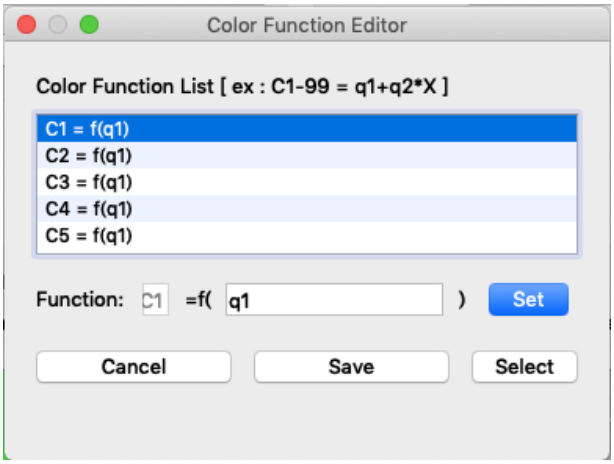


図 6-10 Color Function Editor 画面

- Color Function List
  作成されている伝達関数を表示する。
- Function
  伝達関数 C[N] = f(変数) を入力する
- Set ボタン
  Color Function List に反映する
- Cancel ボタン
  本パネルを閉じる
- Save ボタン
  Color Function List の伝達関数を適用する。
- Select ボタン
  Color Function List の伝達関数の適用、リスト選択の伝達関数を選択する。

6.3.3.1.2. Color Map Editor：Freeform Curve Edit タブ

Color Map Editor の Freeform Curve Edit タブでは、C1～C[N]に対応する色関数をマウスによる自由曲線入力で作成できる。

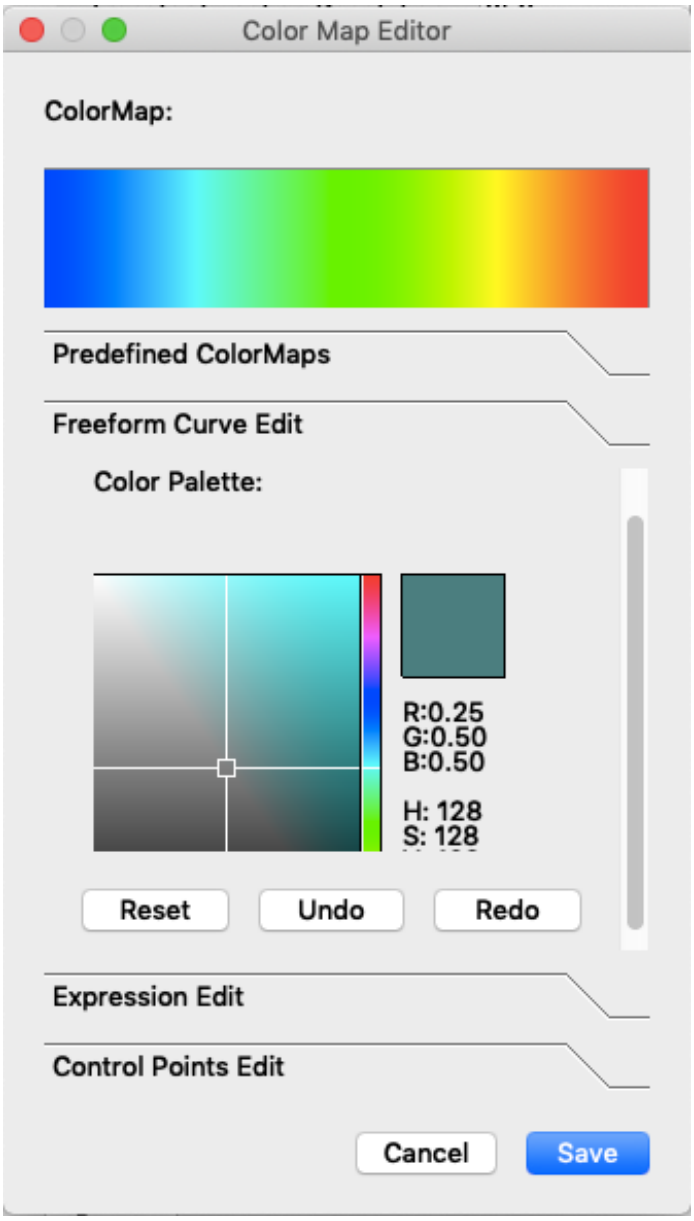


図 6-11　Color Map Editor の Freeform Curve Edit タブ

- Color palette
  横軸が彩度、縦軸が明度で、マウスの位置によりこれらを指定する。
- RGB 指定バー
  色相をマウスの位置により指定する。Color palette には選択した色相が反映される。
- Reset ボタン
  本パネルを初期状態に戻す。

• Undo ボタン

　　１マウスアクションを取り消す。

• Redo ボタン

　　取り消したマウスアクションを再実行する。

• Save ボタン

　　本パネルで作成した伝達関数を保存する。

• Cancel ボタン

　　本パネルを閉じる。

ColorMap を左クリックでなぞることによって、Color palette と RGB 指定バーで作成した色でバーを上塗りする。ここで、既存の色との合成比率はバーの縦軸方向のマウス位置によって決定される。例えば、カラーバー上辺を左右になぞれば、なぞった範囲を指定色のみで塗りつぶし、カラーバー上下中央位置を左右になぞれば、なぞった範囲を既存色と 50%の割合で合成する。

6.3.3.1.3. Color Map Editor：Expression Edit タブ

Color Map Editor の Expression Edit タブでは、C1～C[N]に対応する色関数を数式記述で作成できる。

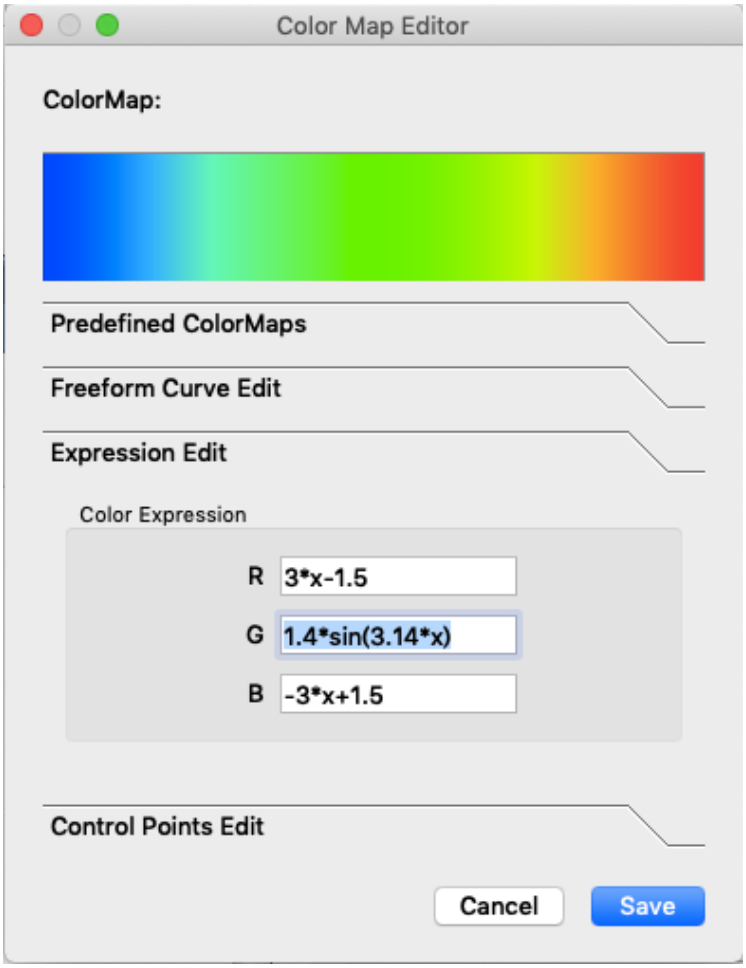


図 6-12　Color Map Editor の Expression Edit タブ

- R

  色の R 成分の色関数式を代数式で記述する。

- G

  色の G 成分の色関数式を代数式で記述する。

- B

  色の B 成分の色関数式を代数式で記述する。

色関数の変数は x であり、色関数の定義域と値域は 0 から 1 である。

6.3.3.1.4. Color Map Editor：Contlol Points Edit タブ

Color Map Editor の Control Points Edit タブでは、C1～C[N]に対応する色関数を制御点指定で作成できる。

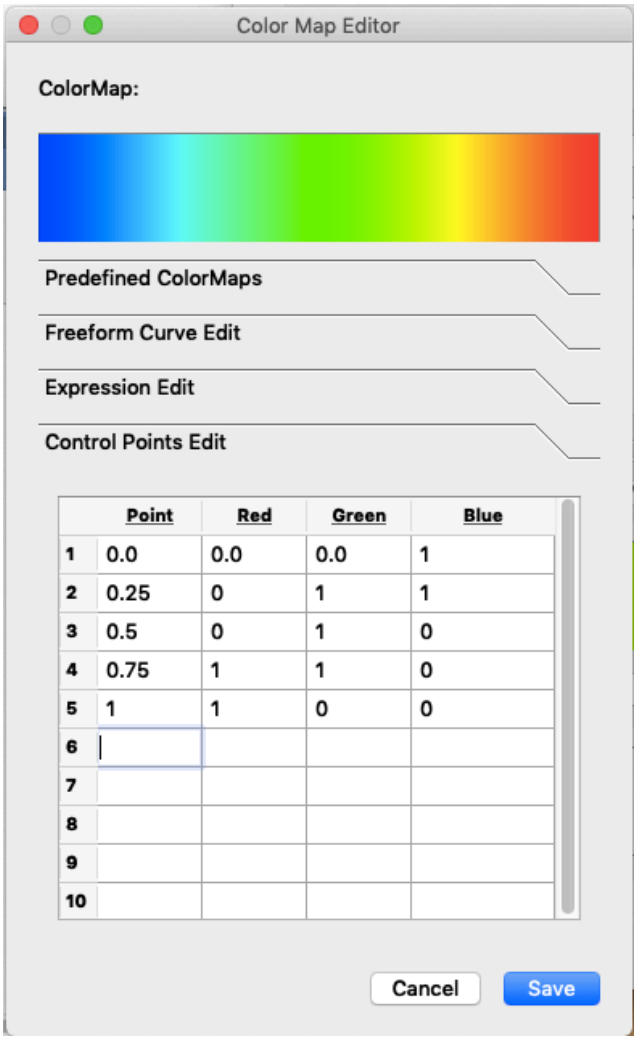


図 6-13　Color Map Editor の Control Points Edit タブ

- Point

  制御点（最大 10 個）の値を指定する。定義域は0から１の範囲である。

- Red

  制御点の値に対応する色のＲ成分値を指定する。値域は0から１の範囲である。

- Green

  制御点の値に対応する色のＧ成分値を指定する。値域は0から１の範囲である。

- Blue

  制御点の値に対応する色のＢ成分値を指定する。値域は0から１の範囲である。

各制御点は区分線形関数で補間される。

6.3.3.1.5. Color Map Editor：Predefined ColorMaps タブ

Color Map Editor の Predefined ColorMaps タブでは、C1～C[N]に対応する色関数をあらかじめ用意されている色関数から選択できる。

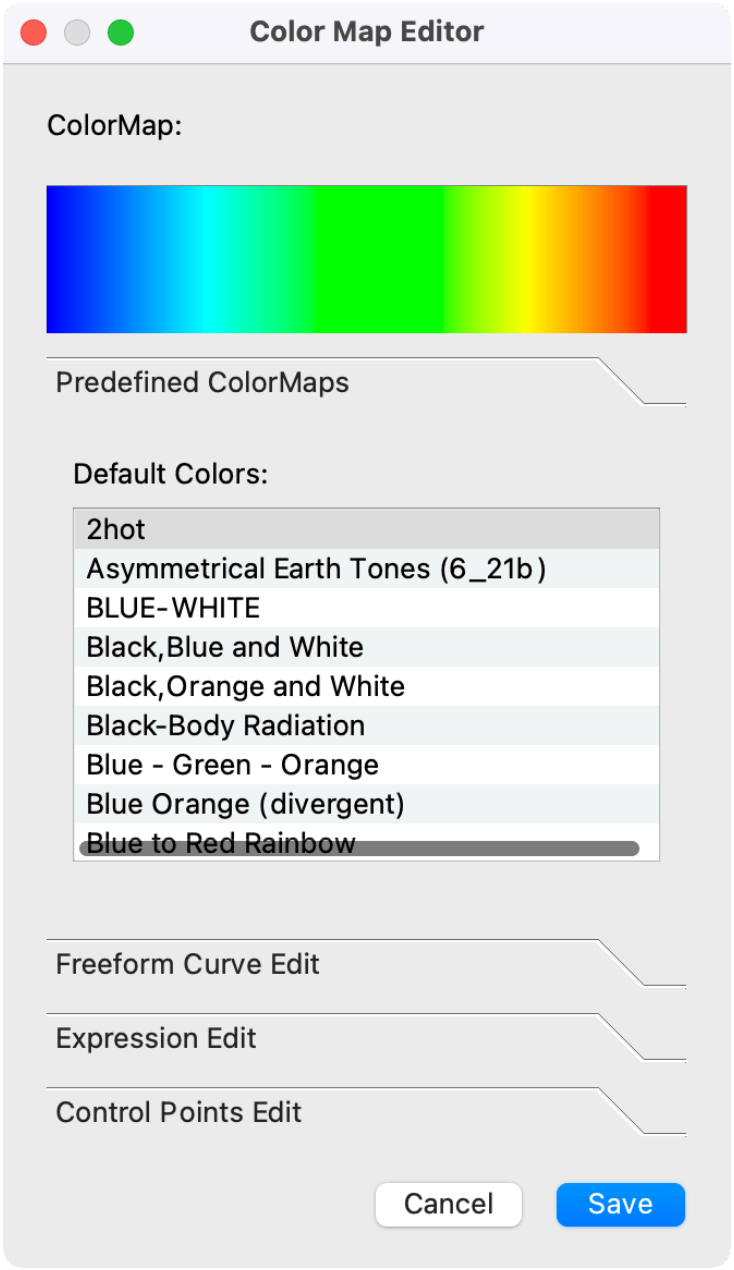


図 6-14　Color Map Editor の Predefined ColorMaps タブ

- Color

  本パネルで作成した伝達関数のカラーバーが表示される。

- Default Color プルダウンメニュー

  伝達関数として設定するカラーバーを選択する。選択肢は以下。

  - RainBow
  - Blue-white-red

- Black-red-yellow-white
- Black-blue-violet--yellow-white
- Black- yellow-white
- Blue-green-red
- Green-red-violet
- Green- blue--white
- HSV model
- Gray-scale
- Black
- White

6.3.3.2 不透明度指定機能

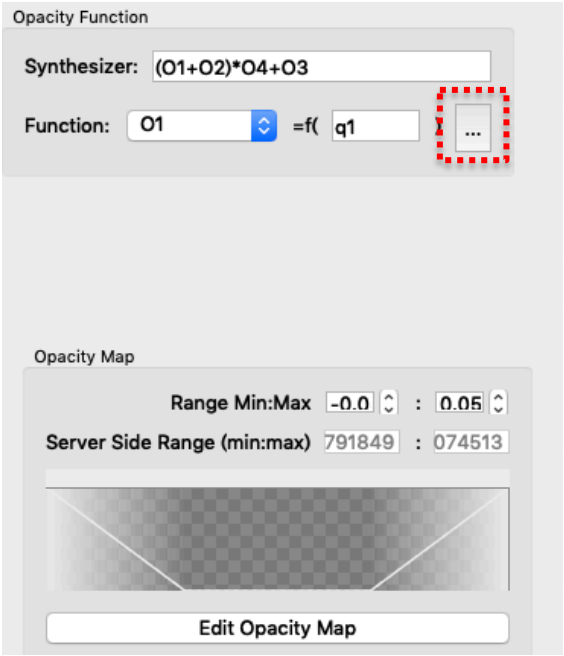


図 6-15　上部：Opacity Function カテゴリ、下部：Opacity Map カテゴリ

【Opacity Function カテゴリ】

- Synthesizer

  不透明度関数 O1～O[N] ※1 による合成式を指定する。

- Function スピンボタン

  編集する不透明度関数 O1～O[N]を選択する。

- Color Function Editor ボタン

  上図 6-15 中の赤い点線で囲まれたボタン。Opacity Function Editor 画面を表示して、選択された不透明度関数 O1～O[N]の引数となる（合成）変量を作成する。

【Opacity Map カテゴリ】

- Range Min:Max

  選択した不透明度関数名に対する（合成）変量の最小値：最大値を指定する。

- Server side range (min:max)

  Opacity Function Editor で指定した（合成）変量について、サーバ側で取得した最小値：最大値を表示する。

- Edit Opacity Map ボタン

  選択した不透明度関数名に対応する不透明度関数を作成するための Opacity Map Editor パネルを開く。

※1：[N]は Number Transfer Fuctions にて指定した伝達関数の制限数の値である。

6.3.3.2.1. Opacity Function Editor

Opacity Function Editor 上では、不透明度関数 O1～O[N]の引数となる（合成）変量の定義式が指定される。定義式で使用できる変量の名称は以下に示される。

・物理量：q1、 q2、 q3、・・・qn

・座標値：X、 Y、 Z

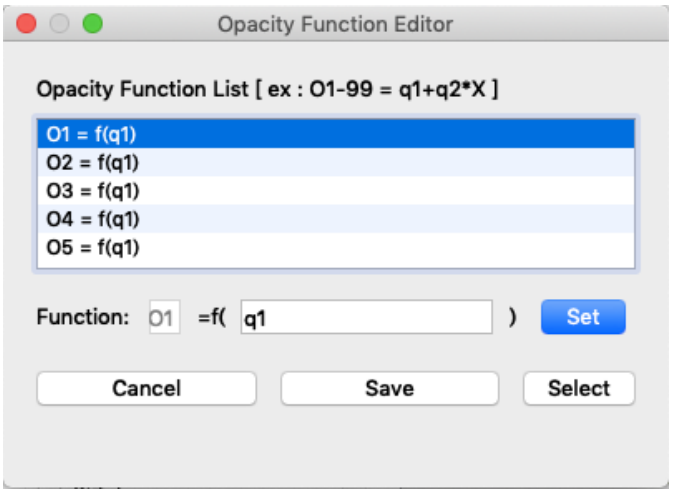


図 6-16 Opacity Function Editor 画面

・Opacity Function List

作成されている伝達関数を表示する。

・Function

伝達関数 O[N] = f(変数)を入力する

・Set ボタン

Opacity Function List に反映する

・Cancel ボタン

本パネルを閉じる

・Save ボタン

Opacity Function List の伝達関数を適用する。

・Select ボタン

Opacity Function List の伝達関数の適用、リスト選択の伝達関数を選択する。

6.3.3.2.2. OpacityMap Editor：Freeform Curve Editor タブ

Opacity Map Editor の Freeform Curve Edit タブは、O1～O[N]に対応する不透明度関数をマウスによる自由曲線入力で作成する機能を提供する。

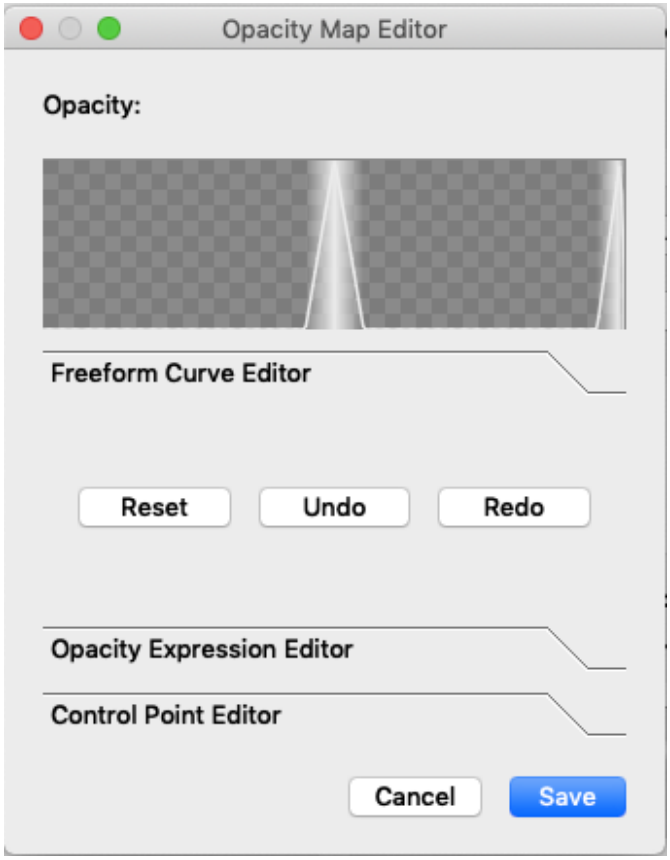


図 6-17　Opacity Map Editor の Freeform Curve Editor タブ

- Reset ボタン
  本パネルを初期状態に戻す。
- Undo ボタン
  １マウスアクションを取り消す。
- Redo ボタン
  取り消したマウスアクションを再実行する。
- Save ボタン
  本パネルで作成した伝達関数を保持する。
- Cancel ボタン
  本パネルを閉じる。

マウス操作により不透明度の伝達関数が作成される。左ドラッグで自由曲線が、右クリックで 2 点間の線形補間直線が描画される。

6.3.3.2.3. Opacity Map Editor：Opacity Expression Editor タブ

Opacity Map Editor の Opacity Expression Editor タブでは、O1～O[N]に対応する不透明度関数を数式記述で作成するパネルを開くことができる。

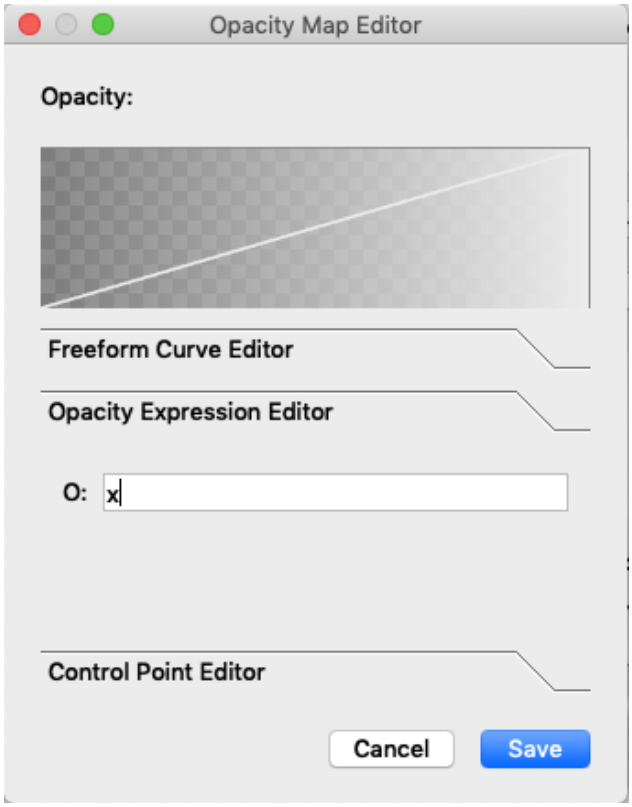


図 6-18　Opacity Map Editor の Opacity Expression Editor タブ

・O

不透明度関数をユーザ指定の代数式で記述できる。不透明度関数の変数は x であり、不透明度関数の定義域および値域は0から１である。

6.3.3.2.4. Opacity Map Editor：Control Point Editor タブ

Opacity Map Editor の Control Points Editor タブでは、O1～O[N]に対応する不透明度関数を制御点指定で作成できる。

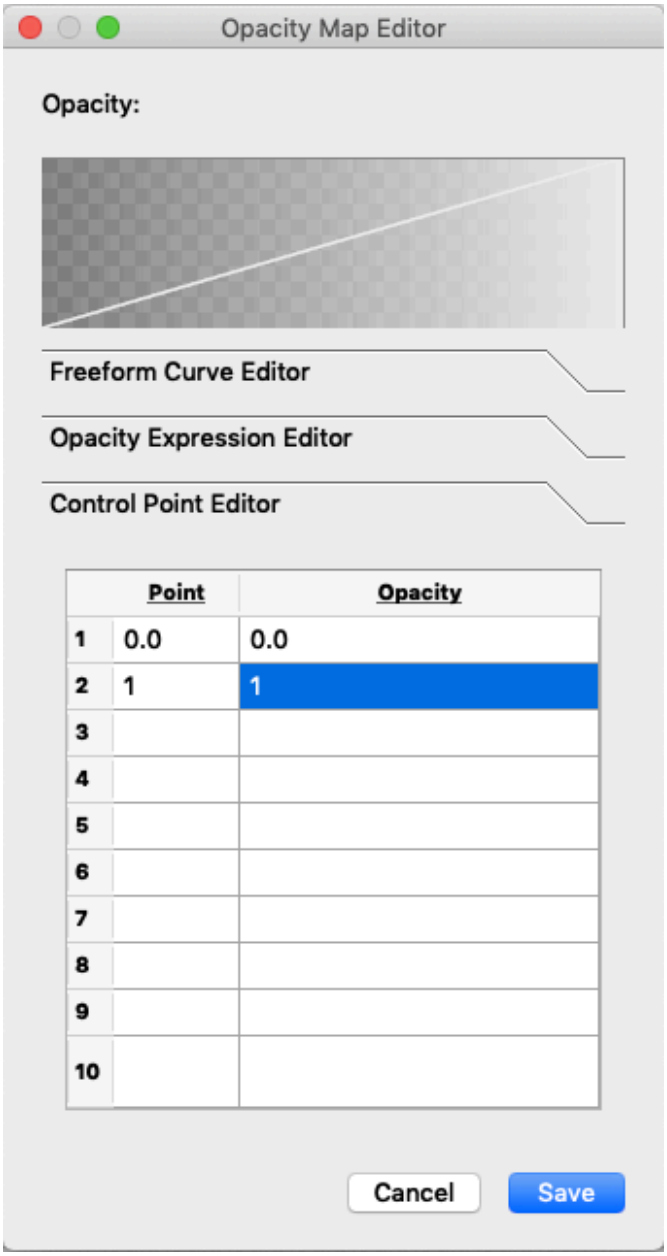


図 6-19　Opacity Map Editor の Control Point Editor タブ

- Control Point

  制御点（最大 10 個）の値を CP1)～CP10)に指定する。値の範囲は0から１の実数である。

- Opacity（中段右側）

  制御点の値に対応する不透明度を指定する。値の範囲は0から１の実数である。

6.3.3.3 関数エディタ

伝達関数エディタにおける伝達関数合成、変量合成、カラーマップ曲線、不透明度曲線の入力に使用される関数エディタで使用できる組み込み関数は以下の通り。

表 6.3-1　関数エディタで利用可能な演算

演算	書式
+	+
-	-
×	*
/	/
Sin	sin(x)
Cos	cos(x)
Tan	tan(x)
Log	log(x)
Exp	exp(x)
平方根	sqrt(x)
冪乗	x^y

関数エディタの演算処理で NaN が現れた場合には PBVR はエラーメッセージを出力して描画処理を停止する。

6.3.4. タイムステップ制御パネル

タイムステップ制御パネルを図 6-20 に示す。各ウィジットの動作は以下のとおり。

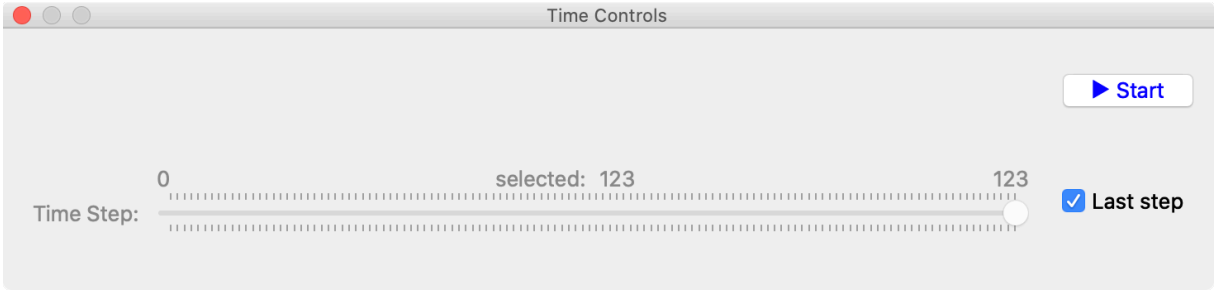


図 6-20 タイムステップ制御パネル

- Time step

  レンダリングするタイムステップを指定する。

- Last step

  常に最新のタイムステップを表示する。

- Start/Stop

  サーバ/クライアント間の通信を開始または停止する。

6.3.5. 粒子・ポリゴンの統合表示

CS-PBVR は、ボリュームレンダリングとポリゴンの合成表示をサポートしている。複数の粒子データおよびポリゴンを統合して表示する粒子統合エディタを図 6-21 に示す。粒子統合エディタはメインパネルの Particle Panel ボタンを押下すると開く。粒子統合エディタの操作方法は以下の通り。

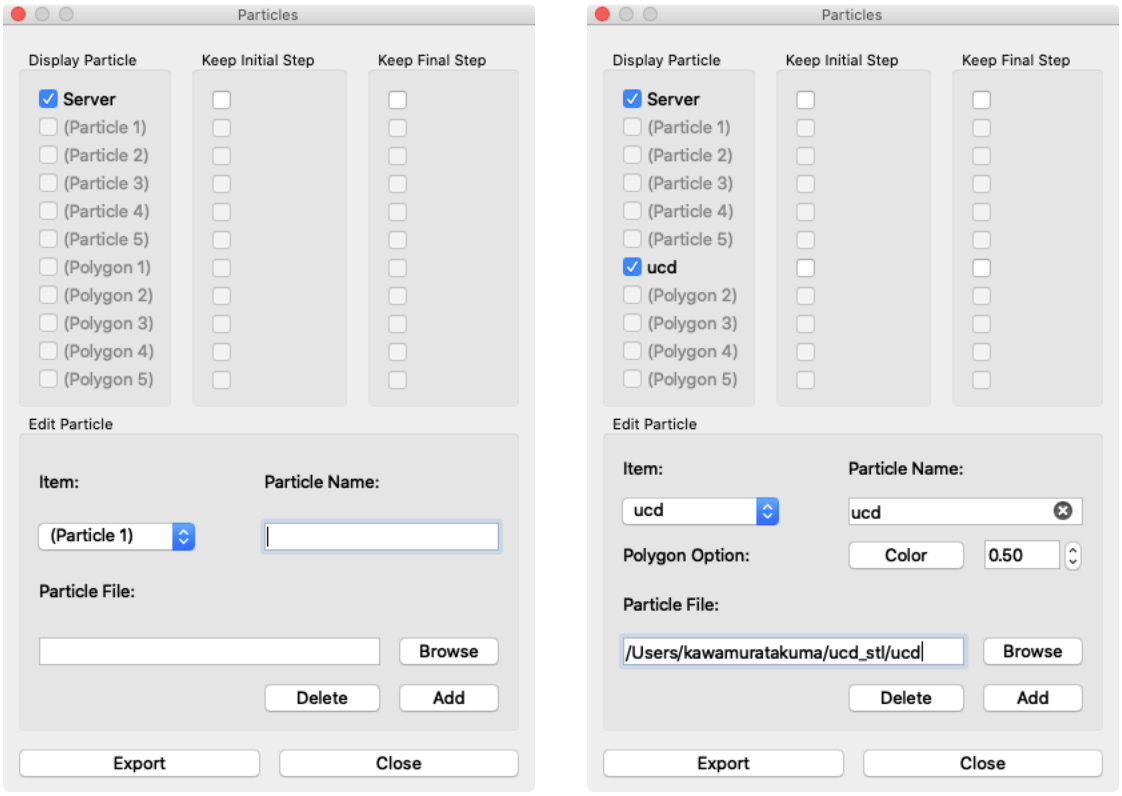


図 6-21　粒子統合エディタ。左と右はそれぞれ、起動時の状態とポリゴンを選択した状態のエディタである。

Display Particle カテゴリは統合表示の対象となる粒子データおよびポリゴンの一覧を示す。粒子データはサーバから送られてきたデータ、もしくは、クライアントのローカルファイルから読み込んだデータ（最大 5 個まで）から構成され、ポリゴンはクライアントのローカルファイルから読み込んだデータ（最大 5 個まで）から構成される。

粒子統合エディタは STL 形式のポリゴンをサポートしている。本機能は以下の命名規則に従う 1 タイムステップにつき 1 ファイルに分割された STL ファイルを処理可能である。

prefix_*****.stl　（*****は 5 桁表示のタイムステップ数）

サーバから送信されてくる粒子データと、ローカルの粒子データのサブピクセルレベルが異なる場合、サーバのサブピクセルレベルが優先される。スタンドアロンモードにおいて、複数のローカルの粒子データを表示する場合で、かつ各粒子データのサブピクセルレベルが異なる場合、後に読み込んだ粒子データのサブピクセルレベルが優先される。

Keep Initial Step カテゴリは開始タイムステップが異なる粒子データ/ポリゴンを統合表示するときに、時系列の開始前に先頭のタイムステップのデータを表示する粒子データ/ポリゴンの一覧を示す。

Keep Final Step カテゴリは終了タイムステップが異なる粒子データ/ポリゴンを統合表示するときに、時系列の終了後に最終のタイムステップのデータを表示する粒子データ/ポリゴンの一覧を示す。

【Display Particle/ Keep Initial Step/ Keep Final Step カテゴリ】

- Server チェックボックス
  クラサバモードで動作中にサーバから送られてきた粒子データを統合表示の対象とする場合にチェックする。スタンドアロンモードで動作中のときはチェックできない。
- (Particle1)～(Particle５)チェックボックス
  クライアントのローカルファイルから読み込んだ粒子データを統合表示の対象とする場合にチェックする。粒子データが読み込まれていない状態ではチェックできない。後述する Particle file パネルで粒子データを読み込む。
- (Polygon1)～(Polygon5)チェックボックス
  クライアントのローカルファイルから読み込んだポリゴンを統合表示の対象とする場合にチェックする。ポリゴンが読み込まれていない状態ではチェックできない。後述する Particle file パネルでポリゴンを読み込む。

【Edit Particle カテゴリ】

Edit Particle category では、クライアント PC のローカルファイルから粒子統合エディタの一覧へ粒子データやポリゴンを追加したり、統合して表示した粒子データを保存したりできる。

- Item スピンボタン
  ⑴Particle1⑴～⑴Particl5⑴，⑴Polygon1⑴～⑴Polygon5⑴の一覧から、粒子データを追加する項目を選択する。
- Particle Name
  Item スピンボタンで選択した粒子データまたはポリゴンに名前をつける。この記述を省略すると、Particle File 欄に表示されているファイル名が採用される。
- Browse ボタン
  粒子データまたはポリゴンを読み込むためのファイルダイアログを開く。読み込んだ粒子データまたはポリゴンのファイルのパスは Particle File 欄に表示される。全角文字列を含むパスを指定することはできない。
- Add ボタン
  Particle File 欄に表示されている粒子データを、Item スピンボタンで選択した項目へ追加する。既に追加済みの項目を選択した場合は、後から読み込んだ粒子データまたはポリゴンで上書きする。
- Delete ボタン
  Item スピンボタンで選択中の項目に追加されている粒子データまたはポリゴンを削除する。
- Export ボタン
  メモリ上にある粒子データを統合して Particle File 欄で指定したファイルへ出力する。
- Close ボタン
  粒子統合エディタをクローズする。

• Polygon Option

Item スピンボタンで Polygon を選択した場合に出現する。ポリゴンの色と不透明度を指定できる

6.3.5.1 ボリューム・ポリゴンの合成例

Example ディレクトリに含まれるテストプログラム Hydrogen と、その境界形状のポリゴン hydrogen.stl を用いた合成表示の例を示す。サーバとクライアントをポートフォワード接続し、テストプログラム Hydrogen とデーモンを起動し、クライアントでボリュームレンダリングを実行する（図 6-22 左）。次に粒子統合パネルの Edit Particle カテゴリで Item スピンボックスから Polygon を選択し、Particle File で hydrogen.stl のパスを指定する。次に Particle Name でデータ名“hydrogen.stl”を指定し、色と不透明度を設定する。そして Display Particle カテゴリで hydrogen.stl をチェックする（図 6-22 右）と、境界形状が合成される（図 6-22 中央）。

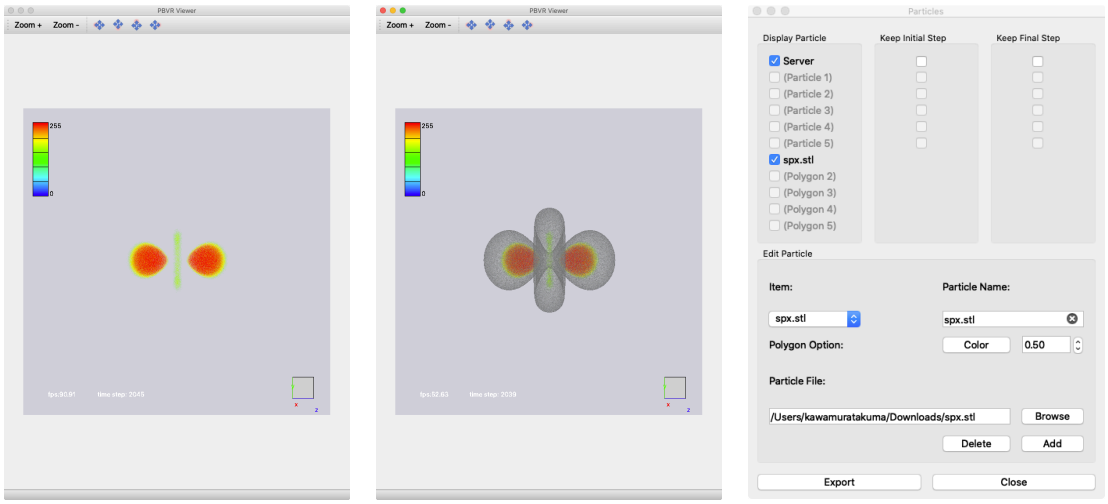


図 6-22 左：Hydrogen のボリュームレンダリング結果、中央：Hydrogen ボリュームと境界形状の合成表示、右：合成表示における粒子統合パネルの設定（spx->hydrogen に読み替えてください）

6.3.5.2 複数タイムステップ統合時の動作例

タイムステップが異なるデータを統合した場合の動作を、図 6-23（サーバ側：1～4タイムステップのデータ、クライアント側：0～3タイムステップの粒子データがある場合）で説明する。

Server チェックボックスと Particle チェックボックスの両方がチェックされている場合、全てのステップが表示対象となる。表 6.3-2 に表示されるタイムステップを示す。

クライアントの Keep Initial Step がチェックされている場合、はじめのタイムステップが表示され続ける。表 6.3-3 に表示されるタイムステップを示す。

クライアントの Keep Final Step がチェックされている場合、最後のタイムステップが表示され続ける。表 6.3-4 に表示されるタイムステップを示す。

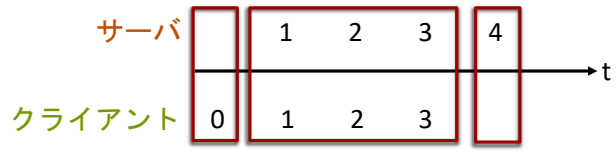


図 6-23　タイムステップの異なる粒子統合の動作

表 6.3-2　デフォルトで表示されるタイムステップ

	ステップ0	ステップ1	ステップ2	ステップ3	ステップ4
サーバ	なし	1	2	3	4
クライアント	0	1	2	3	なし

表 6.3-3　サーバの Keep Initial Step で表示されるタイムステップ

	ステップ0	ステップ1	ステップ2	ステップ3	ステップ4
サーバ	1	1	2	3	4
クライアント	0	1	2	3	なし

表 6.3-4　クライアントの Keep Final Step で表示されるタイムステップ

	ステップ0	ステップ1	ステップ2	ステップ3	ステップ4
サーバ	なし	1	2	3	4
クライアント	0	1	2	3	3

6.3.6. 画像ファイル作成

動画作成用パネルで、ビューワに表示した内容をキャプチャして保存し、動画として再生できる。動画作成用パネルはメインパネルの Animation Control Panel ボタンを押すと現れる。以下の２つのモードでのキャプチャができる。

- イメージキャプチャ

ビューワに表示した内容を時刻毎の連番画像ファイルとして保存する。画像ファイルの形式は BMP である。保存した一連の画像ファイルは、ImageMagic の convert や ffmpeg 等の外部コマンドを用いて、mpeg 等の動画ファイルとして圧縮できる。

- キーフレームアニメーション

ビューワに表示した内容に対応するビュー情報を、任意のタイミングでキーフレームとして保存する。保存した一連のキーフレームは動画として再生できる。

動画作成用パネルの内容を図 6-24 に示す。パネル内の各項目について以下に説明する。

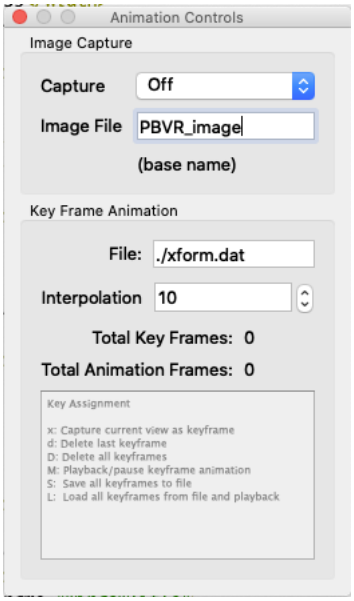


図 6-24　動画作成用パネル

・Capture プルダウンメニュー

イメージキャプチャの開始／終了を措定する。選択肢は off または on。

・Image File

イメージキャプチャした内容を連番画像ファイルとして保存するときのファイル名の、ベース名の部分を指定する。省略値は「PBVR_image」。

・File

キーフレームアニメーションでキャプチャしたキーフレームを保存するファイル名を指定する。省略値は「./xform.dat」。

・Interpolation

キーフレームアニメーション再生時にキーフレーム間のビューを補間するときのフレーム数を指定する。省略値は 10。補間は線形・等間隔で行う。

・Total Key Frames

キーフレームアニメーションで現在キャプチャされているキーフレームの数を表示する。初期値は 0。x キーの操作が成功する度に現在値+1、d キーの操作が成功する度に現在値-1、D キーの操作が成功すれば 0 という表示結果になる。

・Total Animation Frames

キーフレームアニメーション再生時のフレーム数を表示する。フレーム数は、

（Total Key Frames の値 － 1）＊ Interpolation の値

である。

6.3.6.1 イメージキャプチャ

イメージキャプチャを行うときの操作方法を説明する。

【操作方法】

① image file で、連番画像として保存するファイル名のベース名の部分を指定する。
② capture プルダウンメニューで on を選択する。
③ 時刻の更新毎にビューワの表示内容が連番画像として保存される。
④ capture プルダウンメニューで off を選択すると、連番画像の保存を停止する。

連番画像は、クライアント起動時に-iout オプションで指定したディレクトリの下に保存される。-iout を省略した場合、クライアントを起動したディレクトリの直下に連番画像が保存される。

file で指定したベース名が PBVR_image の場合、保存される連番画像の名前は以下のようになる。

PBVR_image.00001.bmp
PBVR_image.00002.bmp
:
:

後述するキーフレームアニメーションの再生をイメージキャプチャした場合は、以下に示す例のように、連番画像として保存するファイル名のベース名の最後に ”_k” が補われる。

PBVR_image_k.00001.bmp
PBVR_image_k.00002.bmp
:
:

6.3.6.2 静止画のキーフレームアニメーション

タイムステップ制御パネルの stop ボタンを押した状態の静止画からキーフレームアニメーションを作成するときの操作方法を説明する。

【キーフレームをキャプチャし、ファイルに保存する】

① file で、キーフレームを保存するファイル名を指定する。
② ビューワウィンドウをマウスカーソルでクリックしてアクティブにする
③ キーフレームとしてキャプチャしたい描画内容に対して、キーボードのｘキーを押す。これによりｘキーを押した時点の描画内容に対応するビュー情報をキーフレームとしてメモリ上に保存する。
④ ③の操作を必要な数だけ繰り返す。
⑤ キーボードのＭ（Shift+m）キーを押す。これによりメモリ上に保存したキーフレームをアニメーションとして再生する。
⑥ 再生内容に問題が無ければ、キーボードのＳ（Shift+s）キーを押して、キーフレームをファイルに保存する。

【ファイルに保存したキーフレームを再生する】

① file で、キーフレームが保存されているファイル名を指定する。
② ビューワウィンドウをマウスカーソルでクリックしてアクティブにする
③ キーボードのＬ（Shift+f）キーを押す。これによりファイルに保存したキーフレームをアニメーションとして再生する。

Ｆキーを押した後にｘキーを押すと、ファイルからメモリ上に読み込んだキーフレームに対して新たなキーフレームを追記する。

キーフレームアニメーションで使うキーの機能を表 6.3-5 に示す。

表 6.3-5　キーフレームアニメーションの操作キー

キー	機能
x	現在のビューワの表示に対応するビュー情報をキーフレームとしてメモリ上に保存する
d	メモリ上の最終キーフレームを削除する
D	メモリ上の全てのキーフレームを削除する
M	メモリ上のキーフレームをアニメーション表示／一時停止する
S	メモリ上のキーフレームをファイルへ保存する
L	ファイルからメモリ上に読み込んだキーフレームをアニメーション表示する

6.3.6.3 時系列データのキーフレームアニメーション

時系列データのキーフレームアニメーションを作成する場合の動作について説明する。

① 時系列データを描画中に x キーを押すと、キーフレーム情報として、ビューと共に、現在表示している時系列オブジェクトのタイムステップ番号を保存する。ただしオブジェクトのファイル名は保存しない（これにより、オブジェクト A で作成したキーフレーム情報をオブジェクト B に対して適用することができる）。

② S キーを押したときに、ビューと共に、タイムステップ番号もキーフレーム情報としてファイルへ保存する。

③ F キーを押したときに、ビューと共に、タイムステップ番号もメモリに読み込んでキーフレームアニメーションを開始する。これにあたり、パネルの interpolation で指定したコマ数でタイムステップ番号を補間しながら、対応するタイムステップ番号のオブジェクトをキーフレームとして読み直す。

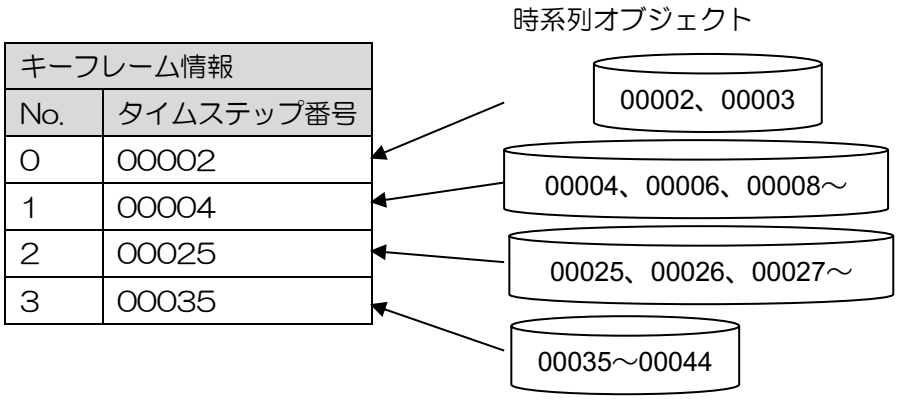
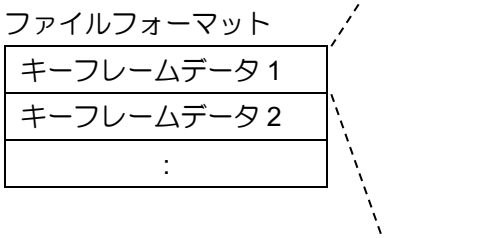


図 6-25　時系列データのキーフレームアニメーション構成

図 6-24　動画作成用パネルの例で interpolation を 10 フレームに指定した場合、キーフレーム No.0 と No.1 の間のビュー情報を補間した 10 個のビューをタイムステップ番号 00002 と 00003 のデータに割り当てて 5 フレームずつ表示する。次に、キーフレーム No.1 と No.2 の間の 10 フレームに関しては 10 個のビューを時間方向に等間隔に割り当てて、タイムステップ番号 00004、00006、…000024 のデータを表示する。

6.3.6.4 キーフレーム情報ファイルのフォーマット

キーフレーム情報を保存するファイル（省略値は「./xform.dat」）はバイナリファイルである。そのファイルフォーマットを図 6-26 に示す。



型	サイズ (byte)	用途
int	4	タイムステップ
float	4	rotation[0].x
float	4	rotation[0].y
float	4	rotation[0].z
float	4	rotation[1].x
float	4	rotation[1].y
float	4	rotation[1].z
float	4	rotation[2].x
float	4	rotation[2].y
float	4	rotation[2].z
float	4	translation.x
float	4	translation.y
float	4	translation.z
float	4	scaling.x
float	4	scaling.y
float	4	scaling.z

図 6-26　キーフレーム情報ファイルのフォーマット

6.3.7. レジェンドパネル

物理量と色との対応を示すカラーバーを表示するレジェンドパネルを図 6-27 に示す。レジェンドパネルはメインパネルの Legend Panel ボタンを押すと現れる。パネル内の各項目について以下に説明する。

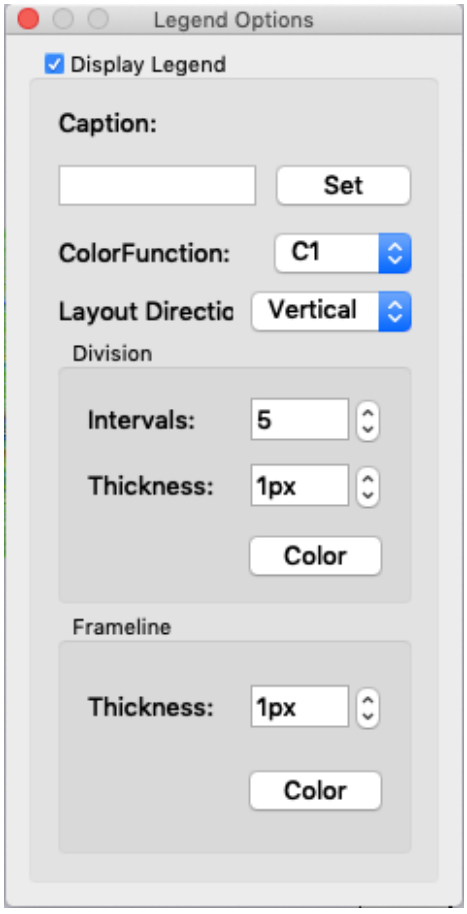


図 6-27　レジェンドパネル

- Display Legend チェックボックス
  レジェンド表示の On/Off を指定する。
- Caption テキストボックス
  レジェンドに対するキャプション文字列を入力する。Set ボタンで内容が反映される。
- Color Function：スピンボタン
  レジェンドのカラーマップと値域を定義する伝達関数を選択する。
- Layout Direction スピンボタン
  レジェンドの向き（縦／横）を選択する。
- Division
  レジェンドの目盛線に関する属性を指定する。
  Intervals：目盛線の数
  Thickness：目盛線の太さ
  Color：目盛線の色

・Frameline

レジェンドの枠線に関する属性を指定する。

Thickness：枠線の太さ

Color：枠線の色

レジェンドの表示例を図 6-28　レジェンドの表示例図 6-28 に示す。

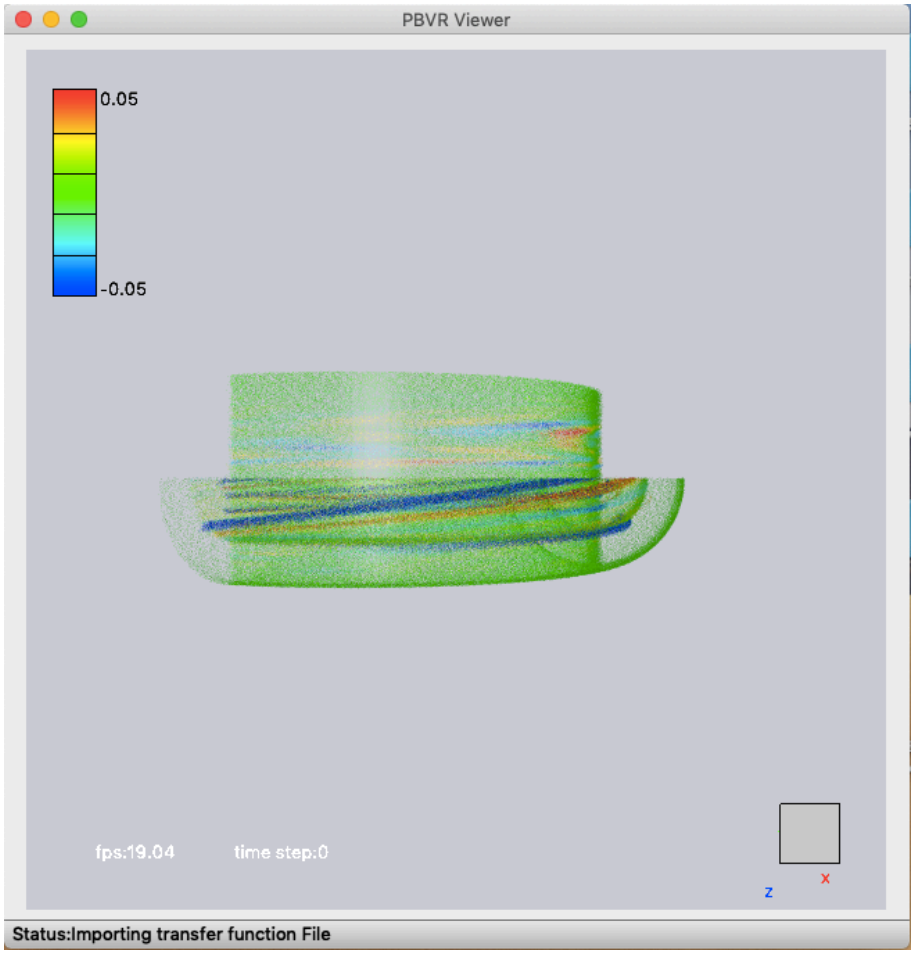


図 6-28　レジェンドの表示例

6.3.8. ビューワ制御パネル

ビューワの属性を制御するビューワ制御パネルを図 6-29 に示す。ビューワ制御パネルはメインパネルの Viewer Control Panel ボタンを押すと現れる。パネル内の各項目について以下に説明する。

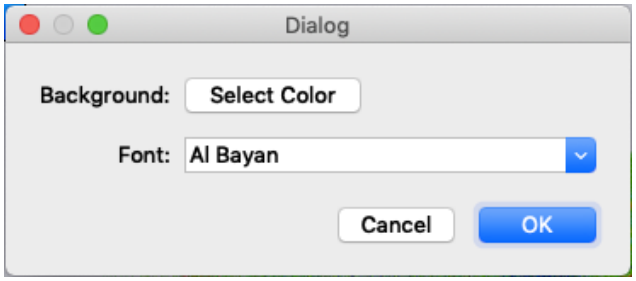


図 6-29　ビューワ制御パネル

• Background

ビューワの背景色を指定する。

• Font

ビューワに表示される文字の Font の種類とサイズを選択する。

7 サンプルの実行

ソースコードパッケージに含まれる Example を用いて、Linux/Mac から富岳にポートフォワード接続し、In-Situ PBVR を実行する例を示す。ここでは、プログラム実行が許可されたログインノードを利用する方法と、対話ジョブを利用する方法を紹介する。また Windows から遠隔のサーバにアクセスし In-Situ PBVR を実行する例を示す。

7.1. プログラム実行が許可されたログインノード

富岳上でプログラム実行が許可されたログインノードにログインし、非構造格子用のテストプログラム（is_pbvr/Example/Hydrogen_unstruct）を実行する手順を示す。この例では、RSA 公開鍵及び ssh のホスト名・ユーザ名の設定が済んでおり、ソースパッケージが富岳上の /home/ に配置され、ビルドが完了しているものとする。この手順では、In-Situ PBVR を結合したテストプログラムをジョブ投入し、ログインノード上でデーモンを、ユーザ PC 上で PBVR クライアントを起動する。

ターミナルを立ち上げ、富岳にログインし、投入スクリプト“run.sh”を用いてテストプログラムをジョブ投入する。

[ターミナル１]

```
[userPC]$ ssh ユーザ ID@login.fugaku.r-ccs.riken.jp
[fugaku]$ cd /home/is_pbvr/Example/Hydrogen_unstruct
[fugaku]$ mkdir particle_out
[fugaku]$ cat run.sh
#!/bin/sh
#PJM -L "node=1"
#PJM -L "rscunit=rscunit_ft01"
#PJM -L "rscgrp=small"
#PJM -L "elapse=0:60"
#PJM --mpi max-proc-per-node=4,proc=4
#PJM -s
export OMP_NUM_THREADS=12
export TF_NAME=default
export VIS_PARAM_DIR=$PJM_JOBDIR
export PARTICLE_DIR=$PJM_JOBDIR/particle_out
mpiexec -n 4 ./run qsub run.sh
[fugaku]$ pjsub run.sh
```

２つ目のターミナルで、プログラム実行が許可されたログインノードとユーザ PC をポートフォワード接続しデーモンを起動する。

[ターミナル２]

```
[userPC]$ ssh -L 61000:localhost:61000 ユーザ ID@loginX.fugaku.r-ccs.riken.jp
[fugaku]$ source /opt/intel/bin/compilervars.sh intel64
[fugaku]$ cd /home/is_pbvr/Daemon
[fugaku]$ export VIS_PARAM_DIR=/home/is_pbvr/Example/Hydrogen_unstruct
[fugaku]$ export PARTICLE_DIR=$VIS_PARAM_DIR/particle_out
[fugaku]$ ./pbvr_daemon -p 61000
```

X は 1~6

Intel コンパイラを active にする

３つ目のターミナルで、ユーザ PC 上の PBVR クライアントを起動する。

[ターミナル３]

```
[usrPC]$ ./pbvr_client -p 61000
```

7.2. 対話ジョブ

１つ目のターミナルを立ち上げ、富岳にログインし、投入スクリプト “run.sh” を用いてテストプログラムをジョブ投入する。

[ターミナル１]

```
[userPC]$ ssh FUGAKU
[fugaku]$ cd /home/is_pbvr/Example/Hydrogen_unstruct
[fugaku]$ mkdir particle_out
[fugaku]$ cat run.sh
#!/bin/sh
#PJM -L "node=1"
#PJM -L "rscunit=rscunit_ft01"
#PJM -L "rscgrp=small"
#PJM -L "elapse=0:60"
#PJM --mpi max-proc-per-node=4,proc=4
#PJM -s
export OMP_NUM_THREADS=12
export TF_NAME=default
export VIS_PARAM_DIR=$PJM_JOBDIR
export PARTICLE_DIR=$PJM_JOBDIR/particle_out
mpiexec -n 4 ./run qsub run.sh
```

富岳はログインノード上でのプログラム実行が許可されていないため、計算ノードとユーザ PC をポートフォワード接続し、対話ジョブでデーモンを実行する。

ターミナル２で対話ジョブを起動し、対話ジョブ実行中の計算ノードの IP アドレスを取得する。ip コマンドにより表示されるネットワーク情報の中で、inet の後ろにある IP アドレスが計算ノードの IP アドレスである。

[ターミナル２]

```
[userPC]$ ssh FUGAKU
[fugaku]$ cd /home/is_pbvr/Example/Hydrogen_unstruct
[fugaku]$ ln -s /home/is_pbvr/Daemon/pbvr_daemon
[fugaku]$ pjsub --interact -L "node=1" -L "rscgrp=int" ¥
[fugaku]$ -L "elapse=0:10:00" --sparam "wait-time=600"
[intrct]$ ip address show tofu1
.....
    inet **.**.**.**/12 brd 10.255.255.255 scope global noprefixroute tofu1
.....
```

計算ノードの IP アドレスを用いて、ターミナル３でユーザ PC と計算ノードをポートフォワード接続する。

[ターミナル３]

```
[userPC]$ ssh -L 60000: **.**.**.**:60000 FUGAKU
```

ターミナル２でデーモンを実行する。

[ターミナル２]

```
[intrct]$ export TF_NAME=default
[intrct]$ export VIS_PARAM_DIR=$PJM_JOBDIR
[intrct]$ export PARTICLE_DIR=$PJM_JOBDIR/particle_out
[intrct]$ ./pbvr_daemon
```

ターミナル４で、ユーザ PC 上の PBVR クライアントを起動する。

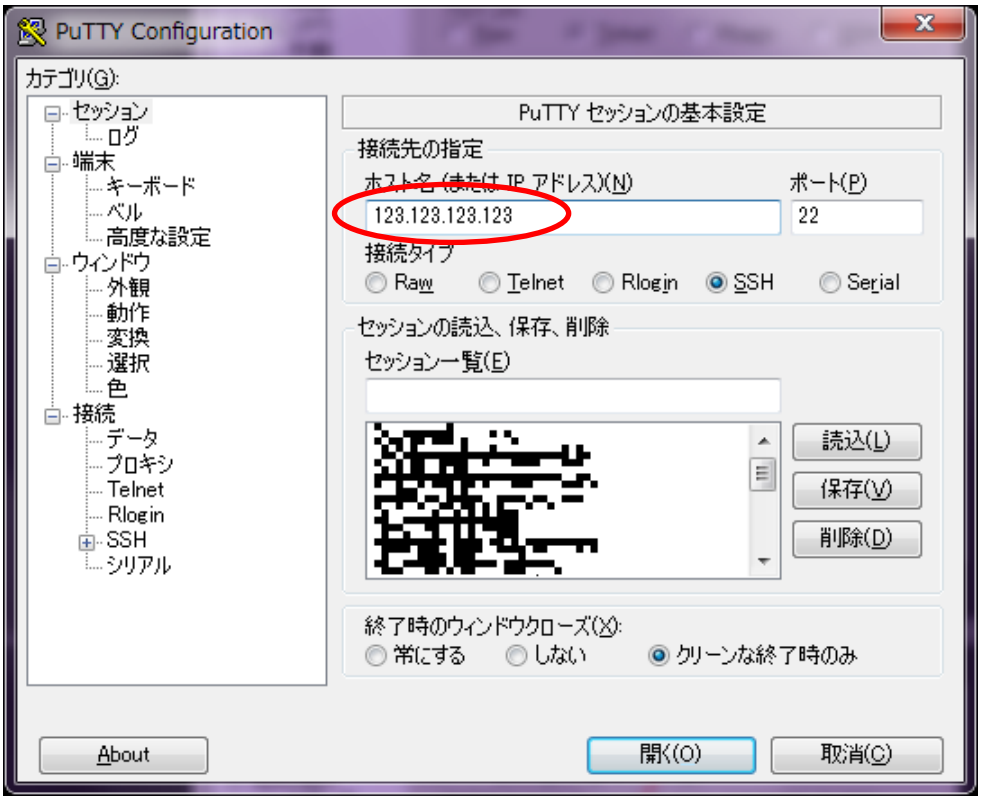
[ターミナル４]

```
[userPC]$ ./pbvr_client -p 60000
```

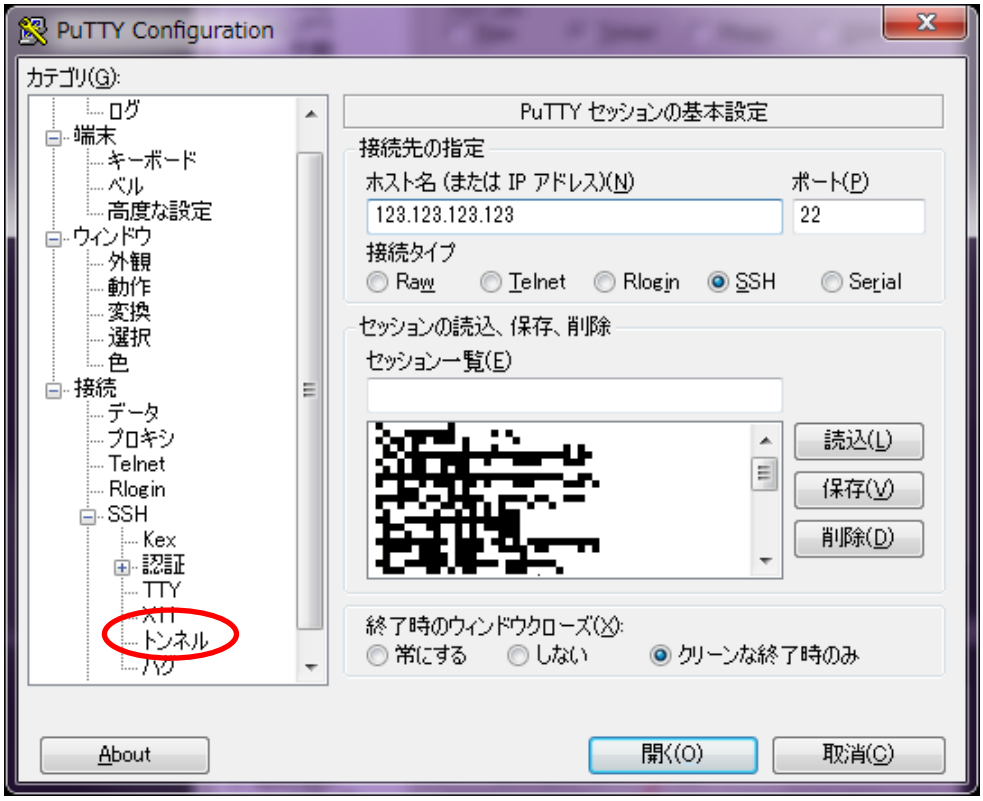
7.3. Windows

Windows クライアントから遠隔サーバにリモート接続する際の手順を以下に示す。尚、以下の手順では、クライアントのポート 60000 からサーバのポート 60000 に転送するものとしている。

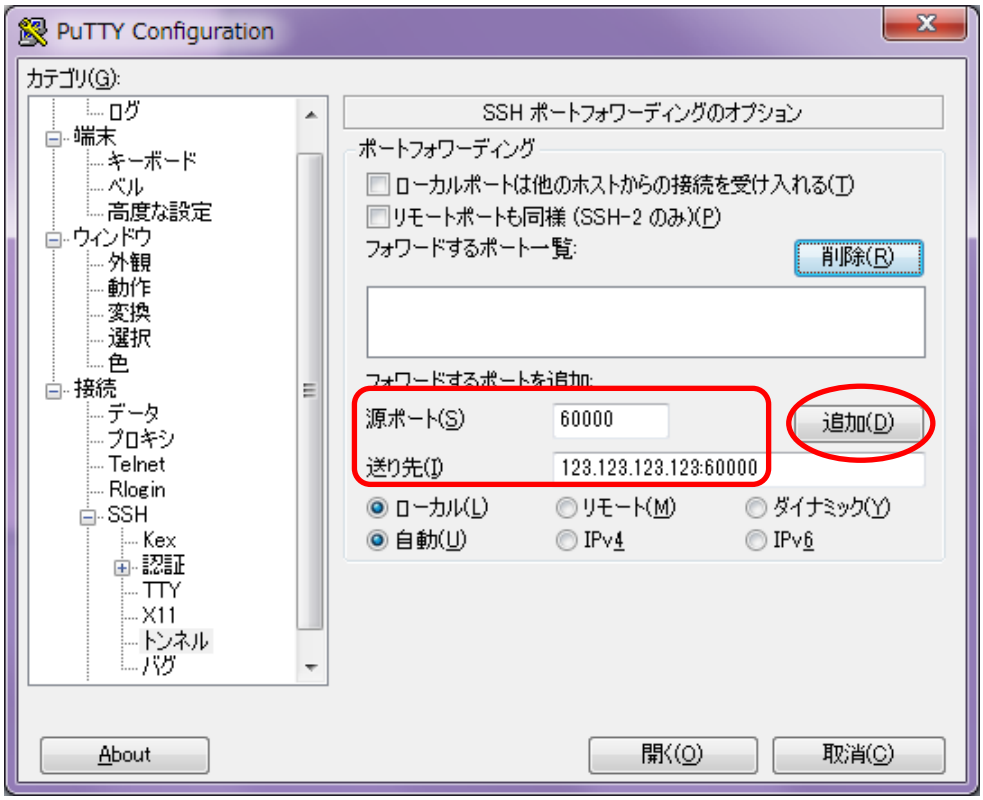
① puttyjp を起動し、接続するサーバのアドレスを入力する。



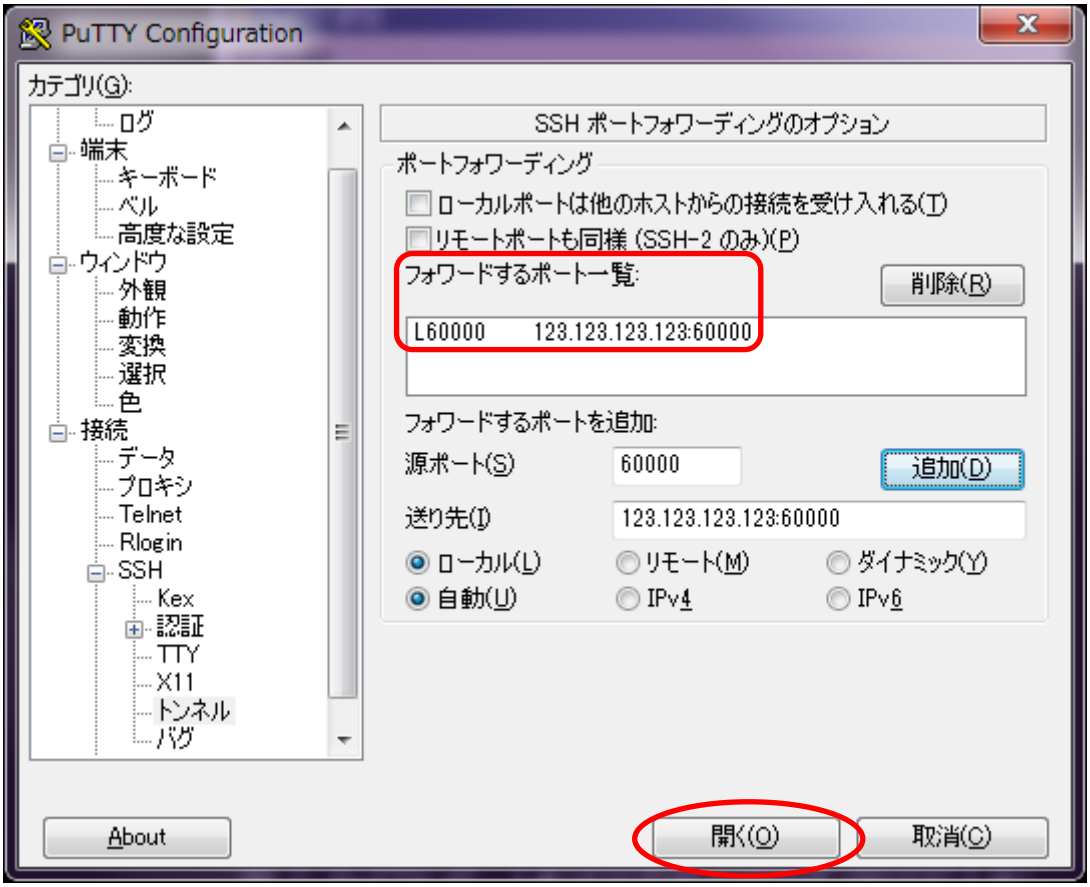
② 左側のカテゴリから「接続 ⇒ SSH ⇒ トンネル」を選択する。



③ 源ポートに転送元のポート番号、送り先にサーバのポートを設定し、「追加」ボタンを押下する。



④ フォワードするポート一覧にポート転送の設定が追加されたら、「開く」ボタンを押下してサーバと接続する。



⑤ サーバに接続しているターミナルからデーモンを起動する。

```
$ cd $HOME/JAEA/pbvr_inSitu_1.00/Example_C/Hydrogen
$ export VIS_PARAM_DIR=$HOME/JAEA/pbvr_inSitu_1.00/Example_C/Hydrogen
$ export PARTICLE_DIR=$HOME/JAEA/pbvr_inSitu_1.00/Example_C/Hydrogen
/jupiter_particle_out
$ export TF_NAME=jupiter
$ ./pbvr_daemon -p 60000
```

⑥ Visual Studio 2017 付属の「VS 2017 用 x64_x86 Cross Tools コマンド プロンプト」を起動し、以下のコマンドを実行する。

```
>set TIMER_EVENT_INTERVAL=1000
>cd C:¥pbvr¥pbvr_inSitu_work¥x64¥Release
>pbvr_client.exe -p 60000
```

⑦　サーバに接続しているターミナルからテストコードを起動する。

```
$ cd $HOME/JAEA/pbvr_inSitu_1.00/Example_C/Hydrogen
$ export VIS_PARAM_DIR=$HOME/JAEA/pbvr_inSitu_1.00/Example_C/Hydrogen
$ export PARTICLE_DIR=$HOME/JAEA/pbvr_inSitu_1.00/Example_C/Hydrogen
/jupiter_particle_out
$ export TF_NAME=jupiter
$ mpiexec -n 4 ./run
```