Remote Visualization Tool PBVR (v1.10)

User Guide

September 2017

Japan Atomic Energy Agency

Center for Computational Science & e-Systems

Revision Record

Version number	Date revised	Revised chapter	Revised content
1.04	2015.3.31	-	Release
1.05	2015.5.18	5.2	Parameter file function is added on Server
		5.4.2	File button is added on MainPanel
		5.4.6	Particle data output function is added
1.06a	2015.10.30	5.4.2	CROP panel is enabled for Mac
		5.4.3	Histogram function is added on Transfer Function Editor
		5.4.5	Image file production function is updated (file output directory, key frame animation)
1.07	2016.2.1	1.2	ICEX and VTK library are added to the platform list
		2	The package is updated including serial versions and PBVR Filter for VTK is added
		3.4	Data formats and parameter files are extended including STL, PLOT3D, and VTK
		4.2	A new command line option “-pd” is added, “-plimit” is modified, and “-sl” is removed
		5.2	A new command line option “-pd” is added, “-plimit” is modified, “-sl” is removed, and “-pin” is extended up to “-pin10”
		5.4.5	Particle panel is added
		6.4	An example of particle integration is added
1.08	2016.4.25	2	Updated source code package. Change compile / install method
		5.4.2	Following three buttons are added on MainPanel. “Transfer Function Editor”, “Particle Panel”, and “Animation Control Panel”.
1.09	2017.3.2	4.2	A new command line options “-Bs”, “-Be” and “-Bd” are added
		4.2.1	A usage of command line option “-vin” is extended for multiple pfi files.
		4.2.2	processing method of distributed file is added

		5.2	**Command line option “-vin” is extended for multiple pfi files.**
		5.4.2	**“no-repeat sampling until Transfer Functions be edited” checkbox is added on MainPanel, and following three buttons are added on MainPanel.** **“Legend Panel”, “Coordinate Panel”, and “Viewer Control Panel”.** **Display Particle Number is added.**
		5.4.3	**Change the way to activate the transfer function editor and add the Close button.**
		5.4.3.3	**An action when NaN appears by the arithmetic processing of function editor is added.**
		5.4.5	**Change the way to activate the particle integration editor and add the Close button.**
		5.4.6	**Change the method to activate the animation creation panel and add the Close button.**
		5.4.7	**A new panel “Legend panel” is added.**
		5.4.8	**A new panel “Coordinate panel” is added.**
		5.4.9	**A new panel “Viewer Control panel” is added.**
1.10	**2017.3.15**	**3**	**Delete “filter”** **Deletion by Integration of “Filter program” and “Server program”**
		3	**“IN/OUT files” are added.**
		5.4.2	**“Main Panel” is changed.**
		5.4.3	**Editor of “Transfer function” is changed.** **“Function changes” and “GUI changes” by TFS function expansion works**
		4.2	**Launching Method: Program Arguments:** **“pin deletion”, “fin addition”** **Changes by Integration of Filter program and server program**
		5.2	**Launching Method: Program Arguments:** **“pin deletion”, “fin addition”** **Changes by Integration of filter program and server program**

		6.1	**Launching Program: Program arguments: “pin deletion”, “fin addition”** **Changes by Integration of filter program and server program**
		6.5	**Batch mode processing: Program arguments: “pin deletion”, “fin addition”** **Changes by Integration of filter program and server program**

Table of Contents

1 Introduction

1.1 Overview

This document is a user guide for Particle Based Volume Rendering (PBVR), a remote visualization system developed at the Center for Computational Science & e-Systems in Japan Atomic Energy Agency. PBVR provides high-speed remote visualization of large-scale volume data by making use of the KVS library, and by employing the particle-based rendering algorithm from the Koyamada Visualization Laboratory in Kyoto University. PBVR consists of the following three components.

1) PBVR Server
   Input volume data of analysis result, parallel visualization by PBVR method, and output particle data.
2) PBVR Client
   PBVR Client renders the particle data as images using Open GL.

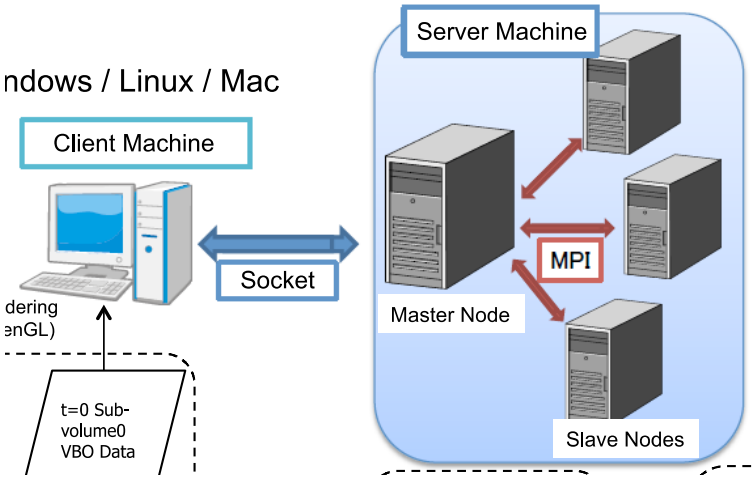


Figure 1 The system configuration of PBVR

1.2 System Requirements

The system is verified for the following platforms and compilers.

●PBVR Server

Platform	Compiler	Library
Linux 64bit	g++ version 4.4.6	KMATH, NTL *1
Mac 64bit *2	g++ version 4.8.2	
Windows 64bit*3	Visual Studio 2013	Visual C++ runtime component
K Computer	Fujitsu Compiler	KMATH, NTL *1
FX100	Fujitsu Compiler	KMATH, NTL *1
ICEX	Intel Compiler 15.0.3	KMATH, NTL *1

●PBVR Client

Platform	Compiler	Library
Linux 64bit	g++ version 4.4.6	OpenGL, GLUT
Mac 64bit *2	g++ version 4.8.2	OpenGL, GLUT
Windows 64bit*3	Visual Studio 2013	OpenGL, GLUT, Visual C++ runtime component

*1. KMATH is a high-performance pseudorandom number generator library, which was developed at RIKEN Advanced Institute for Computational Science. Installing KMATH further requires the NTL library.

*2. The current version was verified on Marverics, Yosemite, and El Capitan. On Macs, OpenMP is not available for the default gcc that is shipped with Xcode. To use the load modules or to compile the source code, install a newer version of gcc that works with OpenMP. The prebuilt binaries were compiled with gcc48 installed through MacPorts. To obtain this gcc version, run the following commands with root privilege in the terminal'.
port install gcc48
port select –set gcc mp-gcc48

*3. The current version was verified on Windows 7, 8.1, and 10. The Visual c++ runtime component is needed on Windows without Visual Studie 2013.

*4. VTK6.0 or later version is needed for compiling VTK server program.

2 Installation

PBVR consists of a load module package and a source code package. The following sections show how to install them.

2.1 Installation of Load Modules

The server program is implemented in C ++, and “sequential processing version”, “OpenMP version for thread parallel processing” and “MPI + OpenMP version for massively parallel processing” are prepared.
The client program is implemented by C ++ and OpenGL.

2.1.1 Server

Load Modules of the server program are prepared with “sequential processing version”, “OpenMP version for thread parallel processing” and “MPI + OpenMP version for massively parallel processing”. The installation is completed by copying the load modules of each environment to the appropriate directory through which the pass passed. The configuration of the load modules is as follows.

Table 1 List of load modules for PBVR Server

Platform	**Parallelization**	**Name of load module**
Linux 64bit	Serial	pbvr_server_linux pbvr_server_linux_vtk
	OpenMP	pbvr_server_linux_omp pbvr_server_linux_omp_vtk
	MPI+OpenMP	pbvr_server_linux_mpi_omp pbvr_server_linux_mpi_omp_vtk
Mac 64bit	Serial	pbvr_server_mac pbvr_server_mac_vtk
	OpenMP	pbvr_server_mac_omp pbvr_server_mac_omp_vtk
Windows 64bit	Serial	pbvr_server.exe pbvr_server_vtk.exe
	OpenMP	pbvr_server_omp.exe pbvr_server_omp_vtk.exe
K Computer *1	OpenMP	pbvr_server_k_omp
	MPI+OpenMP	pbvr_server_k_mpi_omp

FX100 *1	OpenMP	pbvr_server_fx100_omp
	MPI+OpenMP	pbvr_server_fx100_mpi_omp
ICEX	OpenMP	pbvr_server_icex_omp
	MPI+OpenMP	pbvr_server_icex_mpi_omp

*1. The load modules for supercomputers are used only in computing nodes. Therefore, if the login nodes and the post-processing nodes are on Linux servers, use the load modules that are compiled for Linux.

*2. VTK library is required for VTK data server.

2.1.2 Client

The load modules of the client program are parallelized by pthread. The installation is completed by copying the load modules of each environment to an appropriate directory passed by the pass. The configuration of the load modules is as follows.

Table 2 List of load modules for PBVR Client Program

Platform	**Parallelization**	**Name of load module**
Linux 64bit	pthread	pbvr_client_linux
Mac 64bit	pthread	pbvr_client _mac
Windows 64bit *1	pthread	pbvr_client .exe

*1. For the Windows version, copy also ‘glut32.dll’ to the destination directory.

2.2 Installation from Source Code

Extract the source code package to an arbitrary directory and compile it using pbvr.conf, Makefile under PBVR / directory. Configure make with the pbvr.conf file and compile the server and client program. The configuration of PBVR /directory is as follows.

Table 3 Directory structure of source code package

Directorys / Files	Explanation
PBVR/	
KMATH/	Parallel random number generation library: KMATH
KVS/	Visualization library KVS
glui/	Widget library for GUI
FunctionParser/	Function editor library
Common/	Protocol, Communication, Common Library
PbvrFilter/	Filter library
Server/	Server Program

Client/	Client Program
arch/	Configuration file for each compile
pbvr.conf *1	Configuration file for make file
Makefile *1	Compile PBVR/ under source code

*1. In Windows environment, use solution file “pbvr.sln” for VisualStudio instead of Makefile and pbvr.conf.

Designate the function to be installed by changing the input value of the variable in pbvr.conf. A list of variables in pbvr.conf is as follows.

Table 4 List of the parameter in *pbvr.conf*

Parameter name	Value	Notes
PBVR_MACHINE	String	Setting File in the arch/ Directory
PBVR_MAKE_CLIENT	0 or 1	Presence support of the Client
PBVR_MAKE_SERVER	0 or 1	Presence support of the Server
PBVR_SUPPORT_KMATH	0 or 1	Presence support of the KMATH (Server only)
PBVR_SUPPORT_VTK	0 or 1	Presence support of the VTK (Server only)

*1. Since the parallel random number generation library KMATH can not be used in the Mac version and windows version server, we use sequential random number generation library TinyMT.

The variable PBVR_MACHINE specifies the configuration file of the Makefile stored under the arc / directory according to the compilation environment.

Table 5 List of compile configuration files

File name	Notes
Makefile_machine_gcc	Specifies compiler and compile option of gcc compiler for Serial version
Makefile_machine_gcc_omp	Specifies compiler and compile option of gcc compiler for OpenMP version
Makefile_machine_gcc_mpi_omp	Specifies compiler and compile option of gcc compiler for MPI+OpenMP version
Makefile_machine_intel	Specifies compiler and compile option of intel compiler for Serial version
Makefile_machine_intel_omp	Specifies compiler and compile option of intel compiler for OpenMP version
Makefile_machine_intel_mpi_omp	Specifies compiler and compile option of intel compiler for MPI+OpenMP version

Makefile_machine_fujitsu	Specifies compiler and compile option of fujitsu compiler for Serial version
Makefile_machine_fujitsu_omp	Specifies compiler and compile option of fujitsu compiler for OpenMP version
Makefile_machine_fujitsu_mpi_omp	Specifies compiler and compile option of fujitsu compiler for MPI+OpenMP version
Makefile_machine_icex	Specifies compiler and compile option on SGI ICE X for MPI+OpenMP version
Makefile_machine_icex_omp	Specifies compiler and compile option on SGI ICE X for Serial version
Makefile_machine_icex_mpi_omp	Specifies compiler and compile option on SGI ICE X for OpenMP version

The following table lists the stting file on each environment.

Table 6 List of the setting file on each environment

File name	Environment					
	Linux	Mac	ICEX	FX100	K *2	K(Pre/Post)
Makefile_machine_gcc	SC	SC	-	-	-	S
Makefile_machine_gcc_omp	SC	SC	-	-	-	S
Makefile_machine_gcc_mpi_omp	SC	-	-	-	-	S
Makefile_machine_intel	S	-	-	-	-	S
Makefile_machine_intel_omp	S	-	-	-	-	S
Makefile_machine_intel_mpi_omp	S	-	-	-	-	S
Makefile_machine_fujitsu	-	-	-	S	S	-
Makefile_machine_fujitsu_omp	-	-	-	S	S	-
Makefile_machine_fujitsu_mpi_omp	-	-	-	S	S	-
Makefile_machine_icex	-	-	S	-	-	-
Makefile_machine_icex_omp	-	-	S	-	-	-
Makefile_machine_icex_mpi_omp	-	-	S	-	-	-

*1. “SC” is specifies available compiling Server and Client.
“S” is specifies available compiling Server only.

*2. Use Pre / Post node when running client/server mode on K computer.

2.2.1.1 Linux / Mac

The procedure for compiling and installing on Linux and Mac is following.

1) Edit pbvr.conf under PBVR/ directory according to the function used in each environment. Here is an example of compiling clients and servers parallelized by OpenMP using gcc compiler.

■ Example *pbvr.conf*

```
PBVR_MACHINE=Makefile_machine_gcc_omp
PBVR_MAKE_CLIENT=1
PBVR_MAKE_SERVER=1
PBVR_SUPPORT_KMATH=0
```

2) Compile under PBVR/ directory as below

```
$ make
```

Execution modules are created under PBVR/ as follows.

Server: Server/pbvr_server

Client: Client/pbvr_client

3) Copy the created Execution modules to an appropriate directory passed by the path

2.2.1.2 Windows

The procedure for compiling and installing on Windows is follws.

1) Install GLUT(64 bit)
   i) Download *glut-3.7.6-bin_x64.zip(64bit)* from the link below.
   http://ktm11.eng.shizuoka.ac.jp/lesson/modeling.html
   ii) Extract the following files:
   glut.h
   glut32.lib
   glut32.dll
2) Extract *PBVR* on a Windows machine that has Visual Studio 2013 installed.
3) On the *PBVR* directory, run *¥¥pbvr.sln* and launch Visual Studio 2013.
4) Choose **Release** and **x64** from the pull-down list as shown in Figure 2.

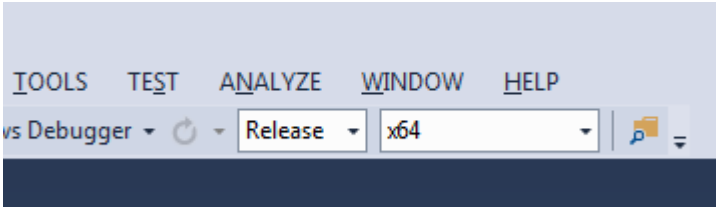


Figure 2 Build configuration of Visual Studio 2013

5) Go to the menu **Build** > **Build Solution**.

The load module *pbvr_server.exe*(Server) and *pbvr_client.exe*(Client) is created under *¥¥x64¥Release*.

2.2.1.3 Installing server program for VTK

VTK 6.0 or higher version is required to install the VTK server program. For installation of the VTK library, refer to the VTK website (http://www.vtk.org/). In addiotion, the following settings in CMake-gui is requrierd.

① Turn on options “BUILD_SHARED_LIBS”
② Make the option “BUILD_SHARED_TYPE” "Release"
③ Set “CMAKE_INSTALL_PREFIX” to the location where VTK is installed.

The necessary settings for the VTK server program are shown below.

pbvr.conf

Set the value of "PBVR_SUPPORT_VTK" to "1" (with VTK support) in the variable setting of the pbvr.conf file.

Excerpt from Table 5 List of compile configuration files

PBVR_SUPPORT_VTK	“1”	Presence support of the VTK (Server only)

Linux, Mac

Set environment variables as follows.

% export VTK_VERSION=n.n
% export VTK_LIB_PATH=/usr/local/lib
% export VTK_INCLUDE_PATH=/usr/local/include/vtk-n.n

Enter the version of VTK to be used in “n.n”.
The path should be change according to the installation environment of the VTK library.

Windows

Control Panel -> System -> Advanced -> Environment Variables.
Set the following environment variables.

Variable	**Value**
VTK_VERSON	**n.n**
VTK_LIB_PATH	**d:¥environments¥VTK¥lib**
VTK_INCLUDE_PATH	**d:¥environments¥VTK¥include¥vtk-n.n**

Enter the version of VTK to be used in “n.n”.
The path should be change according to the installation environment of the VTK library.

Launch Visual Studio 2013 and change properties as follows.

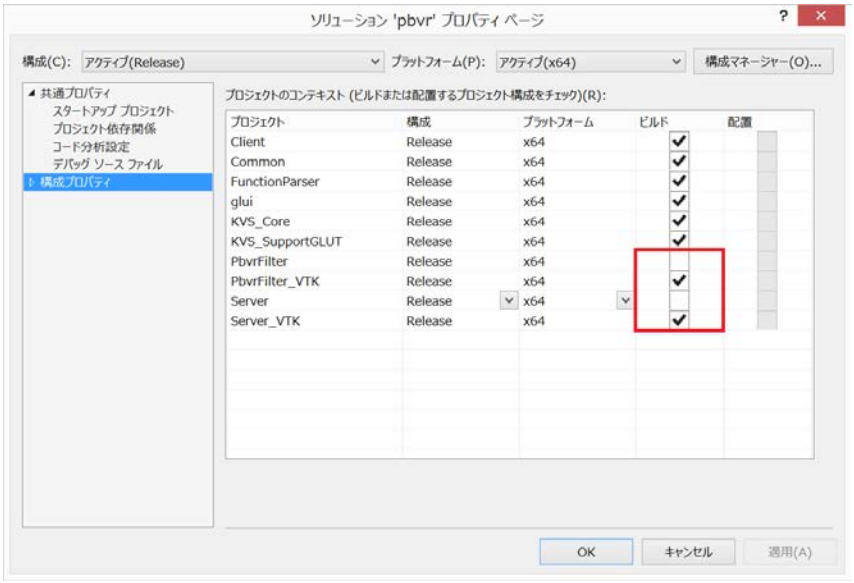
1. Open property page of solution ‘pbvr’ from solution explorer.
2. Select configuration property in property page of the solution ‘pbvr’, and set “build” of following project, and push “OK”.

[Select OK]

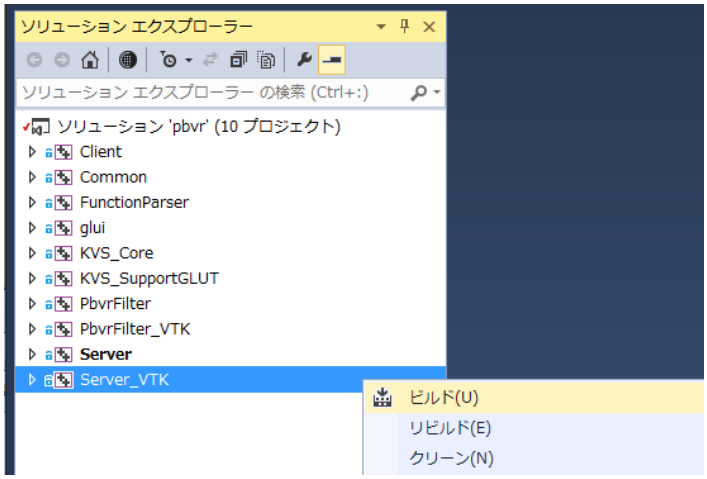
- “PbvrFilter_VTK” project
- “Server_VTK” project

[Select OFF]

- “PbvrFilter” project
- “Server” project



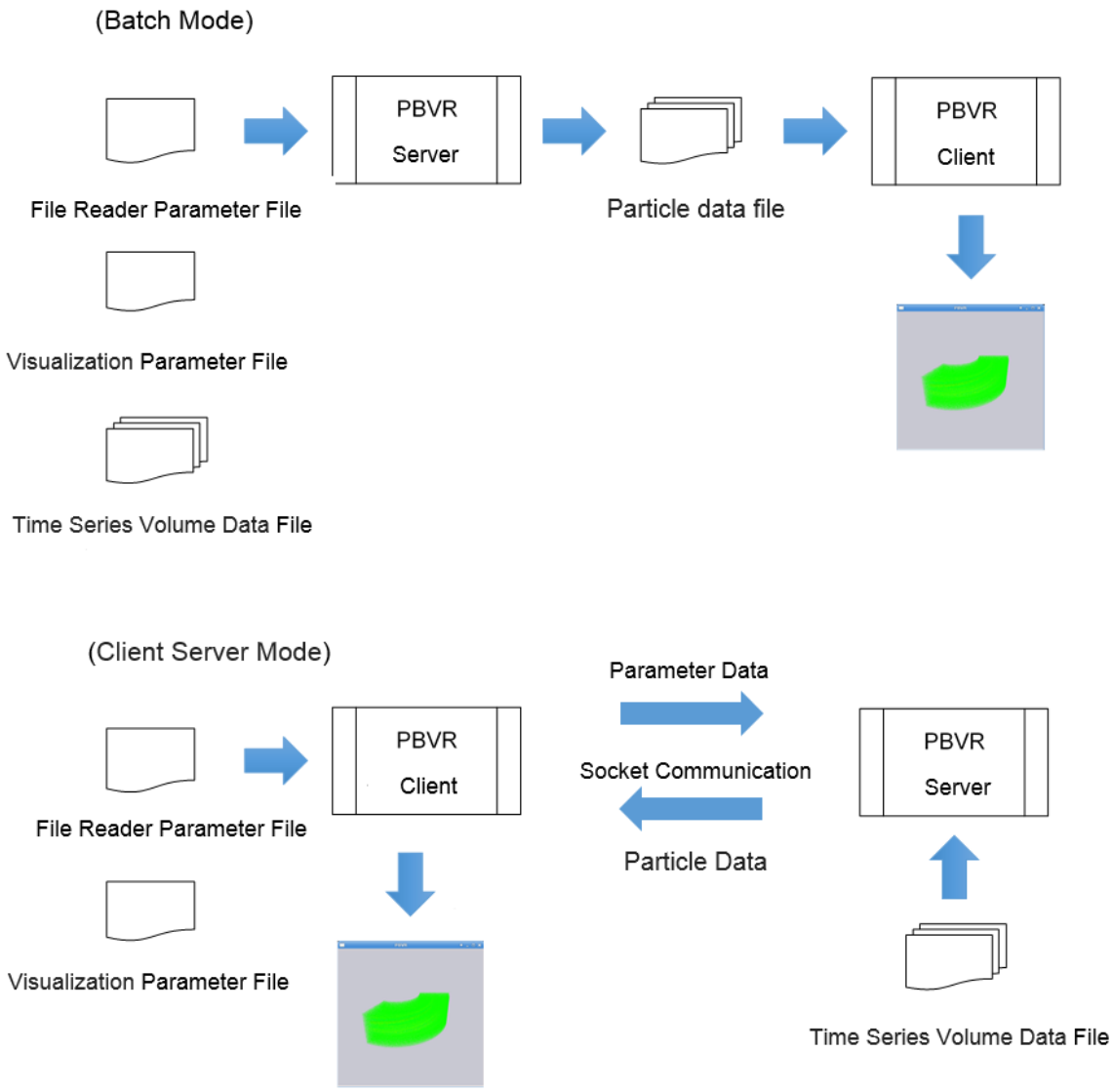
Select “Server_VTK” in solution explorer, and execute “build” from context menu.



In this case, pbvr_server.exe (server module for VTK) is created under x64¥Release. Please note that if there is already pbvr_server.exe, it is overwritten.

3 I/O Files

PBVR generates particle data by inputting a file reader parameter file describing a time series volume data file and a visualization parameter file for generating particle data.



1.) Batch mode

It is possible to draw the model at high speed with the PBVR client program by creating the particle data file from the time series volume data file by the PBVR server program beforehand.

2.) Client / server mode

By socket communication, particle data is generated by the PBVR server program, and the model is drawn by the client.

3.1 File Reader Parameter File

The file reader parameter file is an ASCII file commonly used for (AVSFLD / UCD, PLOT3D, STL) and VTK. By specifying a file name as an argument at run time, the file is interpreted as an input parameter and the parameter is set.

Table 7 List of File reader input parameter

Parameter name	Parameter detail	Default value	Notes
in_dir	Input file directory	'.'	Directory path of input files *1
field_file	AVSFLD file name	-	*2, *3, *4
stl_binary_file	STL file name	-	*2
Plot3d_config_file	PLOT3D configuration file name	-	*2, *3
vtk_file	VTK file name	-	*2, *3, *5
vtk_in_prefix	Prefix of time series VTK data files	-	*2, *3, *5
vtk_in_suffix	Suffix of time series VTK data files	-	*2, *3, *5
ucd_inp	AVSUCD file name	-	Ascii format*2
in_prefix	Prefix of time series AVSUCD data files	-	Binary format*2
in_sufix	Suffix of time series AVSUCD data files	-	Binary format*2
format	Step number format for time series data	"%05d"	
start_step	Starting step number	'1'	*6
end_step	Ending step number	'1'	*6
multi_elem_type	Flag on mixed element type unstructured grid	'0'	'0': data with a single element type '1': data with multiple element types

*1. Directories can be specified either with an absolute path or a relative path, although tilde (~) cannot be used as an abbreviation for the HOME directory.

*2. One of the following options, field_file, stl_binary_file, plot3d_config_file, vtk_file, vtk_in_prefix(suffix), ucd_inp, and in_prefix(suffix) should be given.

*3. When input data is two dimensional or three dimensional structured grid data, the output data is converted to unstructured grid data with linear quadrilateral or hexahedral elements, respectively.

*4. Only the parameters 'nstep', 'ndim', 'dim1', 'dim2', 'dim3', 'veclen', 'coord [123]', and 'variable' are referred.

*5. Five VTK Legacy data formats (VTK Structured Points, VTK Structured Grid, VTK Rectilinear Grid, VTK UnstructuredGrid, and VTKPolygonalData) are automatically recognized by PBVR Filter.

*6. Specified only for time series data.

3.1.1 PLOT3D Configuration File

PLOT3D data formats are described by a PLOT3D configuration file. Here, usebytecount should be chosen to be 1 and 0 for Fortran and C binary data, respectively.

Table 8 List of PLOT3D setting parameter

Parameter name	Parameter detail	Default value
coordinate_file_prefix	Prefix of coordinate file	-
coordinate_file_suffix	Suffix of coordinate file	-
coordinate_mode_precision	Precision (float | double)	double
coordinate_mode_usebytecount	1 for true, 0 for false	true
coordinate_mode_endian	Endian (little | big)	little
coordinate_mode_iblanks	1 for true, 0 for false	false
solution_file_prefix	Prefix of solution file	-
solution_file_suffix	Suffix of solution file	-
solution_mode_precision	Precision (float | double)	double
solution_mode_usebytecount	1 for true, 0 for false	true
solution_mode_endian	Endian (little | big)	little
function_file_prefix	Prefix of function file	-
function_file_suffix	Suffix of function file	-
function_mode_precision	Precision (float | double)	double
function_mode_usebytecount	1 for true, 0 for false	true
function_mode_endian	Endian (little | big)	little

3.1.2 PFL File

If you want to specify multiple file reader parameter files as input volume data files, make a PFL file that lists file reader parameter files and specify the PFL file with the option “-fin”.

The description of "#PBVR FILTER FILES" is required at the beginning of the PFL file. After that, the file reader parameter file is described by absolute path or relative path from the directory containing the file reader parameter file for each line. An example of description of PFL file is shown.

```
#PBVR FILTER FILES
hex.txt
hex2.txt
```

The extension of the PFL file must be "pfl".

3.2 Visualization Parameter File

In the PBVR client program, it is possible to save all the visualization parameters including the transfer function in the visualization parameter file.(6.5 reference)

The same visualization as at the time of storage can be made by reusing this visualization parameter file,

Table 9　Visualization parameter list

No	Item	Explanation	Output Example
1	SUB_PIXEL_LEVEL	SUB_PIXEL_LEVEL	2
2	PBVR_PARTICLE_LIMIT	PBVR_PARTICLE_LIMIT	10000000
3	SAMPLING_TYPE	SAMPLING_TYPE	3
4	EYE_POINT_X	EYE_POINT_X	0
5	EYE_POINT_Y	EYE_POINT_Y	0
6	EYE_POINT_Z	EYE_POINT_Z	0
7	CENTER_POINT_X	CENTER_POINT_X	0
8	CENTER_POINT_Y	CENTER_POINT_Y	0
9	CENTER_POINT_Z	CENTER_POINT_Z	0
10	UP_VECTOR_X	UP_VECTOR_X	0
11	UP_VECTOR_Y	UP_VECTOR_Y	1
12	UP_VECTOR_Z	UP_VECTOR_Z	0
13	RESOLUTION_WIDTH	RESOLUTION_WIDTH	620
14	RESOLUTION_HEIGHT	RESOLUTION_HEIGHT	620
15	FILTER_PARAMETER_FILENAME	File reader parameter file	param.txt
16	CROP_TYPE	CROP_TYPE	1
17	CROP_CXMIN	CROP_CXMIN	0
18	CROP_CXMAX	CROP_CXMAX	3
19	CROP_CYMIN	CROP_CYMIN	-2.98023e-08
20	CROP_CYMAX	CROP_CYMAX	2
21	CROP_CZMIN	CROP_CZMIN	-1.49012e-08
22	CROP_CZMAX	CROP_CZMAX	1
23	CROP_PBVR_SRADIUS	CROP_PBVR_SRADIUS	1.87083

24	CROP_SCENTERX	CROP_SCENTERX	1.5
25	CROP_SCENTERY	CROP_SCENTERY	1
26	CROP_SCENTERZ	CROP_SCENTERZ	0.5
27	CROP_PBVR_PRADIUS	CROP_PBVR_PRADIUS	1.80278
28	CROP_PBVR_PHEIGHT	CROP_PBVR_PHEIGHT	1
29	CROP_PCENTERX	CROP_PCENTERX	1.5
30	CROP_PCENTERY	CROP_PCENTERY	1
31	CROP_PCENTERZ	CROP_PCENTERZ	0.5
32	ROTATION1X	ROTATION1X	1
33	ROTATION1Y	ROTATION1Y	0
34	ROTATION1Z	ROTATION1Z	0
35	ROTATION2X	ROTATION2X	0
36	ROTATION2Y	ROTATION2Y	1
37	ROTATION2Z	ROTATION2Z	0
38	ROTATION3X	ROTATION3X	0
39	ROTATION3Y	ROTATION3Y	0
40	ROTATION3Z	ROTATION3Z	1
41	TRANSLATIONX	TRANSLATIONX	0
42	TRANSLATIONY	TRANSLATIONY	0
43	TRANSLATIONZ	TRANSLATIONZ	0
44	SCALINGX	SCALINGX	1
45	SCALINGY	SCALINGY	1
46	SCALINGZ	SCALINGZ	1
47	COORD1_SYNTH	COORD1_SYNTH	X
48	COORD2_SYNTH	COORD2_SYNTH	Y
49	COORD3_SYNTH	COORD3_SYNTH	Z
50	FONTTYPE	font	8x13
51	BACKGROUND_COLOR_R BACKGROUND_COLOR_G BACKGROUND_COLOR_B	Model display screen background color	200 200 210
52	TF_RESOLUTION	resolution	256
53	TF_NUMBER	Number Transsfer Function (Function limit number)	5
54	TF_SYNTH_C	First order transfer color function	C1+C2
55	TF_SYNTH_O	First order transmission opacity function	O1+O2

56	TF_NAME[n]_C	Color function definition name	C1
57	TF_NAME[n]_VAR_C	Color function variable expression	q1
58	TF_NAME[n]_MIN_C	Color function: Minimum granularity	0
59	TF_NAME[n]_MAX_C	Color function: maximum granularity value	500
60	TF_NAME[n]_TABLE_C	Color function: Color map	0,0,255,0,4,255,0,...
61	TF_NAME[n]_O	Opacity Function Definition Name	O1
62	TF_NAME[n]_VAR_O	Opacity function variable expression	q2
63	TF_NAME[n]_MIN_C	Opacity function: Minimum granularity	0
64	TF_NAME[n]_MAX_C	Opacity function: maximum granularity value	300
65	TF_NAME[n]_TABLE_C	Opacity function: Function curve	0,0.00392157,...
66	LEGEND_DISPLAY	Presence or absence of legend display 0= Hide, 1=Display	1
67	LEGEND_FUNCTION	Color function name in legend display（Change from numeric value to character string）	C2
68	LEGEND_POSITION	Legend display direction 0=Vertical, 1=Horizontal	0
69	LEGEND_CAPTION	Legend display character string	CAPTION_C2
70	LEGEND_INTERVALS	Legend Internal ruled line spacing	5
71	LEGEND_INTERVAL_THICKNESS	Legend Internal borders width	3
72	LEGEND_INTERVAL_COLOR_R LEGEND_INTERVAL_COLOR_G LEGEND_INTERVAL_COLOR_B	Legend Internal border color	248 14 42

73	LEGEND_FRAMELINE_THICKNESS	Legend frame width of the border	5
74	LEGEND_FRAMELINE_COLOR_R LEGEND_FRAMELINE_COLOR_G LEGEND_FRAMELINE_COLOR_B	Legend frame border color	8 6 245

[n] ＝1～99：The Number of Color / Opacity Function

3.3 Time Series Volume Data File

AVSFLD / UCD, PLOT3D, STL, and time-series volume data files described in the VTK format.

3.3.1 Input data format

Input data that can be processed with the PBVR program is as follows.

1) AVSFLD binary data *1
2) AVSFLD ASCII binary data *1
3) STL binary data *2
4) PLOT3D binary data *3
5) VTK Legacy binary data *4

*1 For details of the AVS data format, please refer to http://www.cybernet.co.jp/avs/products/avsexpress/dataformat.html etc. AVSUCD data is only data format, geom format, data_geom format is not supported. Element type corresponds to two-dimensional element, three-dimensional element, and mixing element is also available.
*2 For details of the STL data format, please refer to https://en.wikipedia.org/wiki/STL_(file_format) etc.
*3 For details of the PLOT3D data format, see http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19900013774.pdf etc.
*4 For details of the VTK data format, see http://www.vtk.org/ etc.
*5 VTK Structured Points, VTK Structured Grid, VTK Rectilinear Grid, VTK Unstructured Grid, and VTK PolygonalData are available.

3.3.2 Endian

The binary file handled by PBVR program is unified to little endian. On little endian machines there is no need for special handling on endian. But on big endian machines it is necessary to

output visualized data in little endian or convert it to little endian before running PBVR program or convert it to little endian and execute PBVR program.

4 PBVR Server

4.1 Overview

PBVR Server reads sub-volume files, which are produced by PBVR File Reader, and performs parallel visualization with the PBVR technique to generate particle data as visualization results.

4.2 Launching PBVR Server

PBVR can operate in supercomputers both in batch mode, which generates only particle data in batch processing, and in client-server mode, which generates particle data in interactive processing by connecting PBVR Client and PBVR Server via a socket communication. Stand-alone processing on PCs or workstations is also possible by launching PBVR Client and PBVR Server in the client-server mode on the same machine. The followings show how to launch PBVR Server.

Examples:

Launch the MPI+OpenMP version, and use *N* processes

```
$ mpiexec -n N pbvr_server
```

Launch the OpenMP version

```
$ pbvr_server
```

*1. Since the MPI+OpenMP version of PBVR Server operates with master-slave MPI processing, the number of process *N* should be specified by the number of slave process + 1.

*2. In both processing modes, the number of OpenMP threads is set with OMP_NUM_THREADS environment variable.

*3. In Windows, these commands should be launched from Visual Studio 2013 x64 Native Tools command prompt.

Table 10 List of command line options for the PBVR Server program

Option	Launch mode	Possible parameters	Default parameters	Functionality
-h	CS,B	-	-	This shows the list of available options and parameters
-B	B	-	-	To launch in the batch mode
-pa	B	File name	-	Visualization parameter file
-pd	B	Real number	1.0	Particle density *2

-S	B	u, m	u	Method for sampling particles u: uniform sampling m: metropolis sampling
-plimit	B	1-99999999	1000000	Maximum number of particles
-fin	B	File name	-	File reader parameter file *2
-p	CS	Port number	60000	Port number for socket communication
-pout	B	File name	./	Name of the output particle data file *3
-viewer	B	100-9999 ×100-9999	620×620	Viewer resolution
-Bd	B	-	-	Create particle files separately without integrating the subvolumes
-Bs	B	Integer of 0 or more	First step of specified file reader parameter file group	First time step to be rendered
-Be	B	Integer of 0 or more	Last step of specified file reader parameter file group	Last time step to be rendered

*1. In launch mode, CS and B denote client-server mode and batch mode, respectively.

*2. Specify the file reader parameter file as an absolute path or a relative path. This option overrides the options in the visualization parameter file specified with option “-pa”.

*3. This generates a set of particle data files with names
“[file name]_[time step]_[number of sub-volumes]_[sub-volume index].kvsml,”
where [file name] is the prefix specified with this option. If the prefix is omitted, the prefix ’server’ will be inserted automatically.

4.2.1 Launching PBVR Server in Batch Mode

When the command line option ‘-B’ is given, PBVR Server is launched in batch mode. The following example shows how to launch PBVR Server in the batch mode (for the MPI+OpenMP version).

```
$ mpiexec –n 5 pbvr_server –B -fin ./param.txti -pout ./output/case –pa ./param.in
```

In this example, the time series volume data described in the file reader parameter file ./param.txt is processed with the parameters described in the visualization parameter file ./param.in to output the following particle data.

./output/case_XXXXX_YYYYYYY_ZZZZZZZ.kvsml

XXXXX	：Number of steps（5 digit number）
YYYYYYY	：Index for the sub-volume（7 digit number）
ZZZZZZZ	：Total number of Sub-volumes（7 digit number）

Usually all of the subvolume for each time is integrated, so both YYYYYYY, ZZZZZZZ shall be 1. If you want to output particle data for each of the subvolume without the integration, command-line options '-Bd' must be specified the server startup of batch mode.
The visualization parameter file is specified with the command line option ‘-pa’. This file is generated in client-server mode interactively. Large-scale data processing in the batch mode is executed by using this file as it is, or with desirable modifications to the parameters.

4.2.2 Processing Distributed Files

Multiple volume data distributed and stored in storage can be integrated and visualized. Make a PFL file (extension: pfl) that describes the file reader parameter file of multiple volume data. Then specify the PFL file with "-fin" option.
The description of "#PBVR FILTER FILES" is required at the beginning of the PFL file, and thereafter, the file reader parameter file is described by absolute path or relative path from the directory where the PFL file exists.
A description example of PFL file is shown.

```
#PBVR FILTER FILES
hex.txt
hex2.txt
```

4.2.3 Launching PBVR Server in Client-Server Mode

When the command line option ‘-B’ is not specified, PBVR Server is launched in the client-server mode. See the following example.

```
$ mpiexec -n 5 pbvr_server
first reading time[ms]:0
Server initialize done
```

Server bind done
Server listen done
Waiting for connection ...

When “Waiting for connection” appears as in the above example and PBVR Server waits for a socket communications with PBVR Client, launch the PBVR Client in another terminal. In the client-server mode, input volume data name should be given to PBVR Client rather than to PBVR Server.
The default port number for the socket communication is 60000. To change the port number, use the command line option ‘-p’:

$mpiexec -n 5 pbvr_server -p 55555

4.2.4 Connecting Client and Server via Socket Communication

4.2.4.1 Local Connection

The following example shows how to launch both PBVR Client and PBVR Server on a single machine ‘machineA’. In this example, they cooperate using the default port number 60000 of ‘machineA’.

Step 1 [Launch PBVR Server]
machineA> mpiexec -n 5 pbvr_server
Step 2 [Launch PBVR Client]
machineA> pbvr_client -fin filename

4.2.4.2 Remote Connection between Two Machines

The following example shows how to launch PBVR Client on a machine ‘machineA’ and PBVR Server program on another machine ‘machineB’ where the two machines are located at distant places. This example uses SSH port forwarding to connect the port 60000 of ‘machineA’ to the port 60000 of ‘machineB’. In this way, PBVR Server and Client on the two machines cooperate through the default port number 60000. Once the SSH port forwarding is established, the launching procedure is basically the same as that in stand-alone mode. In Windows, SSH port forwarding can be setup by using a third-party application such as TeraTerm or Putty.

Step1 [SSH port forwarding from machineA to machineB]
machineA> ssh -L 60000:localhost:60000 username@machineB
（Forwarding the 60000 port of machineA to the 60000 port of machineB）
Step2 [Launch PBVR Server]
machineB> mpiexec –n 5 pbvr_server
Step3 [Launch PBVR Client]

machineA> pbvr_client –fin filename

4.2.4.3 Remote Connection with Several Machines

This section provides an example of connecting PBVR Server and PBVR Client on two remote machines ‘machineA’ and ‘machineB’ via ‘machineC’ for some reason, e.g. security. Once the SSH port forwading is established, the launching method is basically the same as the stand-alone mode, as wtih the two point remote concection mentioned before.

Step1 [SSH port forwarding from machineA to machineC]
machineA> ssh -L 60000:localhost:60000 username@machineC
（Forwarding the 60000 port of machineA to the 60000 port of machineC）
Step2 [SSH port forwarding from machineC to machineB]
machineC> ssh -L 60000:localhost:60000 username@machineB
（Forwarding the 60000 port of machineC to the 60000 port of machineB）
Step3 [Launch PBVR Server]
machineB> mpiexec –n 5 pbvr_server
Step4 [Launch PBVR Client]
machineA> pbvr_client –fin filename

4.2.4.4 Testing SSH Port Forwarding Connection

To check if SSH port forwarding is available, use the following test program, which simply transfers characters input from PBVR Server to PBVR Client. This program is available from the link below.

“C for Linux 2” Mitsuyuki Komata, SYUWA System, Inc., September 2005 (Japanese).
http://www.ncad.co.jp/~komata/c4linux2/

Launch PBVR Server
server port_number

Launch PBVR Client
client server_hostname port_number

4.2.4.5 Connecting to Pre-post Server of K computer

This section shows an example of connecting a PC (‘machineA’) in s laboratory to the data processing server of the K computer (Pre-post server pps3) via the login node of the K computer (klogin).

Step1 [SSH port forward from machine to klogin]
machineA> ssh -L 60000:localhost:60000 username@k.aics.riken.jp

（Forwarding the 60000 port of machineA to the 60000 port of klogin）

Step2 [SSH port forward from K login node to pre-post server]

klogin> ssh -L 60000:localhost:60000 username@pps3

（Forwarding the 60000 port of klogin to the 60000 port of pps3）

Step3 [Launch PBVR Server]

pps3> mpiexec –n 5 pbvr_server

Step4 [Launch PBVR Client]

machineA> pbvr_client –fin filename

（Forwarding the 60000 port of klogin to the 60000 port of pps3）

4.2.4.6 Local Connection in Windows

This section shows how to launch both PBVR Sever and PBVR Client on a single Windows machine. The Visual Studio 2013 x64 Cross Tools command prompt in Visual Studio 2013 is used as the terminal for launching the programs.

Step1 [Launch PBVR Server]

Windows> pbvr_server.exe

Step2 [Set the client parameter for Windows]

Windows> set TIMER_EVENT_INTERVAL=1000

Step3 [Launch PBVR Client]

Windows> pbvr_client.exe –fin filename

Another way of launching PBVR Server and Client is to execute a batch file with the following lines.

```
set TIMER_EVENT_INTERVAL=1000
start pbvr_server_win.exe
pbvr_client.exe –fin filename
```

4.2.4.7 Remote Connection from Windows Client

To connect PBVR Client in a Windows machine to PBVR Server in a remote machine, setup port forwarding with the help of an SSH client software such as TeraTerm or Putty. The following shows an example for TeraTerm.

1) Launch TeraTerm and hit cancel in the “New connection” dialog.

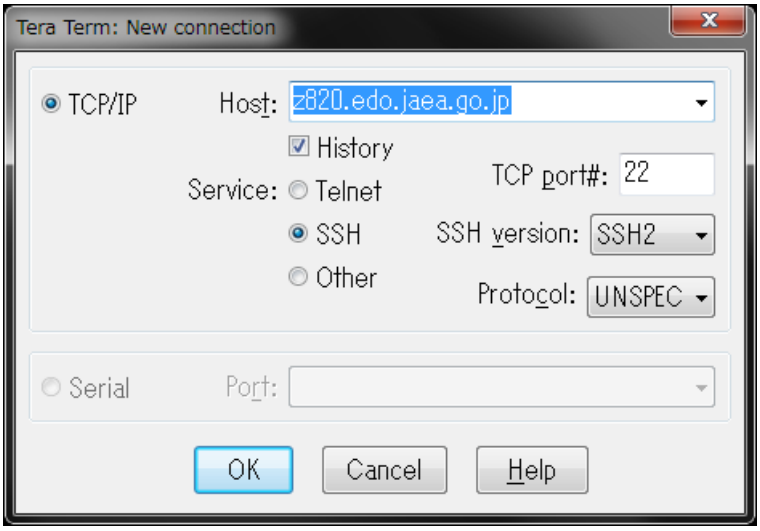


Figure 3　Tera Term dialog 1)

2) Select **Setup** > **SSH Transfer** from the menu bar. Click **Add…** in the **Forwarding Setup** dialog.

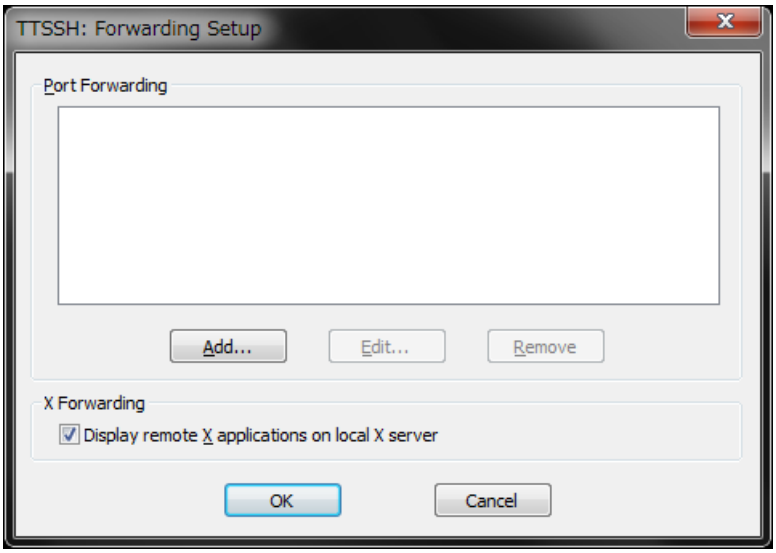


Figure 4　Tera Term dialog 2)

3) In the **Select Direction for Forwarded Port** dialog, select **Forward Local Port** and enter the port number to be used for PBVR Client. In the **to remote machine** text field, enter the domain name or the IP address of the server. In the **port** field, enter the port number to be used on PBVR Server. Click on **OK** to complete the setup of port forwarding.

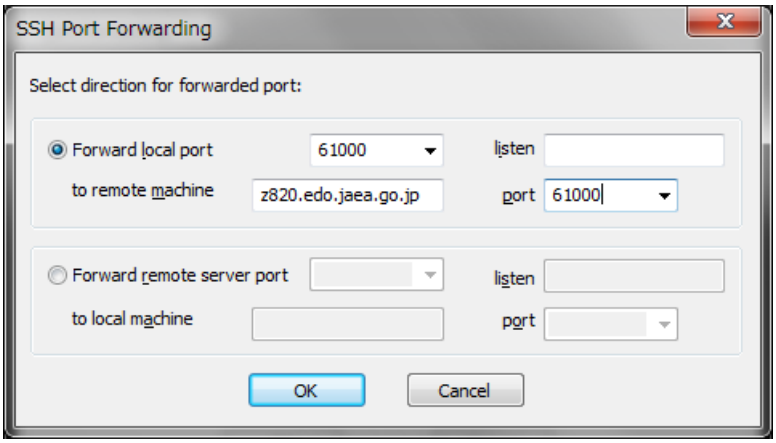


Figure 5　Tera Term dialog 3)

4) Connect to the server. Select **File** > **New Connection** from the menu bar. In the **New Connection** panel, enter the host name of the serve and click on **OK**. In the **SSH Authentication** panel, enter the user name and passphrase, or specify the location of the private key file, and click on **OK**.

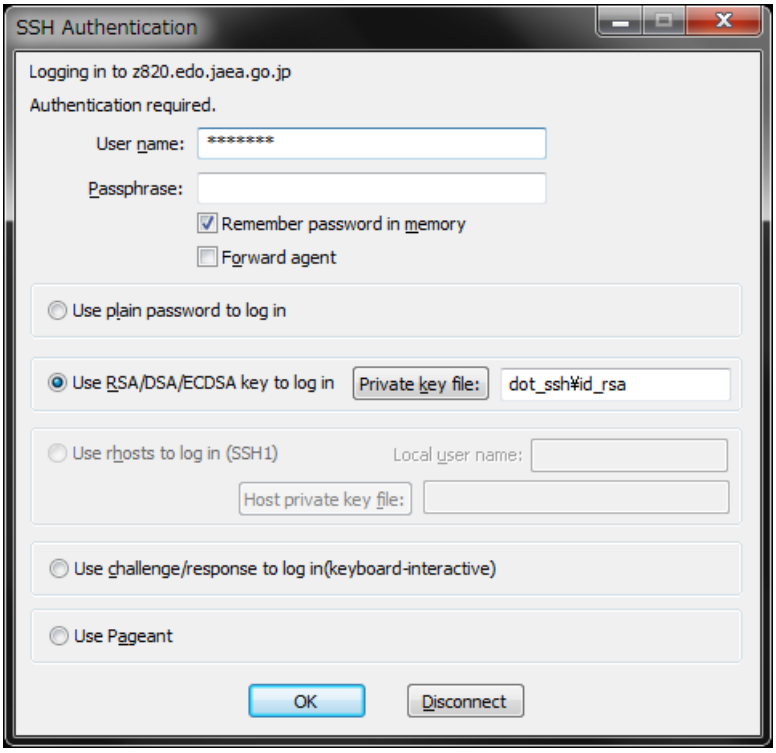


Figure 6　Tera Term dialog 4)

The following procedures show how to launch PBVR Server and Client after establishing port forwarding. This example uses the Visual Studio 2013 x64 Cross Tools command prompt in Visual Studio 2013 as the terminal for launching PBVR Client.

Step1 [Launch PBVR Server]
Server> mpiexec –n 4 pbvr_server –p port_number
Step2 [Set a client parameter for Windows]
Windows> set TIMER_EVENT_INTERVAL=1000
Step3 [Launch PBVR Client]
Windows> pbvr_client.exe –fin filename –p port_number

Note that PBVR Client on a Windows machine can be launched also by executing a batch file with the following lines.

```
set TIMER_EVENT_INTERVAL=1000
pbvr_client.exe –vin filename –p port_number
```

4.2.5 Launching Server for VTK data

In order to launch the server program for VTK data included in the load module package, it is necessary to set environment variables according to each environment.

LINUX

Set environment variables as follows.

```
% export LD_LIBRARY_PATH=${VTK_LIB_PATH}:$LD_LIBRARY_PATH
```

Mac

Set environment variables as follows.

```
% export DYLD_LIBRARY_PATH=${VTK_LIB_PATH}:$DYLD_LIBRARY_PATH
```

Windows

Control Panel -> System -> Advanced -> Environment Variables

Set environment variables as follows.

Variable	Value
Path	d:¥Program Files¥VTK 6.3.0¥bin

The Value specifies the bin directory under the installation directory of vtk.

4.3 How to run in staging environment

Describe how to execute in the staging environment in K computer. Regarding the designation of the file reader parameter file and the designation of the file transfer, it is necessary to establish consistency and start the server program. Depending on the output format of the server program, there are cases where output is performed from one process to another process. In such a case, it is necessary to specify a shared directory accessible from multiple processes as an output destination.

4.3.1 Execution Shell and Parameter File

```
#!/bin/bash -x
#
#PJM --rsc-list "elapse=01:00:00"
#PJM --rsc-list "node=64"
#PJM --rsc-list "rscgrp=small"
#PJM --stg-transfiles all
#PJM --mpi "proc=64"
#PJM --mpi "use-rankdir"
#PJM --stgin    "rank=* ./filter                %r:./"          ……①
#PJM --stgin    "rank=* ./param.txt             %r:./"                ……②
#PJM --stgin    "rank=0 /data/ucd/ucd*.dat      0:../"                ……③
#PJM --stgout "rank=* %r:../output*.dat          ./"                   ……④
#PJM --stgout "rank=* %r:./pbvr_filter.*         ./LOG/"         ……⑤
#PJM -S

. /work/system/Env_base

export PARALLEL=8
export OMP_NUM_THREADS=8

mpiexec -n 64 lpgparm -p 4MB -s 4MB -d 4MB -h 4MB -t 4MB filterpbvr_server param.txt
……⑥
```

① Transfer the execution module to rank directory of each process
② Transfer the parameter file to the rank directory of each process
③ Transfer input data to shared directory(The File specified in the parameter file)
④ Transfer output data from shared directory to local directory
⑤ Transfer log and error file from rank directory to local directory
⑥ Launch execution modules in the rank directory of each process, wtih rank of each process Parameter file in the directory as an argument

```
#
in_dir=../                    ……⑦
field_file=pd3d.fld
out_prefix=case0
out_dir=./                    ……⑧
file_type=0                   ……⑨
n_layer=3
start_step=0
end_step=511
```

⑦ Specify the path of input data.(In the staging environment, specification by relative path, in the above example, input data is read from the shared directory.)

⑧ Specify the path of output data.(In the staging environment, specification by relative path, in the above example, output to rank directory within each process.)

⑨ Specification of file output format.(in the above example,SPLIT format.)

4.3.2 I / O Files and Directorys

The relationship between the Input / Output files and directories of the server program in the staging environment is shown below. There is no problem when I / O destination is specified in shared directory.(Except log files and error files). Regarding output data, it is possible to output to the rank directory.

Table 11 Correspondence table of I / O files and directories on K computer

I/O	Species	Rank Directory	Shared Diractory
Input	Parameter file	○*1	○
	Input data	○*2	○
Output	Output data	○	○
	Log & Error file	○*3	×

*1 Only the rank 0 is input as the parameter file.

*2 When all input files can be transferred to the rank directory of all processes.

*3 Output destination is fixed to rank directory.

5 PBVR Client

5.1 Overview

PBVR Client can operate either in client-server mode or in stand-alone mode.

In client-server mode, PBVR Client receives particle data that is rendering primitives generated in PBVR Server. Further, PBVR Client renders the data using OpenGL. PBVR Client also gets visualization parameters (a transfer function etc.) via user interaction and sends the parameters to PBVR Server. In this way, PBVR Client controls the volume rendering process in PBVR Server. Data transfer between PBVR Client and PBVR Server uses a socket communication with a user-specified port number.

In contrast, when PBVR is in stand-alone mode, it reads and displays particle data generated by PBVR Server operating in batch mode.

5.2 Launching PBVR Client

The following examples show how to launch the client program in client-server mode and to do so in stand-alone mode. When PBVR Client starts, it opens three panels: **Viewer**, **Main panel**, and **Time Panel**.

Launch PBVR Client in client-server mode *1
$ pbvr_client -fin [file reader parameter file name *2] [command line options]

Launch PBVR Client in stand-alone mode
$ pbvr_client –pin 1 [particle data file name] [command line options]

*1. Client-server mode requires starting PBVR Server beforehand.
*2. The file reader parameter file can be specified with the absolute or the relative path to the .pfi file.
To specify two or more file reader parameter files, make the pfl file that lists the file reader parameter files, and specify the pfl file with the command line option '-fin'.
It is necessary to write "#PBVR FILTER FILES" to the head of the pfl file, from the second line, write the file reader parameter file as an absolute path or a relative path from the directory where the PFL file is located.

An example of description of PFL file is shown.

```
#PBVR FILTER FILES
hex.txt
hex2.txt
```

Table 12　List of command line option for client

Option	Launch mode *1	Parameter value	Default parameters	Function
-h	CS,SA	-	-	Display the list of options and parameters
-pd	CS	Real number	1.0	Particle density
-S	CS	u, m	u	Particle sampling method u: uniform sampling m:metropolis sampling
-plimit	CS	1~99999999	1000000	Particle limit *2
-tdata	CS	all, div	all	Particle data transfer method all: step batch transmission, div: sub-volume divide forwarding）
-pa	CS,SA	File name	-	Visualization parameter file
-fin	CS	File name	-	File reader parameter file *2
-tf	CS	File name	-	Name of the transfer function file *3
-p	CS	Port number	60000	Port number of socket communication
-viewer	CS,SA	100-9999 ×100-9999	620×620	Viewer resolution
-shading	CS,SA	{L/P/B}, ka, kd, ks, n	-	Shading method *4
-pout	CS,SA	File name	-	Output file name for particle data *5
-pin1	SA	File name	-	Input file name for particle data

-iout	CS,SA	Directory name	./	Output directory name for image files

*1. CS and SA denote client-server mode and stand-alone mode, respectively.

*2. If this option conflicts with the option in the Visualization parameter file specified with '-pa', the latter is ignored.

*3. Transfer function files are generated by hitting the **Export File** button in the **Transfer Function Editor**. In order to apply the transfer function specified in this option, hit the **Apply** button in **Transfer Function Editor**. Alternatively, the transfer function file can be loaded also with the **Import File** button.

*4. This argument specifies the shading parameters.

L: Lambert Shading
This method ignores specular reflection in the shading process.
Parameters 'ka' and 'kd' are the coefficient for ambient and diffusion, respectively.
They can have a value between 0-1.

P：Phong Shading
This method adds the specular reflection to Lambert shading. Phong shading imitates smooth metal and mirrors. (This is sometimes called highlight).
Parameter 'ka', 'kd','ks' (coefficients for specular reflection lying between 0-1) and 'n' (strength of highlight lying between 0-100) are used.

B：Blinn-Phong Shading
This is a shading model that simplifies Phong shading. Parameters 'ka', 'kd', 'ks', and 'n' exist.

*5. This generates a series of particle data files that are named
"[file name]_[time index]_[number of sub-volumes]_[sub-volume index].kvsml",
where the [file name] is the prefix specified this option.

5.3 Terminating PBVR

5.3.1 Standard Termination

PBVR Client's rendering process for the time-series data starts from the initial time step, and continues to the final time step. When the final time step is rendered, PBVR Client returns to the initial time step to loop over the steps. To terminate PBVR Client, press **Ctrl+C** in the console running PBVR.

In client-server mode, pressing **Ctrl+C** in the client console terminates both PBVR Client and PBVR Server. Just before the termination, PBVR Client and Server will synchronize their time step. However, PBVR Client ignores pressing **Ctrl+C** whenever the client-server communications are interrupted with the **Stop** button in **Time Panel**.

5.3.2 Forced Termination

When PBVR Server is terminated not by pressing **Ctrl+C** in PBVR Client's console, PBVR Client becomes stuck and cannot be terminated with **Ctrl+C**. Furthermore, even if **Ctrl+C** is pressed to terminate PBVR Client, both PBVR Client and Server might become stuck. This can happen if the time step is not updated due to heavy particle generation processes or some other reason. In such a case, obtain the process IDs of PBVR Client and PBVR Server using the ps command in the console, and then force them to quit with the kill command as follows.

[Force the termination of a PBVR Client process]

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

[Force the termination of a PBVR Server process】

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

5.4 Using PBVR Client GUI

5.4.1 Viewer

As shown in Figure 7, **Viewer** displays the rendering result of particle data.

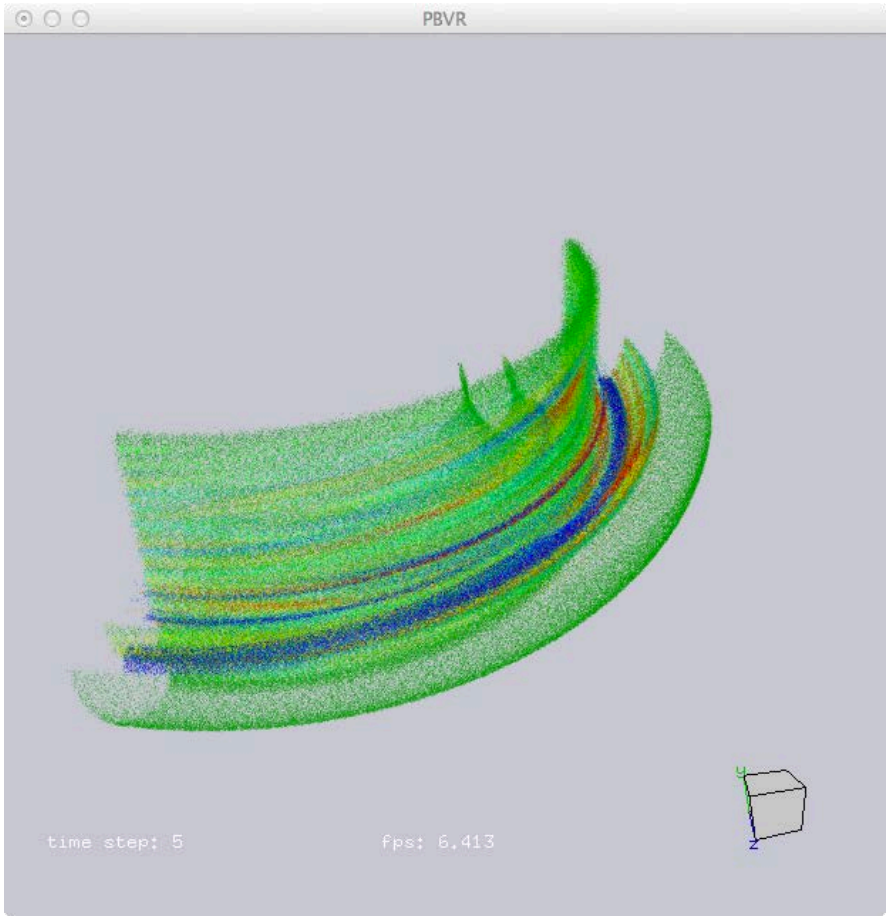


Figure 7 **Viewer**

[Operations]

Rotation: move the mouse while pressing the left-button

Translation: move the mouse while pressing the right-button

Zoom: scroll up/down the mouse wheel, or move the mouse up/down while pressing the **Ctrl** key

Reset: home button (fn + left arrow on Mac)

[Display]

time step: the time step of the displayed data

fps: the frame rate [frame/sec]

5.4.2 Main Panel

Figure 8 shows **Main panel** of PBVR Client. The items of the panel are described below.

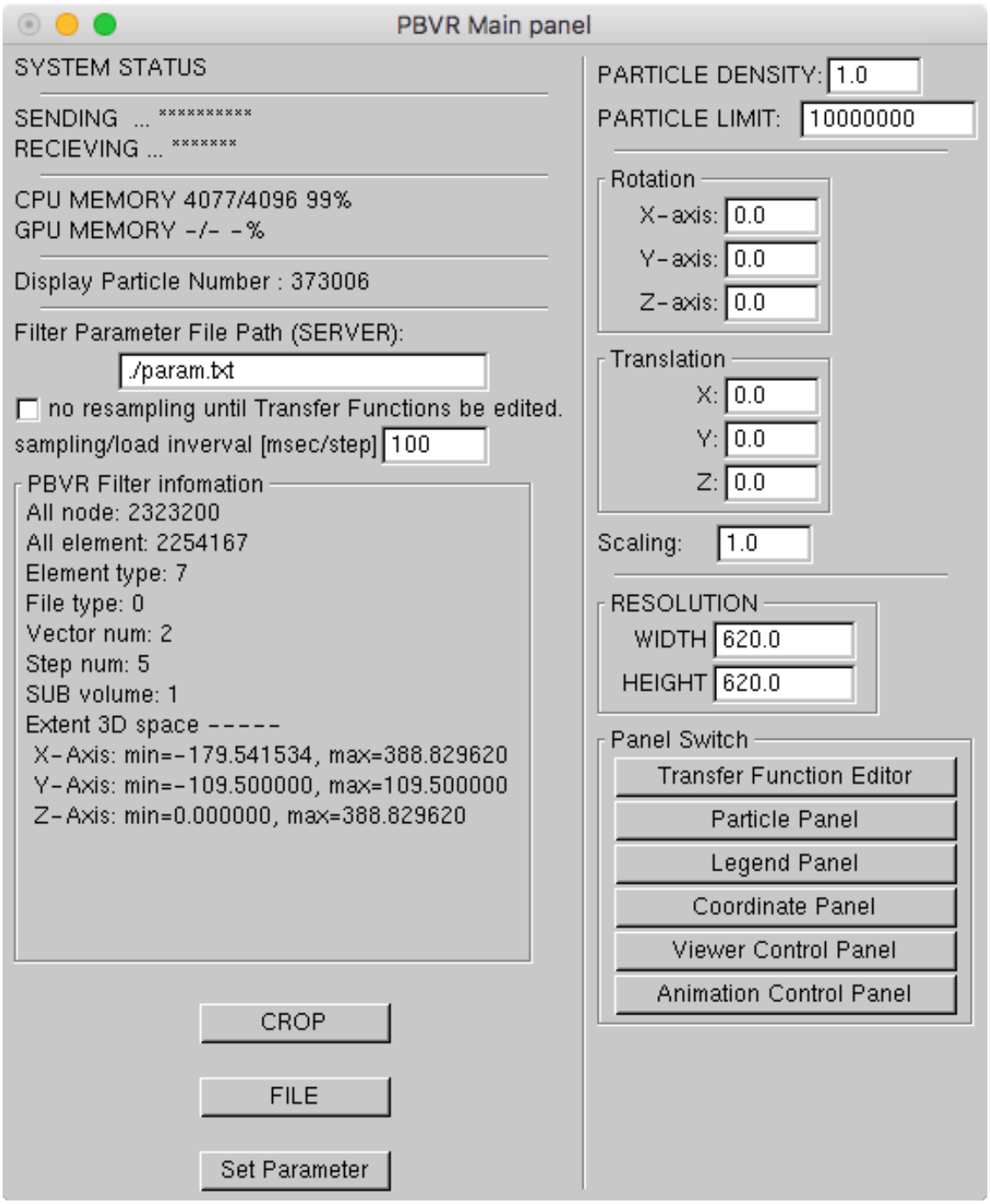


Figure 8 **Main panel**

- **PARTICLE DENSITY**

  Specifies the particle density related to the depth of the image.

- **PARTICLE LIMIT**

  Specifies the maximum number of particles that are generated in PBVR Server. Use this to avoid the explosive increase of the number of particles (e.g. due to the false settings of a transfer function). The number is multiplied by 10^6.

- **Rotation**

  Specifies the rotation angle for each x, y, and z axis by degree.
- **Translation**

  Specifies the translation for each x, y, and z direction.
- **Scaling**

  Specifies the scaling of the object.
- **RESOLUTION**

  Specifies the Viewer's resolution.
- **Transfer Function Editor**

  Displays Transfer Function Editor, which is described in the next section.
- **Particle Panel**

  Displays Particle Panel, which is described in the next section.
- **Legend Panel**

  Displays Legend Panel, which is described in the next section.
- **Coordinate Panel**

  Displays Coordinate Panel, which is described in the next section.
- **Viewer Control Panel**

  Displays Viewer Control Panel, which is described in the next section.
- **Animation Control Panel**

  Displays Animation Control Panel, which is described in the next section.
- **SENDING**

  Shows the progress of data transfer to the server program.
- **RECEIVING**

  Shows the progress of data transfer from the server program.
- **CPU MEMORY**

  Displays the system memory usage in megabytes.
- **GPU MEMORY**

  Displays the GPU memory usage in megabytes.
- **Display Particle Number**

  Display the number of the particle shown to a viewer is indicated.
- **Filter Parameter File path (SERVER)**

  Specifies file reader parameter file or PFL file name.
- **no-repeat sampling until Transfer Functions be edited**

  When the transfer function has not been changed in the client server mode, do repetition drawing of time series data by using particle data in client memory.
- **PBVR Filter information**

  Displays information about the volume data in PBVR Server.
- **FILE**

  This button shows the **FILE Panel**, whose detail is described later.

- **CROP**

  Displays **CROP Panel**, which is described in the next section.

- **Set parameter**

  Sends parameters specified in “Main panel” to the server program.

5.4.2.1 FILE Panel

FILE Panel is a panel for reading and writing visualization parameter files. This panel is shown when the **FILE** button is hit.

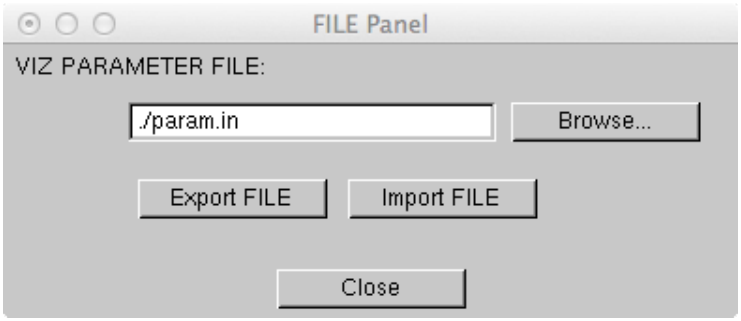


Figure 9 **FILE Panel**

- **VIZ PARAMETER FILE**

  specifies the path to a visualization parameter file.
- **Browse …**

  Opens a file dialog for specifying the path to a visualization parameter file.
- **Export FILE**

  Saves the current parameter settings to a visualization parameter file.
- **Import FILE**

  Imports a visualization parameter file.
- **Close**

  Closes **FILE Panel**.

5.4.2.2 CROP Panel

CROP Panel is activated by hitting the **CROP** button in **Main panel**. Use **CROP panel** for operations related to extracting and rendering elements involved within the Region Of Interest (ROI). ROI can be specified with a cuboid, a sphere, or a cylinder.

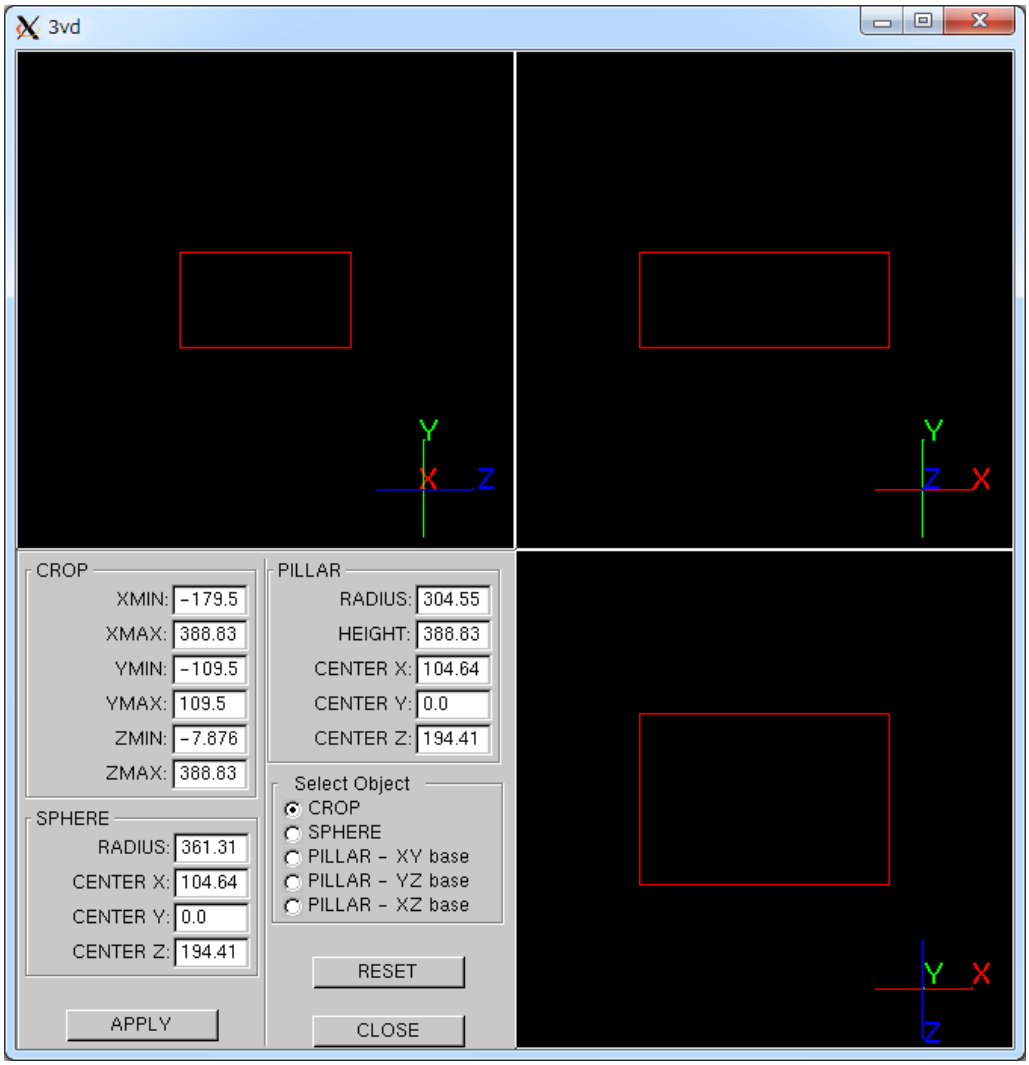


Figure 10 **CROP panel**

[Operations]

Move region: move the mouse on trihedral panel

Zoom region: move the mouse on trihedral panel while pressing Ctrl

Select Object: Specifies the shape of the ROI

CROP: A cuboid

SPHERE: A sphere

PILLAR-XY base: A cylinder with a *X*-*Y* base

PILLAR-YZ base: A cylinder with a *Y*-*Z* base

PILLAR-XZ base: A cylinder with a *X-Z* base
CROP: Specifies the range of the cuboid
SPHERE: Specifies the center and radius of the sphere
PILLER: Specifies the radius, the height, and the center coordinate values of the cylinder
RESET: Resets the **CROP** panel
APPLY: Extracts the ROI
CLOSE: Closes the panel

Displaying **CROP** panel overdraws the shape of the ROI in **Viewer** as in the following figure.

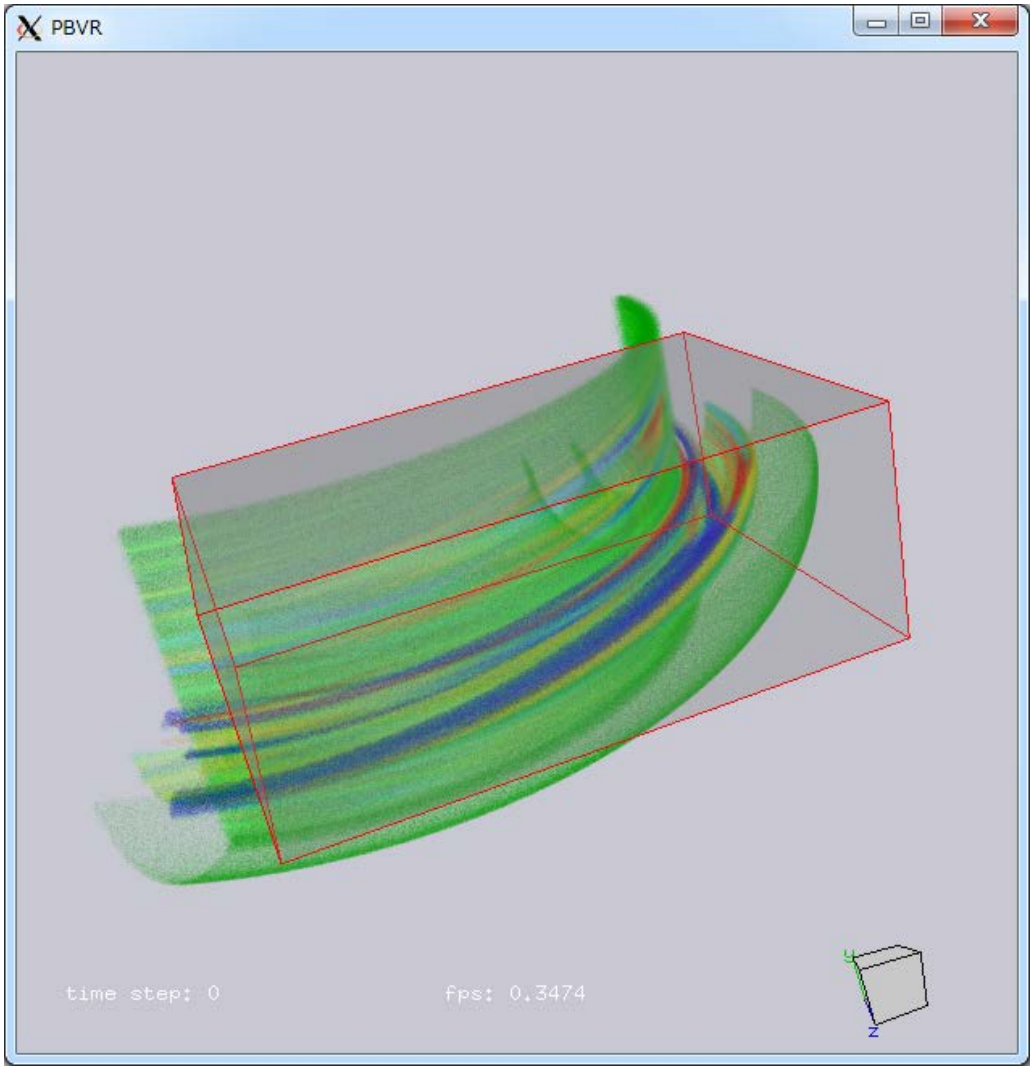


Figure 11 **Viewer panel** interacting with CROP

5.4.3 Transfer Function Editor

Transfer Function Editor edits the transfer functions, which assigns a color/opacity to each scalar value for volume rendering. **Transfer Function Editor** is activated by hitting the Transfer Function Editor button in Main panel. In a standard volume rendering, a transfer function is defined by only one physical quantity. In contrast, PBVR provides a new multi-dimensional transfer function design, which has the following three features:

1) Assign two independent variable quantities to color and opacity.
2) Define each variable quantity with an arbitrary function of the *X-Y-Z* coordinates and variables *q1*, *q2*, *q3*…
3) Synthesize a multidimensional transfer function from one-dimensional transfer functions *t1-t5* using equations.

This new transfer function design adds significant flexibility to visualization. **Transfer Function Editor** is shown Figure 12. Each item in the panel is explained below.

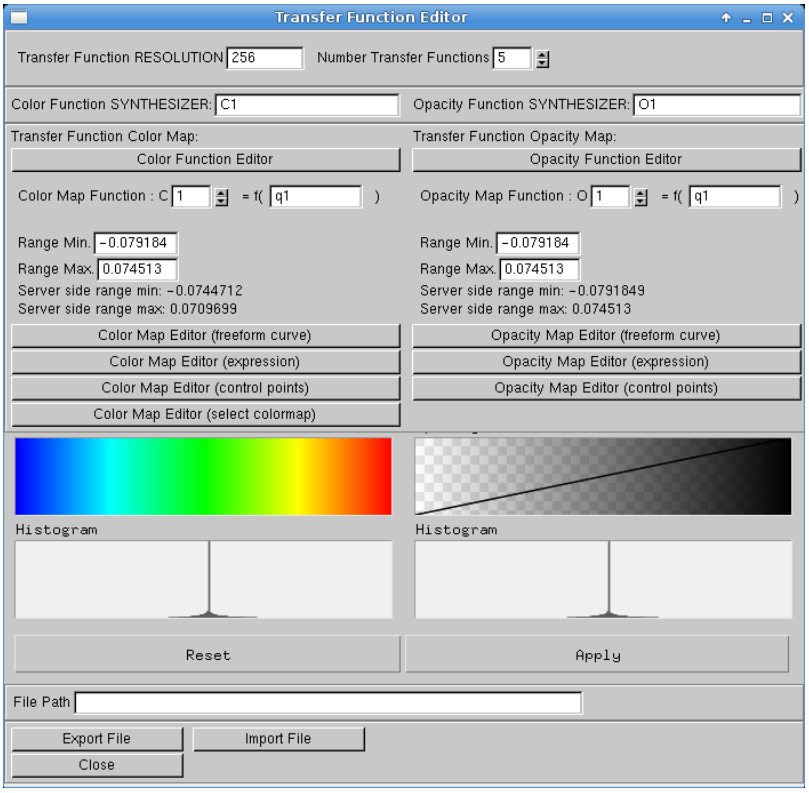


Figure 12 **Transfer Function Editor**

[Operations]

Scale change in histogram: Drag the mouse up/down on **Histogram**

- **Transfer Function RESOLUTION**

  Species the resolution of the transfer function

- **Number Transfer Functions**

  Specify the limit number of transfer functions that can be created.
- **Colar Function SYNTHESIZER**

  Specify the composition formula of transfer color functions created under the names C1 to C [N] *1
- **Opacity Function SYNTHESIZER**

  Specify the composition formula of transfer opacity functions created under the names O1 to O [N] *1
- **Reset**

  Resets the panel.
- **Apply**

  Sends a transfer function defined with this panel to the server.
- **File Path**

  Specifies a file path for saving and loading a transfer function file.
- **Export File**

  Saves a transfer function defined with this panel to a file in the same format as the parameter file specified with the command line option '-pa'.
- **Import File**

  Loads a transfer function stored in a file to this panel
- **Close**

  Close Transfer Function Editor.

*1 [N] is the value of the limit number of the transfer function specified by Number Transfer Fuctions.

5.4.3.1 Color Map Editor Panel

[**Transfer Function Color Map** category]

Create and display transfer function which corresponding to transfer color function name described in transfer function synthesis formula of Color Fuction SYNTHESIZER

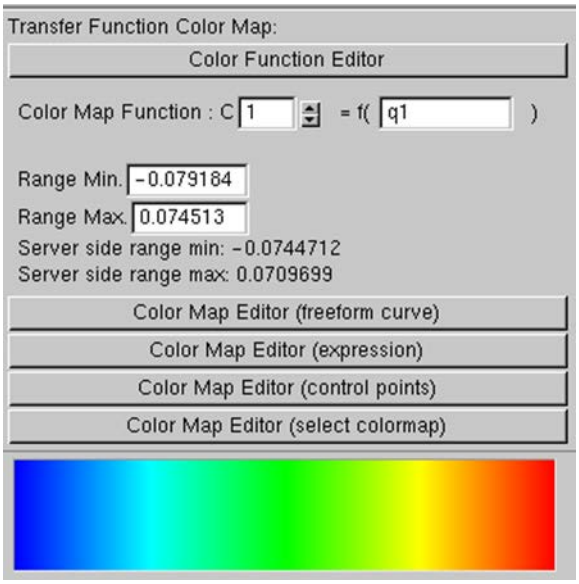


Figure 13 Transfer Fuction Color Map Category

- **Color Function Editor**

  Display the Color Function Editor screen and create and select the transfer color function of C1 to C [N]. *1

- **Color Map Function**

  Create and display the transfer color function by the name of C1 to C [N]. *1

  Defines the (synthesized) variable quantity used for color of the selected transfer function. An equation can be entered, while the following variables are available.

  Physical quantities：q1, q2, q3, .., qn.

  Coordinate values：X, Y, Z.

- **Range Min**

  Specifies the minimum value of the specified variable quantity.

- **Range Max**

  Specifies the maximum value of the specified variable quantity.

- **Server side range min**

  Displays the minimum value of the (synthesized) variable quantity obtained in the server program.

- **Server side range max**

  Displays the maximum value of the (synthesized) variable quantity obtained in the server program.

- **Color Map Editor (freeform curve)**

  Displays a sub-panel, which specifies a transfer function with a freeform curve. Use the mouse to edit the freeform curve.

- **Color Map Editor (expression)Button**

Displays a panel, which creates transfer functions of variables and colors corresponding to selected transfer function names by numerical formula description.

- **Color Map Editor (control points)Button**

  Displays a panel, which creates transfer functions of variables and colors corresponding to selected transfer function names by Control point designation.

- **Color Map Editor (selsect colormap)Button**

  Displays a panel, which creates variables and color transfer functions corresponding to the selected transfer function name by selecting them from color bars prepared in advance.

- **Color**

  The color bar of the variable and color transfer function created by this editor is displayed.

*1 [N] is the value of the limit number of the transfer function specified by Number Transfer Fuctions.

5.4.3.1.4. Color Function Editor

With the Color Function Editor button, display the Color Function Editor screen and create and select the transfer color function of C1 to C [N].
[N] is the value of the limit number of the transfer function specified by Number Transfer Fuctions.

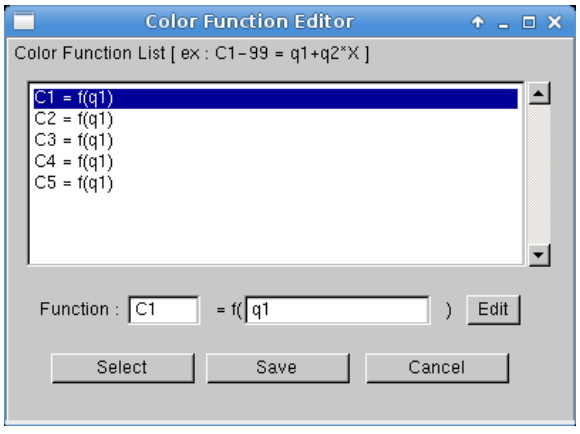


Figure 14　Color Function Editor screen

- **Color Function List**

  Display the created transfer function.

- **Function**

  Enter transfer function. (C [N] = f (variable)

- **Edit Button**

  Reflect on Color Function List.

- **Save Button**

  Apply the transfer function of Color Function List

- **Select Button**

  Apply transfer function of Color Function List and Select transfer function of list selection.

5.4.3.1.4. Color Function Editor

Use a color map editor (freedom curve) button to display a panel that creates a transfer function of variable and color corresponding to the selected transfer function name with free curved line input by mouse.

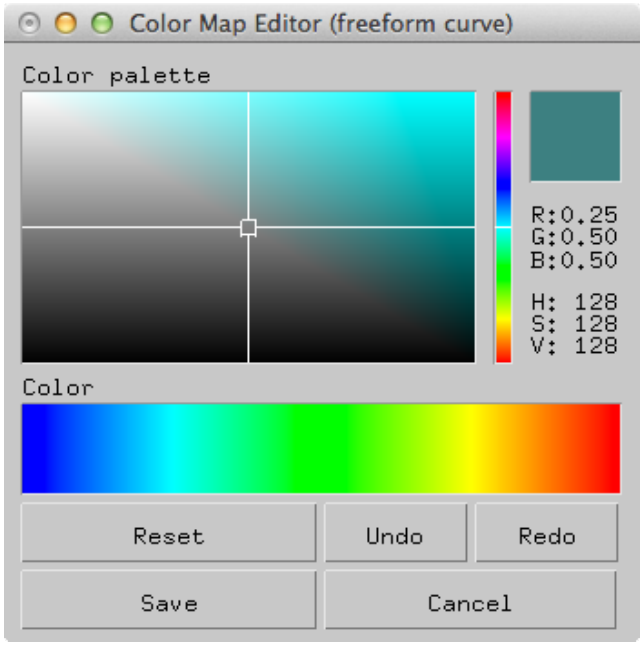


Figure 15 **Color Map Editor (freeform curve)** panel

- **Color palette**

  Specifies the saturation, the brightness, and the hue of a color with mouse cursor. On the left, the horizontal and vertical axes correspond to the saturation and brightness, respectively. The neighboring bar shows the hue.

- **RGB**

  Specifies the hue of the color by placing a mouse cursor. The upper-right box displays the color created by **Color palette** and **RGB bar**.

- **Color**

  Blends the colors in **Color** area with a color specified with **Color palette** and **RGB bar.** To specify the locations in **Color** area, trace the locations by dragging the mouse cursor while pressing the left mouse button. The blending ratio of the original

color and the overpainting color is determined by the mouse cursor's vertical position. For example, when the upper edge of the color bar is traced from left to right, the **Color** bar is painted completely by the specified color rather than by blended colors; when the vertical center line of **Color** bar is traced, the colors are replaced with blended colors with 50% of the original color and 50% of the specified color.

- **Reset**

  Resets the panel.

- **Undo**

  Undoes the last mouse action.

- **Redo**

  Redoes the last mouse action undone.

- **Save**

  Saves the transfer function.

- **Cancel**

  Closes the panel.

5.4.3.1.4. Color Map Editor (expression)

Use a color map editor (expression) button to display a panel to create a transfer function by taking equations as input.

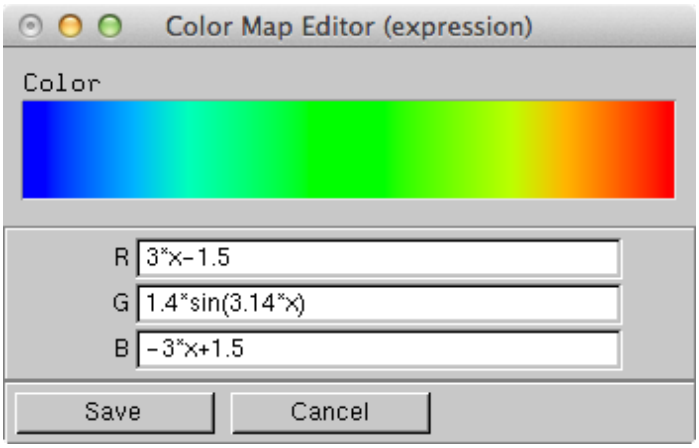


Figure 16 **Color Map Editor (expression) panel**

- **Color**

  Displays a color bar of a transfer function created in this panel.

- **R**

  Describes a transfer function of the R component of the color.

- **G**

  Describes a transfer function of the G component of the color.

- **B**

Describes a transfer function of the B component of the color.

- **Save** button

  Saves a transfer function created in this panel.

- **Cancel**

  Closes the panel.

5.4.3.1.4. Color Map Editor (control points)

Use a color map editor (control points) button to display a panel for creating a transfer function. This editor takes control points as input.

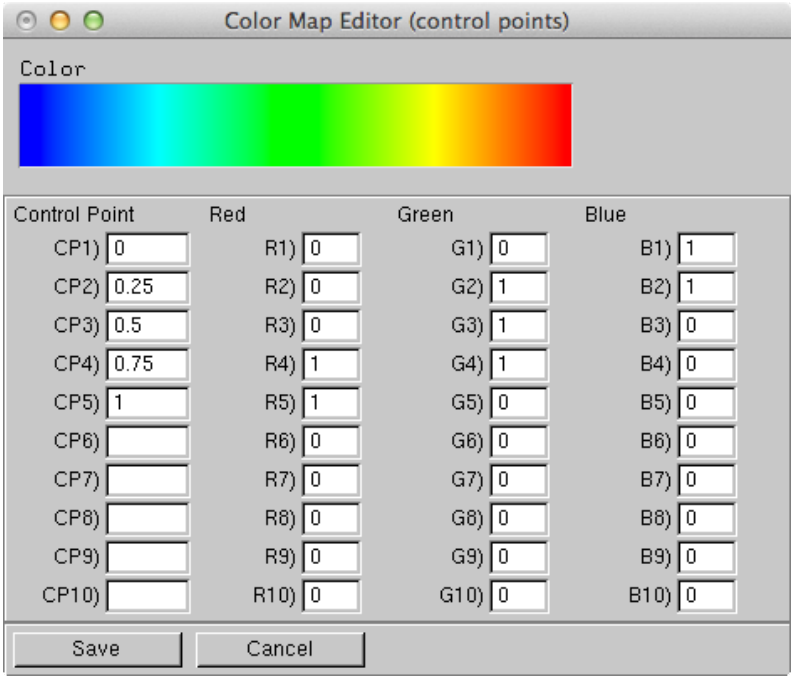


Figure 17 **Color Map Editor (control points) panel**

- **Color**

  Displays a color bar for the transfer function that is being defined with this panel.

- **Control Point**

  Specifies the values of (up to 10) control points with the fields **CP1)-10)** .

- **Red**

  Specifies the R component of the color at the control points.

- **Green**

  Specifies the G component of the color at the control points.

- **Blue**

  Specifies the B component of the color at the control points.

- **Save**

  Saves the transfer function.

- **Cancel**

Closes the panel.

5.4.3.1.4. Color Map Editor (select colormap)

Use a color map editor (select colormap) button to display a panel to create a transfer function from preset color bar templates.

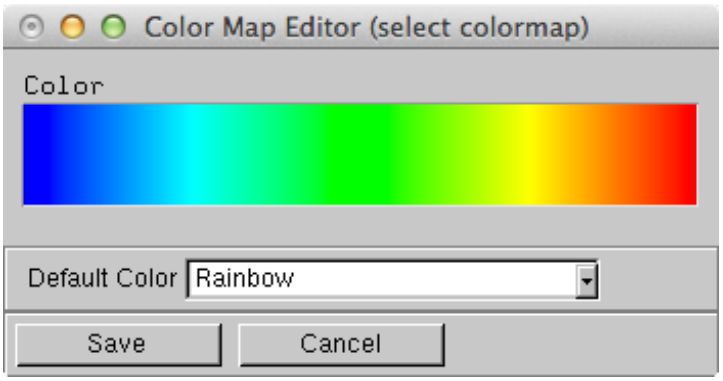


Figure 18 **Color Map Editor (select colormap) panel**

- **Color**

  Displays the color bar of the transfer function that is being created with this panel.

- **Default Color**

  Selects a color bar to be set as the transfer function. The following templates are available.

  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

- **Save**

  Saves the transfer function created with this panel.

- **Cancel**

  Closes the panel.

5.4.3.2 Opacity Editor

[**Transfer Function Opacity Map Category**】

The GUI components in this category set a variable quantity and color for the transfer function specified with the **Transfer Function Name** field.

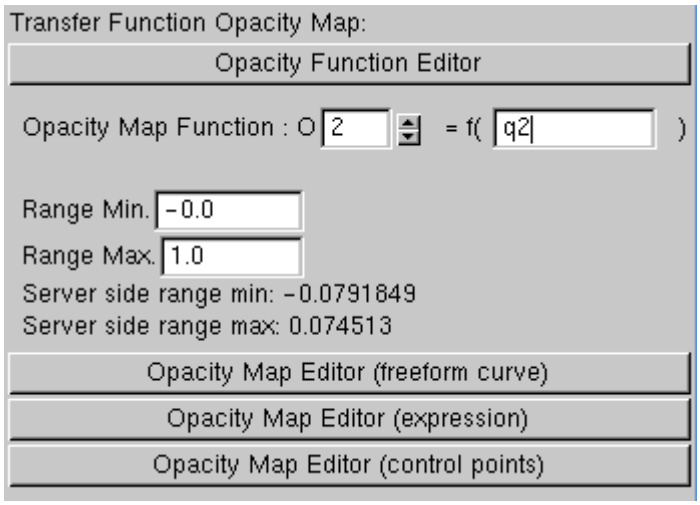


Figure 19 Transfer Fuction Opacity Map Category

- **Opacity Function Editor Button**

  Displays the Opacity Function Editor screen, and creates and selects the transmission opacity function of O1 to O [N]. *1

- **Opacity Map Function**

  Defines the (synthesized) variable quantity used for the opacity of the selected transfer function. An equation can be entered, while the following variables are available.

  Physical quantities: q1, q2, q3, .., qn.

  Coordinate values: X, Y, Z.

- **Range Min**

  Specifies the minimum value of the variable quantity.

- **Range Max**

  Specifies the maximum value of the variable quantity.

- **Server side range min**

  Displays the minimum value of the (synthesized) variable quantity obtained in the server program.

- **Server side range max**

  Displays the maximum value of the (synthesized) variable quantity obtained in the server program.

- **Opacity Map Editor (freeform curve) Button**

  Displays a panel for creating a transfer function with a freeform curve. Use the mouse to edit the freeform curve.

- **Opacity Map Editor (expression) Button**

  Displays a panel for creating a transfer function of variable and opacity corresponding to the selected transfer function name by mathematical formula description.

- **Opacity Map Editor (control point) Button**

  Displays a panel for creating a transfer function of the variable and opacity corresponding to the selected transfer function name with control point designation.

- **Opacity**

  Displays transfer function curve of variable and opacity created with this editor.

*1 [N] is the value of the limit number of the transfer function specified by Number Transfer Fuctions.

5.4.3.2.4. Opacity Function Editor

Use a Opacity Function Editor button to display a panel to create and select the transmission opacity function of O1 to O [N]..

[N] is the value of the limit number of the transfer function specified by Number Transfer Fuctions.

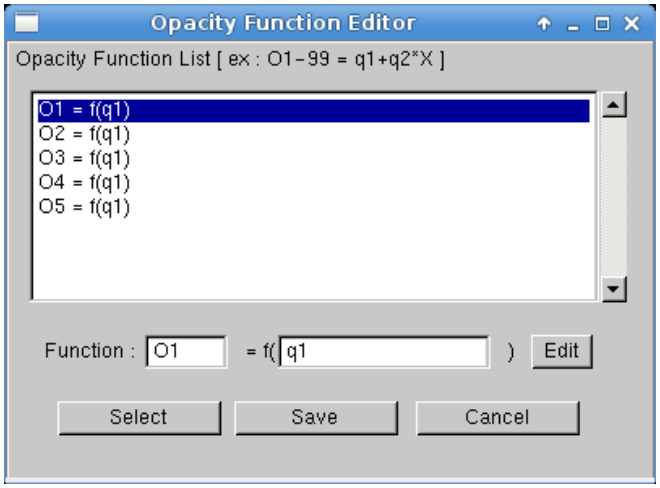


Figure 20　Opacity Function Editor

- Opacity Function List

  Displays the created transfer function.

- Function

  Enter the transfer function. (O [N] = f (variable)

- Edit Button

  Reflects on Opacity Function List.

- Save Button

Applys the transfer function of Opacity Function List.

- Select Button

  Applys transfer function of Opacity Function List, selects transfer function of list selection.

5.4.3.2.4. Opacity Map Editor (freedom curve)

Use a Opacity Map Editor button to display a panel to create a transfer function of variable and opacity corresponding to the selected transfer function name with free-form curve input by mouse operation

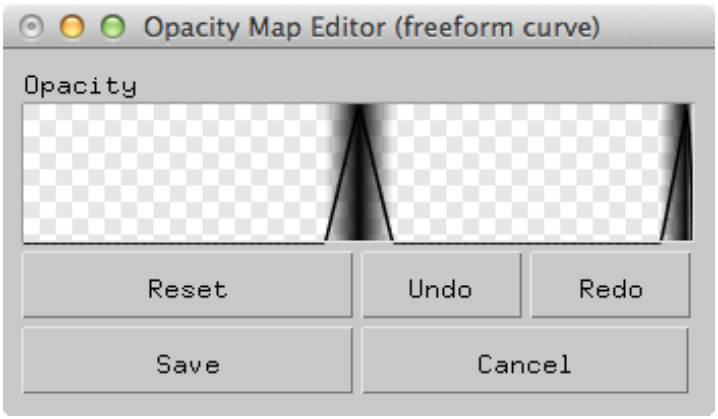


Figure 21 **Opacity Map Editor (freeform curve) panel**

- **Opacity**

  Specifies a transfer function for the opacity. A freeform curve is drawn by dragging the mouse while holding the left mouse button. A piecewise linear curve is drawn by specifying control points with right clicks.
- **Reset**

  Resets the panel.
- **Undo**

  Undoes the last mouse action.
- **Redo**

  Redoes the last mouse action undone.
- **Save**

  Saves the transfer function created with this panel.
- **Cancel**

  Closes the panel.

5.4.3.2.4. Color Map Editor (expression)

Use a Color Map Editor (expression) button to display a panel to display a panel to create a transfer function using equations.

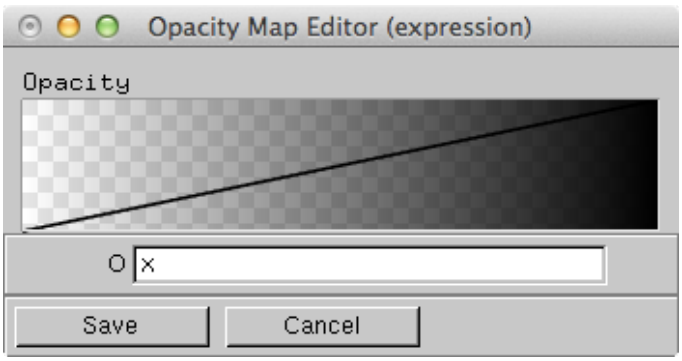


Figure 22 **Opacity Map Editor (expression) panel**

- **Opacity**

  Displays the transfer function for opacity specified by the equation in the field **O**.
- **O**

  Specifies the equation for the curve that specifies the transfer function of opacity.
- **Save**

  Saves the transfer function created with this panel.
- **Cancel**

  Closes the panel.

5.4.3.2.4. Color Map Editor (control points)

Use a Color Map Editor (control points) button to display a panel to display a panel to create a transfer function by taking equations as input.

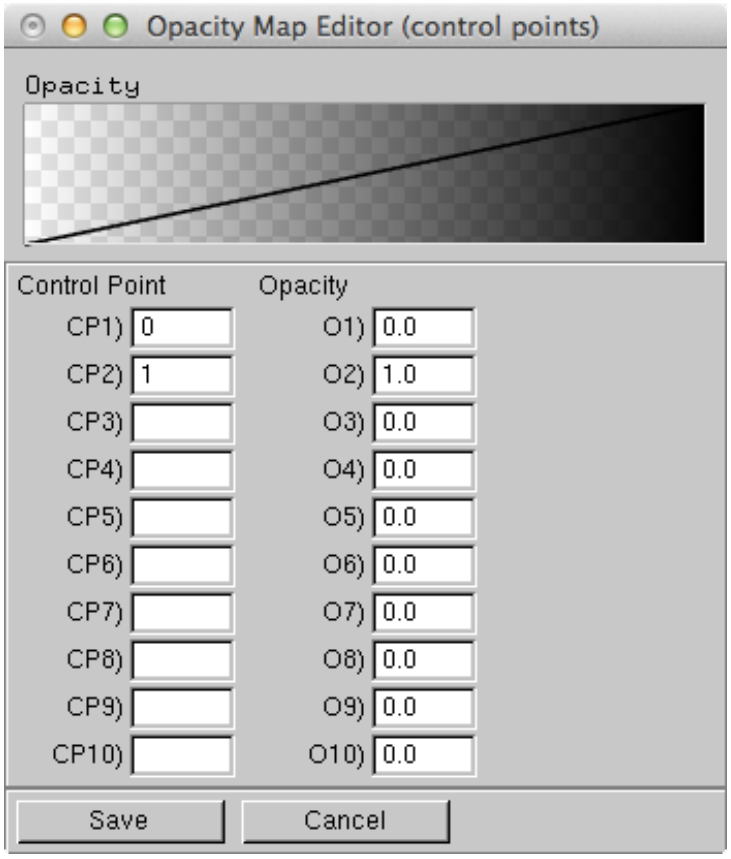


Figure 23 **Opacity Map Editor (control points) panel**

- **Opacity**（on the top）

  Displays the transfer function for the opacities specified in this panel.
- **Control Point**

  Specifies the values of (up to 10) control points in the fields **CP1)-10)** .
- **Opacity**（on the bottom right）

  Specifies the opacities at the control points.
- **Save**

  Saves the transfer function created with this panel.
- **Cancel**

  Closes the panel.

5.4.3.3 Function Editor

Table 13 lists the built-in math operations available in the function editor. They can be used to synthesize transfer functions and variable quantities, and to define colormap/opacity curves.

Table 13 Math operations in function editors

Math operation	**In function editors**
+	+
-	-
×	*
/	/
sin	sin(x)
cos	cos(x)
tan	tan(x)
log	log(x)
exp	exp(x)
square root	sqrt(x)
power	x^y

When NaN appears by the arithmetic processing of the function editor, PBVR outputs the error message and stops the drawing process.

5.4.4 Time panel

Figure 24 shows **Time panel**, which specifies the time steps for visualization. Each widget works as described in the followings.

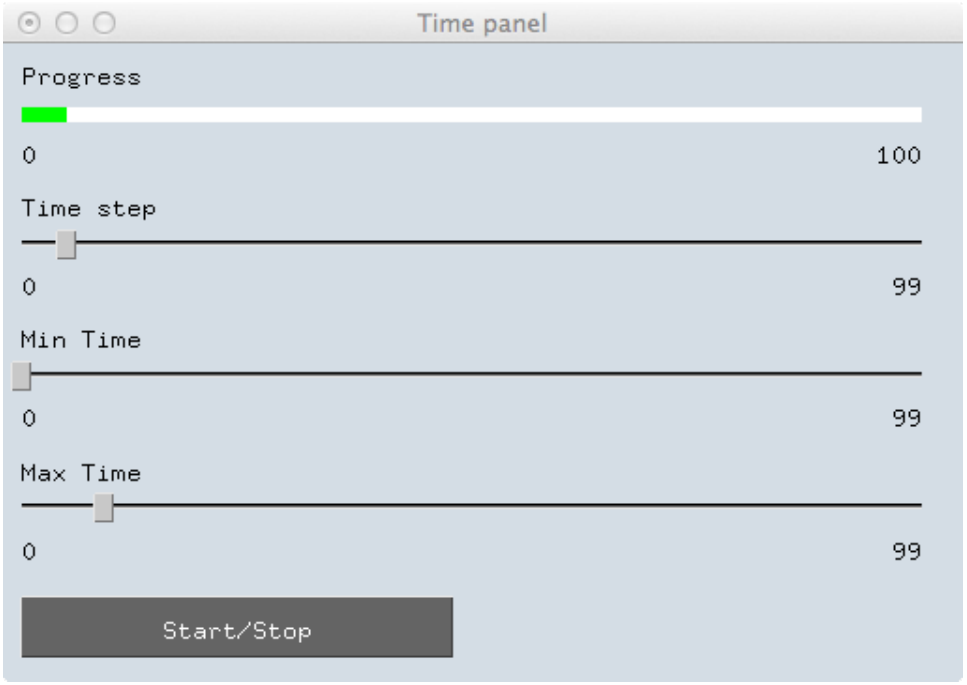


Figure 24 **Time panel**

- **Progress**

  Expresses the current time step as percentage.
- **Time step**

  Specifies the time step of the data to be rendered.
- **Min Time**

  Specifies the minimum time step for ROI.
- **Max Time**

  Specifies the maximum time step for ROI.
- **Start/Stop**

  Starts/stops the communication between PBVR Client and PBVR Server.

5.4.5 Particle panel

Figure 25 shows **Particle panel**, which integrates multiple particle datasets. Particle panel is activated by hitting the Particle panel button in Main panel. Each widget works as described in the followings.

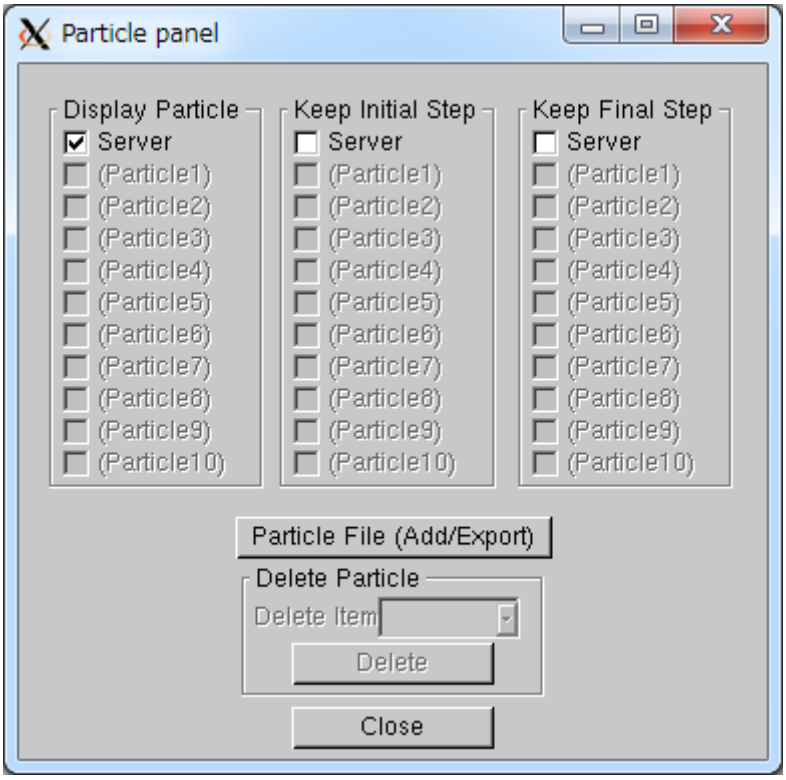


Figure 25 **Particle panel**

- **Display Particle**

  Shows a list of particle datasets, which are sent from PBVR Server, or are loaded from local files (maximum 10 files).

  1)Server check box

  Activated when a particle dataset from PBVR Server is integrated with local particle data sets. This checkbox is not available in stand-alone mode.

  2)(Particle1)-(Particle10) check box

  Activated when particle datasets loaded from local files are integrated. The checkbox is not available before particle datasets are loaded via Particle file panel or command line options –pin1, -pin2, …, -pin10.

- **Keep Initial Step**

  Specifies particle datasets, in which the initial step data is displayed before the time series starts, when integrated particle datasets start from different time steps.

- **Keep Final Step**

- **Keep Final Step**

  Specifies particle datasets, in which the final step data is displayed after the time series ends, when integrated particle datasets end at different time steps.
- **Particle File (Add/Export)**

  Opens Particle File sub-panel.
- **Delete Particle**

  Specifies a particle dataset to be deleted from a list in Display Particle.
- **Delete**

  Delete a particle dataset.
- **Close**

  Close Particle panel .

5.4.5.1 Particle File sub-panel

Particle File panel is a panel for reading and writing particle data files. This panel is shown when the **Particle File** button is hit.

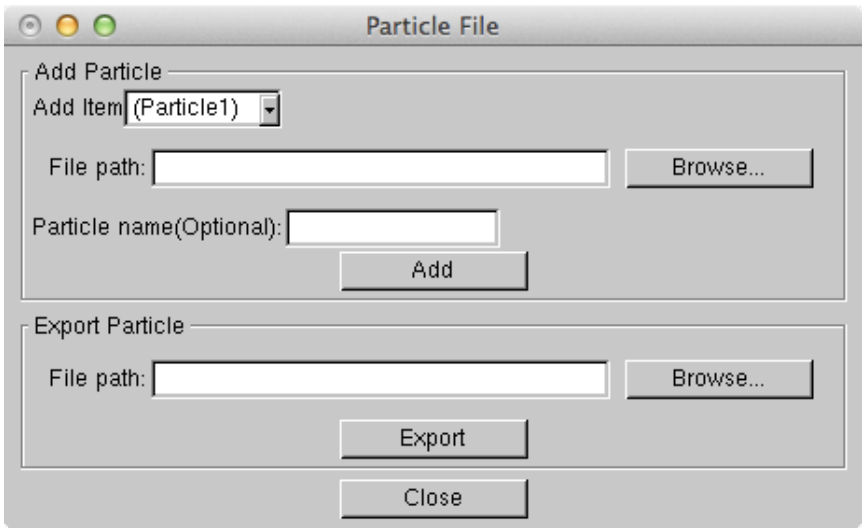


Figure 26 **Particle File panel**

- **Close**

  Close Particle File panel.

【Add Particle categoly】

- **Add Item**

  Specifies a particle data slot, in which a new particle dataset is loaded. When a old particle dataset exists for the particle data slot, the slot is overwritten by a new particle dataset.
- **File path**

  Specify a particle data file.

- **Browse**

  Opens a file dialog for specifying the path to a visualization parameter file.
- **Particle name (Optional)**

  Specifies the name of a particle dataset shown in **Particle panel**.
- **Add**

  Add a particle dataset to **Particle panel**.

【Export Particle Categoly】

- **File path**

  Specify a particle data file.
- **Browse**

  Opens a file dialog for specifying the path to a visualization parameter file.
- **Export**

  Output integrated particle data.

5.4.6 Image file production

PBVR Client saves image data on Viewer in the following two modes, and plays it as a movie. Image file production is activated by hitting the Animation Control Panel button in Main panel.

- Time series data mode

Saves images of time series data as a series of image data files with the BMP format. The image data files are converted or compressed as a movie file via free softwares such as ImageMagic and ffmpeg

- Key frame animation mode

Keeps geometry information of viewer at an arbitrary point as a key frame, and plays a series of key frames as a key frame animation.

Figure 27 shows **Animation Control Panel**. Each widget works as described in the followings.

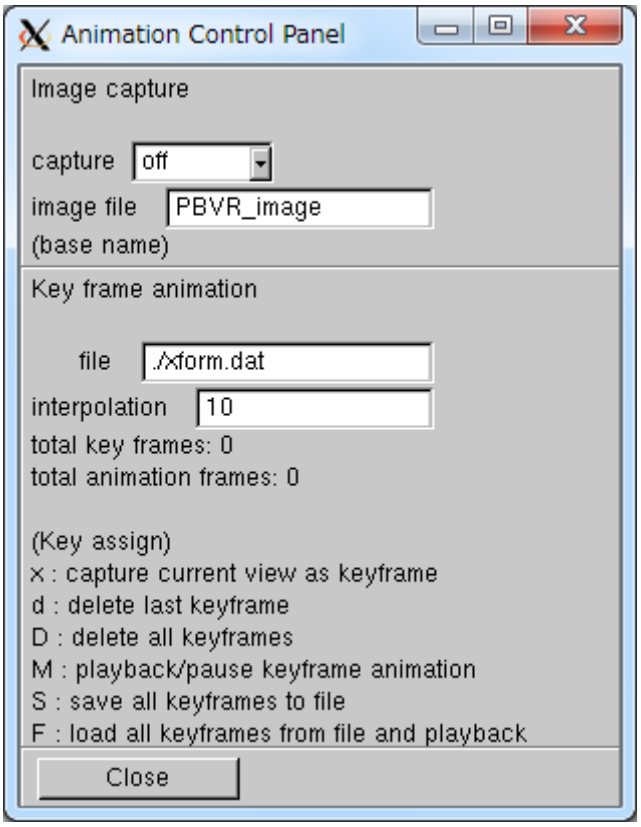


Figure 27 Animation Control Panel

- **capture**

  Controls on/off of image production.

- **image file**

  Specifies a prefix of image data files. The default name is PBVR_image.

- **file**

  Specifies a key frame file, which contains a series of geometry data. The default name is ./xform.dat.

- **interpolation**

  Specifies the number of frames used for linear interpolation of geometry data between two key frames in a key frame animation. The default value is 10.

- **total key frames**

  Shows the number of key frames stored in the current key frame animation. The value is initialized to 0, and incremented (or decremented) by pressing “x” (or “d”). The value is initialized to 0 by pressing “D”.

- **total animation frames**

  Shows the number of total frames stored in the current key frame animation, which is calculated as

  (total key frames – 1) x interpolation

- **Close**

  Close Animation Control Panel .

5.4.6.1 Image production

Image files are produced as follows.

① Specify prefix of image files in **image file**.
② Select “on” in the **capture** drop down menu.
③ A series of image files are saved at each time step.
④ Image production is stopped by selecting “off” in the **capture** drop down menu.

The image files are saved in the directory specified by the command line option ‘-iout’. When ‘-iout’ option is not specified, they are saved in the current directory ‘./’. The following shows an example of image data produced with the default prefix “PBVR_image”.

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

When the image files are produced from a key frame animation, which is explained later, the file names are modified by adding “_k” after the prefix.

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

5.4.6.2 Key frame animation of a still image

A key frame animation of a still image, which is obtained by pressing **Stop** in **Time Panel**, is produced as follows.

【Capture key frames and save them in a file】
① Specify a key frame file in **file**.
② Activate Viewer by clicking it.
③ Adjust view and press ‘x’ to store the geometry information of view on a memory.
④ Repeat ③.
⑤ Press ‘M (Shift+m)’ to play the key frame animation.
⑥ If the contents of the key frame animation is OK, press ‘S (Shift+s)’ to save a series of geometry information in the key frame file.

【Play a key frame file】
① Specify a key frame file in **file**.

② Activate Viewer by clicking it.
③ Press ‘F (Shift+f)’ to play a key frame animation stored in the key frame file.
④ Press ‘x’ to add new key frames to the current key frame animation.

Table 14 Keys used for controlling key frame animation

Key	Function
x	Add geometry information of the current Viewer to key frame data on a memory
d	Delete the last key frame
D	Delete all key frames
M	Play and pause key frame data on a memory
S	Save key frame data on a memory to a key frame file
F	Load a key frame file and play its key frame data

5.4.6.3 Key frame animation of time series data

A key frame animation of time series data is produced as follows.

① By pressing ‘x’ while time series data is rendered, both geometry information and a time step number are stored in a memory.

② Press ‘S’ to save a series of geometry information and time step numbers in the key frame file.

③ Press ‘F’ to load a series of geometry information and time step numbers in the key frame file and play a key frame animation. Here, If one sets key frames at unequal intervals, interpolation frames, which are specified in **interpolation**, are assigned non-uniform in time.

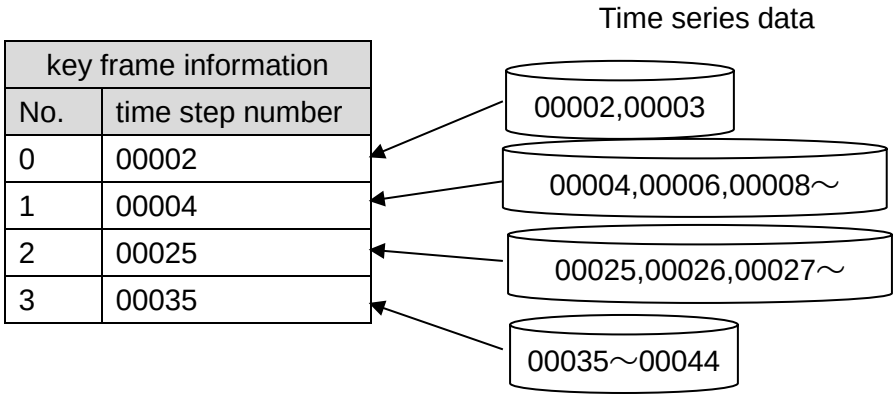


Figure 28　Key frame animation for time series data

In an example in Figure 28, if one uses 10 interpolation frames between key frames, 5 interpolation frames are assigned to the time steps 00002 and 00003 in between key frames No.0 and 1. On the other hand, in between No.1 and 2, 10 interpolation frames are assigned to the time steps from 00004 to 00024. As a results, the time steps, 00004, 00006, …00024 are shown in the key frame animation.

5.4.6.4 Key frame file format

A key frame file contains binary data with the following format. The file format is shown below.

File format

Key frame data 1
Key frame data 2
:

type	size (byte)	data
int	4	time step number
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

Figure 29 Key frame file format

5.4.7 Legend panel

Figure 30 shows Legend panel, which displays a bar relation between physical quantity and color is shown. Legend panel is activated by hitting the Legend Panel button in Main panel. Each widget works as described in the followings.

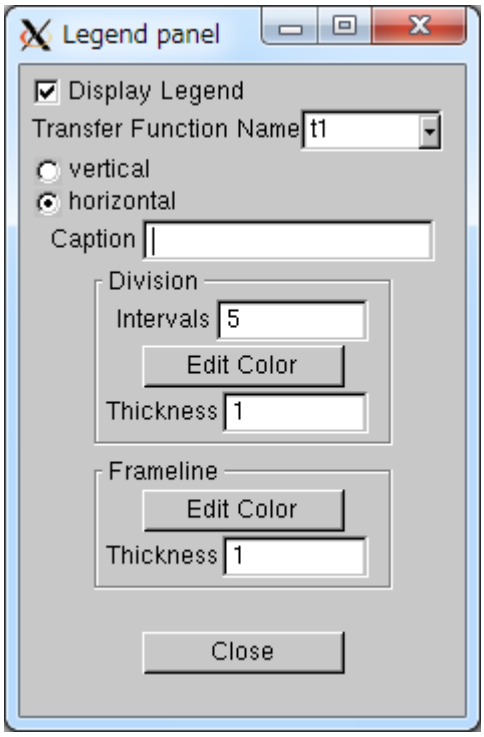


Figure 30 Legend panel

- **Display Legend**

  By turning on the Display Legend check box, the legend is shown.

- **Color Map Function**

  Selects a transfer function (t1-t5) to specify color map and range of legend with a pull-down menu.

- **vertical or horizontal**

  Specifies a direction of legend.

- **Caption**

  Specifies a caption of legend.

- **Division**

  Specifies a properties of tickmark in legend.

  Intervals : Number of tickmarks.

  Edit Color : Color of tickmark.

  Thickness : Thickness of tickmark.

- **Frameline**

  Specifies a properties of frame border in legend .

  Edit Color : Color of frame border.

  Thickness : Thickness of frame border.

- **Close**

  Close Legend panel.

Figure 31 shows an example of legend.

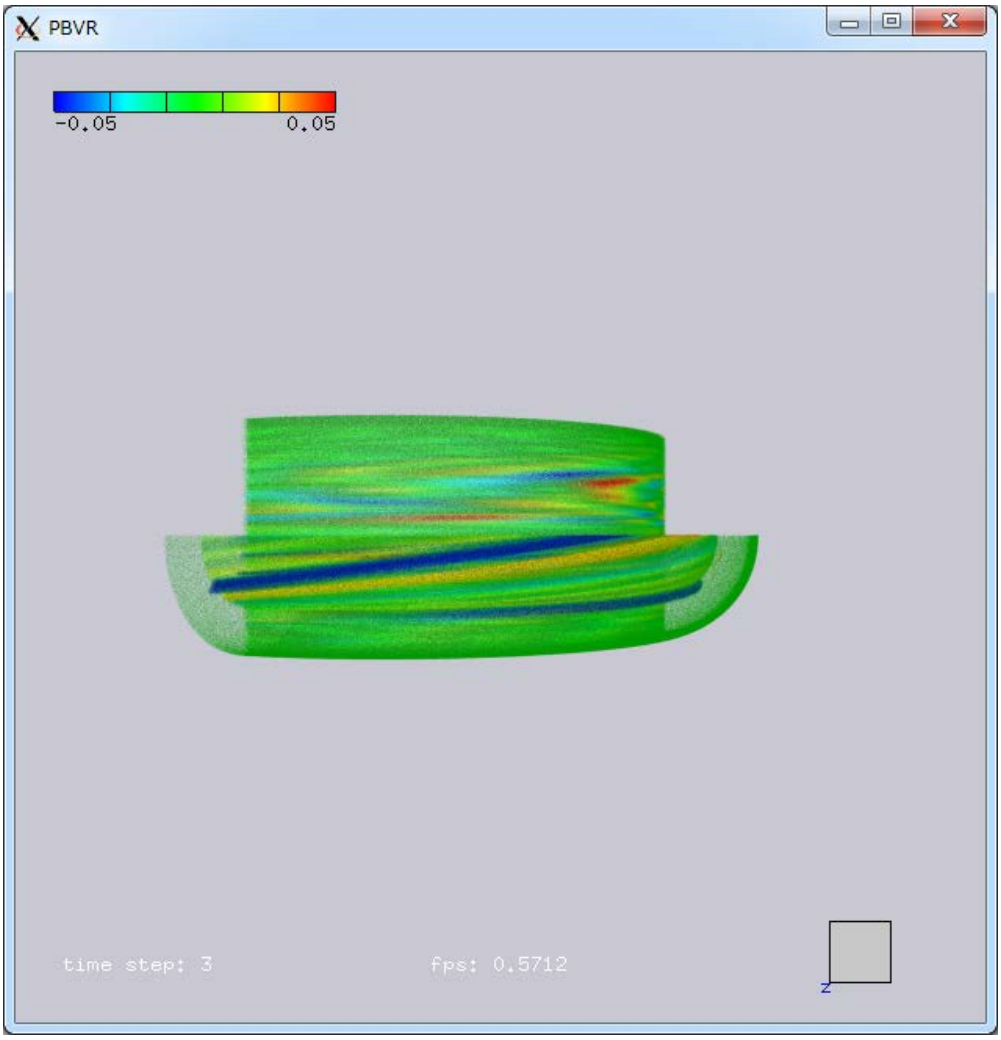


Figure 31　Example of legend

5.4.8 Coordinate panel

Figure 32 shows Coordinate panel, A coordinate change by designation of a numerical formula is performed to each coodinate axis. For example you can change from Cartesian coordinates system to cylindrical coordinates system.

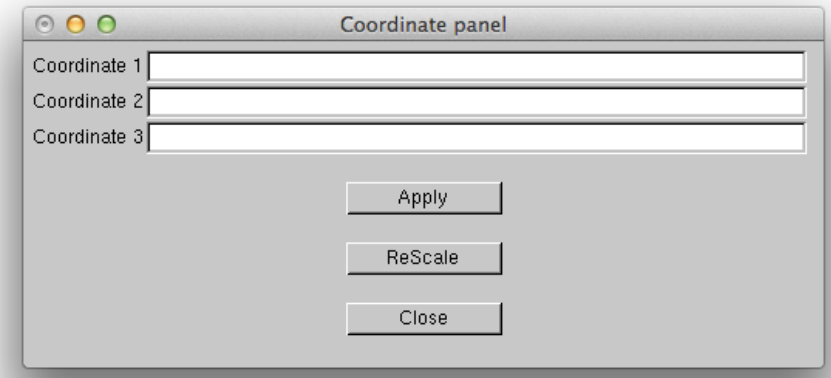


Figure 32 Coordinate panel

- Coordinate 1
  Specifies the numerical formula to calculate the new X coordinate.
  Empty or blank means just "X".
- Coordinate 2
  Specifies the numerical formula to calculate the new Y coordinate.
  Empty or blank means just "Y".
- Coordinate 3
  Specifies the numerical formula to calculate the new Z coordinate.
  Empty or blank means just "Z".
- ReScale Button
  Changes the scale of the view with the new coordinates.
- Apply Button
  Sends the transfer function created on this panel to the server.
- Close Button
  Closes Coordinate panel.

The text box Coordinate 1 - 3, it can be set coordinate transformation formula. Variables that can be described in the formula, is the original coordinates (X, Y, Z), physical data (q1, q2, ... , q9) and time (T). X, Y, Z, and T does not distinguish between the upper / lower case. In

addition, operations that can be described in the formula is the same as the transfer function editor (see 5.4.3.3). If specified physical data is not existed in the data, will be evaluated as 0. The text boxes of Coordinate 1 - 3, after describing the conversion formula, pressing the Apply button reflects the conversion.

5.4.9 Viewer control panel

Figure 33 shows Viewer control panel, which specifies a properties of viewer. Viewer control panel is activated by hitting the Viewer Control Panel button in Main panel. Each widget works as described in the followings.

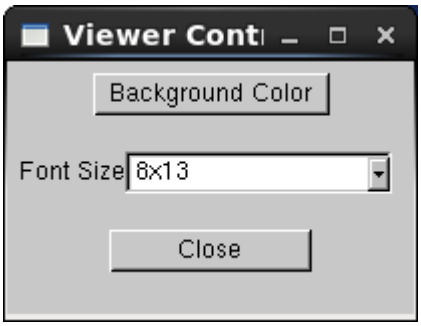


Figure 33 Viewer control panel

- **Background color**

  Specifies a background color of viewer.

- **Font Size**

  A font type of the character shown to a viewer is chosen. The available fonts are:

  8x13 A fixed width font with every character fitting in an 8 by 13 pixel rectangle.

  9x15 A fixed width font with every character fitting in an 9 by 15 pixel rectangle.

  TIMES_ROMAN_10 A 10-point proportional spaced Times Roman font.

  TIMES_ROMAN_24 A 24-point proportional spaced Times Roman font.

  HELVETICA_10 A 10-point proportional spaced Helvetica font.

  HELVETICA_12 A 12-point proportional spaced Helvetica font.

  HELVETICA_18 A 18-point proportional spaced Helvetica font.

- **Close**

  Close Viewer control panel.

6 An Example with the Sample Dataset

The following sections demonstrate the usage of PBVR for a sample dataset *gt5d.tgz*.

6.1 Launch PBVR

[step 1] Launch PBVR Server (which is the OpenMP version)

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

[step 2] Launch PBVR Client. This example uses the metropolis sampling and Phong Shading.

```
$ pbvr_client -S m -fin ./param.txt –tf demo.tf -shading P,0.6,0.6,0.6,30
```

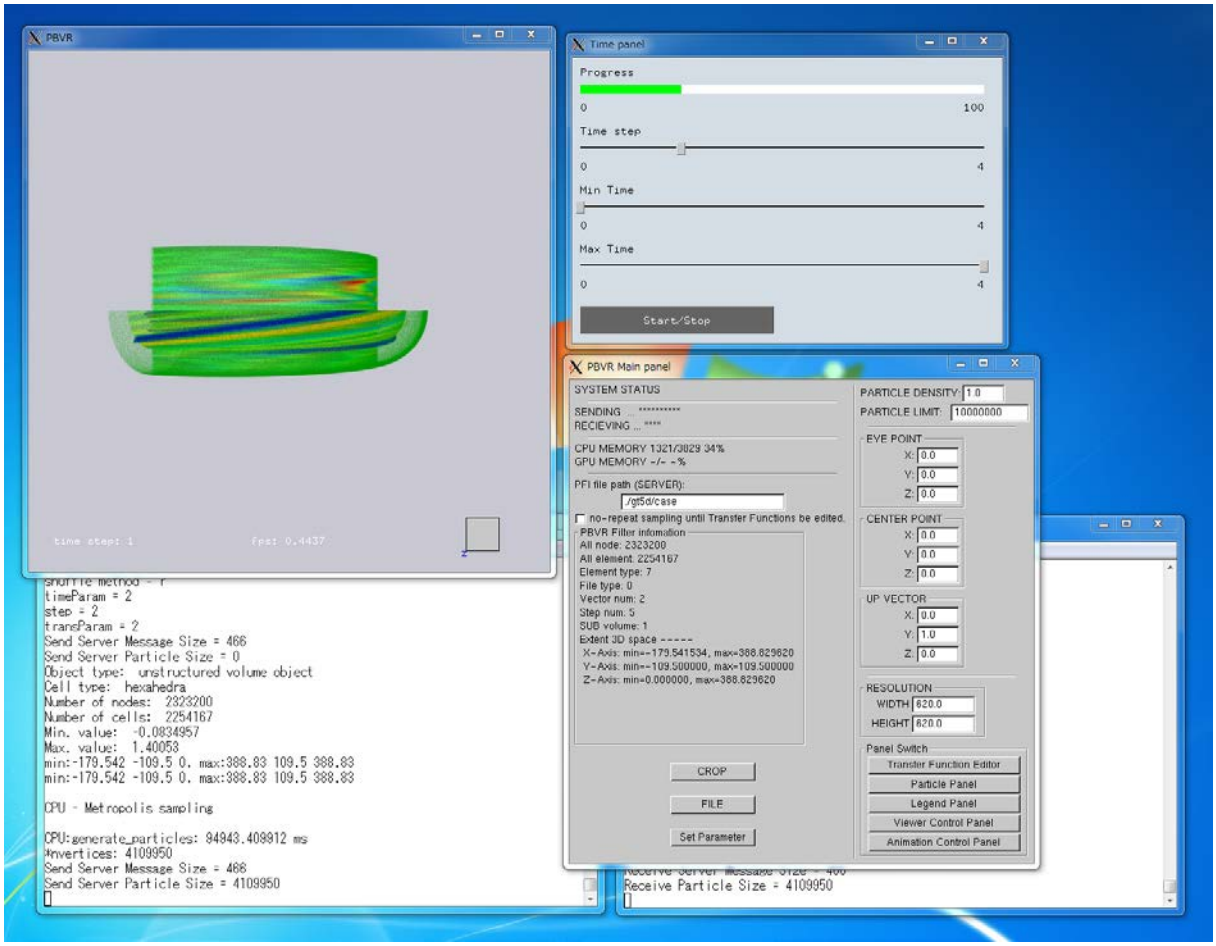


Figure 34　The GUIs of PBVR

6.2 Designing Transfer Functions

This section shows examples of visualizing *gt5d.fld*, using the multi-dimensional transfer function that is produced with the advanced transfer function design capability of PBVR. *gt5d.fld* contains structured grid volume data that consists of two variables.

6.2.1 Volume Rendering for a Single Variable

First of all, understand the variable *q1* by setting the transfer function *t1* as shown in Figure 35 . In this example, the transfer function is designed with **Transfer Function Editor**. Shown in the left of **Transfer Function Editor** is the configuration of colors, while in the right is that of the opacities. Notice that this configuration is the conventional volume rendering for a single variable.

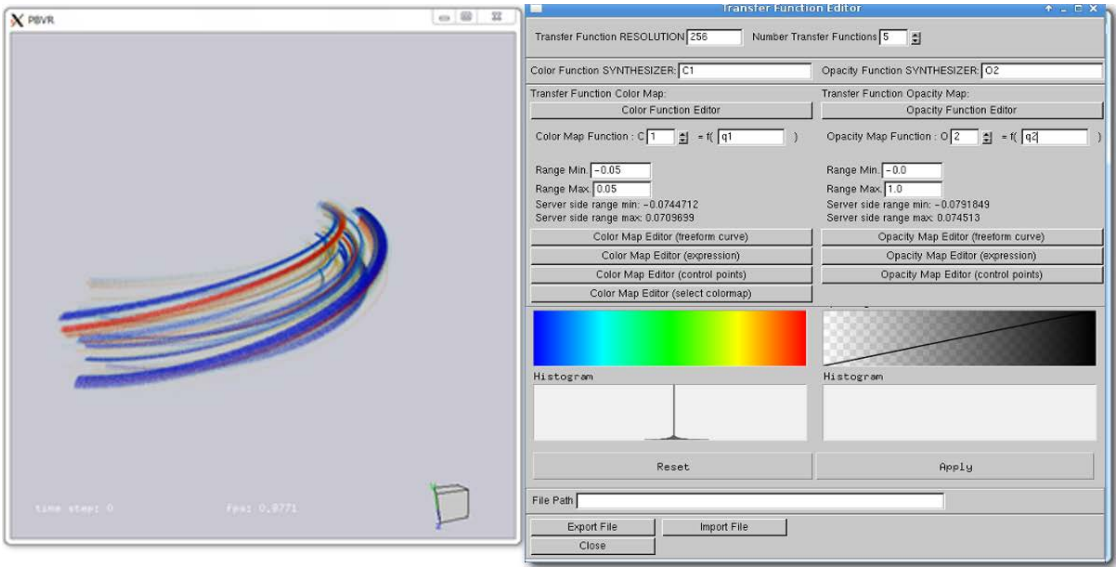


Figure 35　The volume rendering result for the variable *q1*

Color Function SYNTHESIZER　：C1
Opacity Function SYNTHESIZER　：O1
Color Map Function：C1 = f(q1)
Opacity Map Function：O1 = f(q1)

6.2.2 Multivariate Volume Rendering

The next example shows the result of multivariate volume rendering, in which the variables *q1* and *q2* are synthesized as shown below. In this example, the colors are assigned to C1=f(the variable *q1)*, while the opacities are assigned to O2=f(the variable *q2)*. The opacity map extracts two torus surfaces, which are given by the iso-surfaces of the variable *q2*. The colors encode the distribution of the *q1* values in these iso-surfaces.

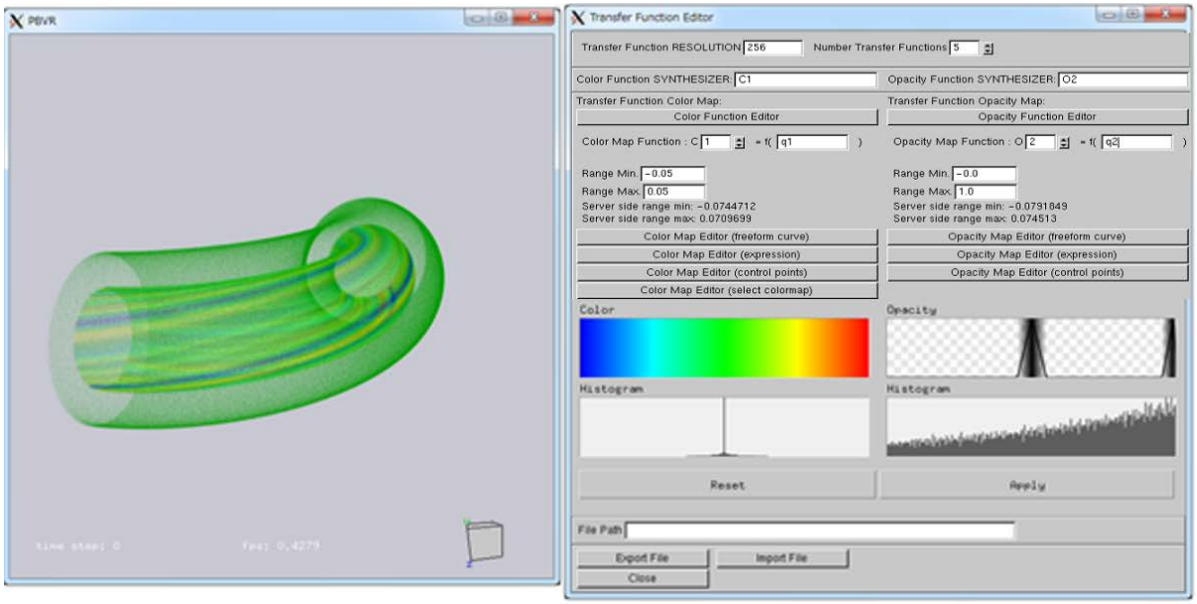


Figure 36　Rendering a multivariate volume. The *q1* values are color-mapped onto the iso-surfaces of *q2*.

Color Function SYNTHESIZER　：C1
Opacity Function SYNTHESIZER　：O2
Color Map Function：C1 = f(q1)
Opacity Map Function：O2 = f(q2)

6.2.3 Slicing Volumes

Figure 37 shows an application of PBVR's multivariate volume rendering for extracting a slice. With PBVR, an arbitrary function can be used to design a transfer function. In this example, the cylindrical surface (*X*^2+*Z*^2=*const.*) is extracted and the color of the variable *q1* is mapped onto it.

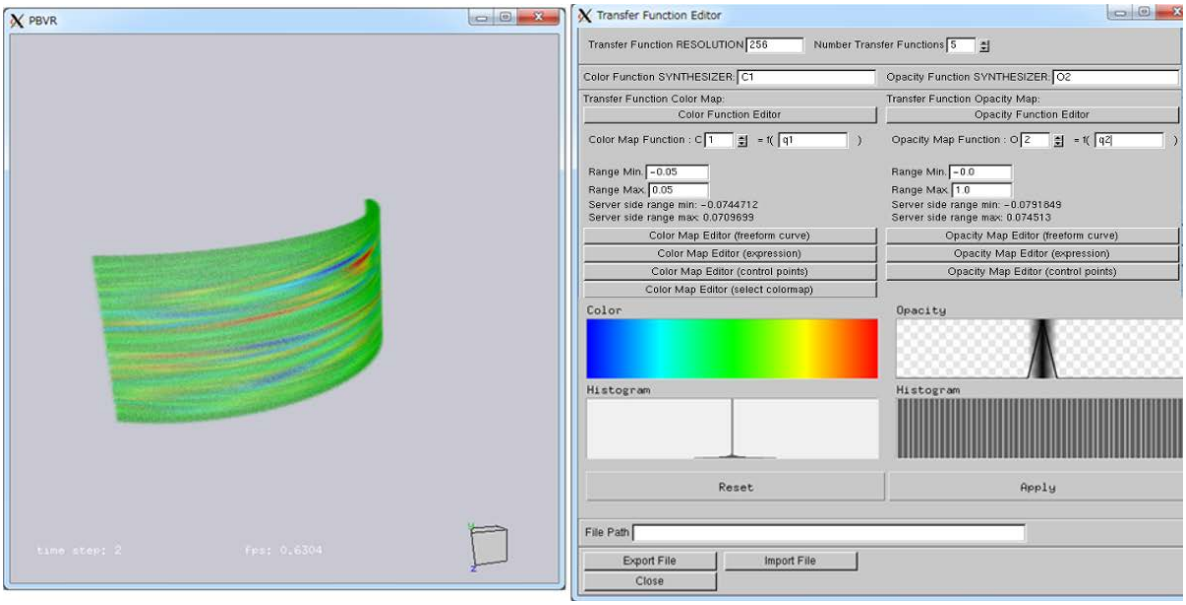


Figure 37　A rendering result for Slicing the volume with PBVR's multivariate volume rendering capability

Color Function SYNTHESIZER　：C1
Opacity Function SYNTHESIZER　：O3
Color Map Function：C1 = f(q1)
Opacity Map Function：O3 = f(X^2+Z^2)
Opacity Range Min：180
Opacity Range Max：380

6.2.4 Synthesis of Transfer Functions

This section explains how to synthesize transfer functions in PBVR. Figure 38 shows a transfer function *O4,* whose opacity function makes the region $Y > 0$ transparent. By synthesizing the previously described transfer functions *O1*, *O2,* and *O3* together with a new transfer function *O4* as (*O1* + *O2*) * *O4* + *O3*, the individually extracted sub-regions can undergo flexible composition through arithmetic operations. In this example, the colors of *t2* and *O3* are set to (*R*, *G*, *B*) = (0, 0, 0), while the color of *O4* is set to (*R*, *G*, *B*) = (1, 1, 1). In the above synthesis equation, the final colors obey the rainbow colormap defined for *O1*. On the other hand, the opacity of *O4* is multiplied to the sum of *O1* and *O2* in order to extract the lower half region ($Y < 0$) of *O1* and *O2*. Then, the resulting region is synthesized with the cylindrical surface given by *O3*. As revealed in these examples, PBVR's ability to synthesize transfer functions is powerful considering the capability to extract arbitrary region for each variable and to carry out a preferred series of operations.

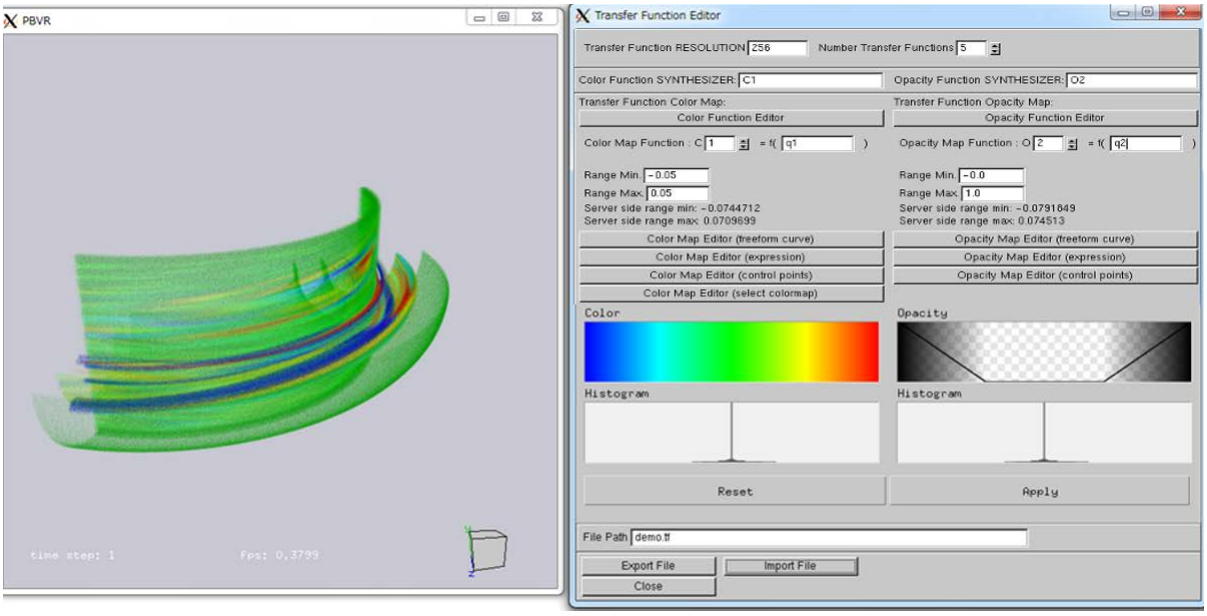


Figure 38　Synthesizing transfer functions

Color Function SYNTHESIZER　：(C1+C2)*C4+C3
Opacity Function SYNTHESIZER　：(O1+O2)*O4+O3

6.3 Integration of particle datasets

While the previous section shows a composition of volume rendering, iso-surfaces, and surface rendering via multi-dimensional transfer functions, the similar image composition is possible also by integrating multiple particle datasets. This section explains an example of particle integration.

6.3.1 Save particle datasets

Particle datasets are stored via **Particle File sub-panel** in **Particle panel**. Figure 39 shows an example of “Export Particle”. In this case, the following files are generated with the prefix “p1”.

./particle/p1_XXXXX_YYYYYYY_ZZZZZZZ.kvsml
./particle/p1_XXXXX_YYYYYYY_ZZZZZZZ_colors.dat
./particle/p1_XXXXX_YYYYYYY_ZZZZZZZ_coords.dat
./particle/p1_XXXXX_YYYYYYY_ZZZZZZZ_normals.dat

Here, XXXXX is the time step, YYYYYYY is the sub-volume number, and ZZZZZZZ is the total sub-volume number. “colors”, “coords”, and “normal” contain color, coordinates, and normal vector of each particle, respectively. By hitting the **Export** button, integrated particle data is stored in the above files, and during the saving process, the **Export** button is de-activated, and after whole time series data is stored, the **Export** button becomes active again.

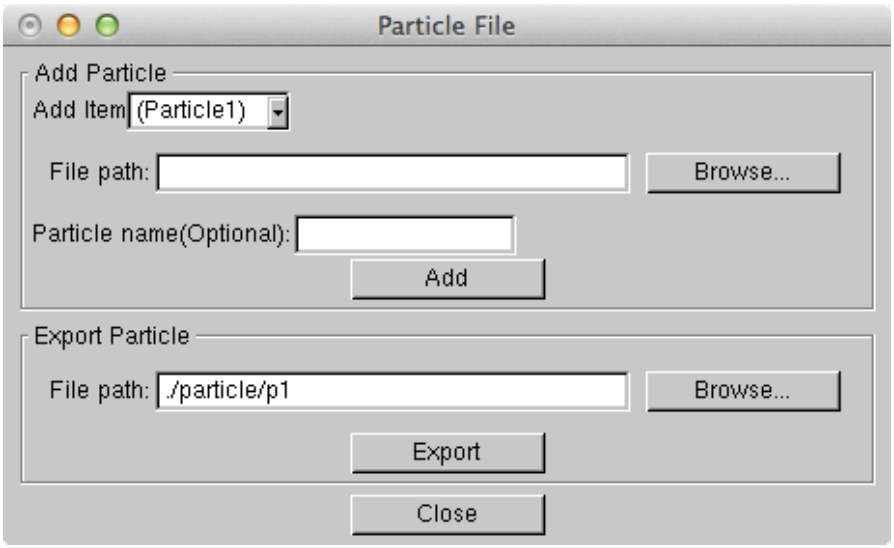


Figure 39 Particle File panel (**Export** is active)

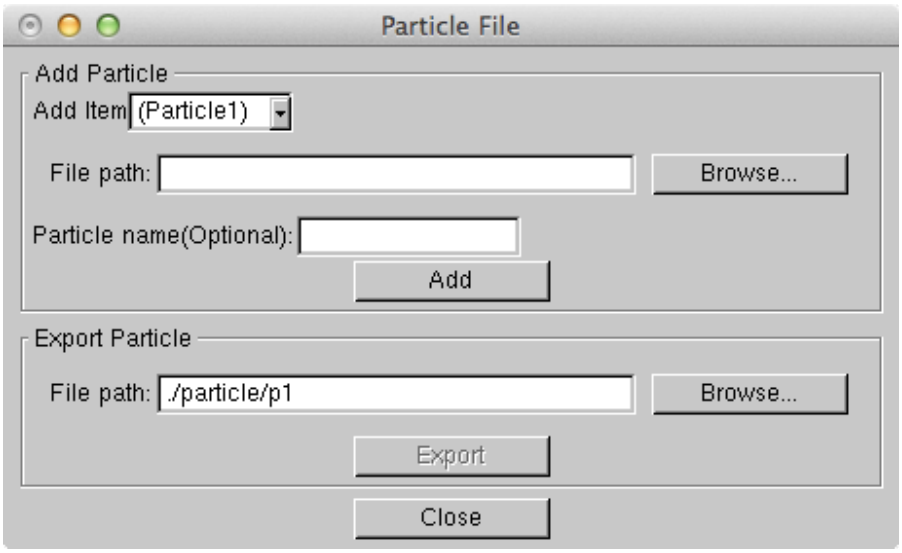


Figure 40 Particle File panel (**Export** is non-active)

6.3.2 Load particle datasets

In the following example, three particle datasets p1, p2, and p3, which corresponds to the images in Figures 35~37, are loaded and integrated. Here, PBVR Client in stand-alone mode is launched with the following command, and the particle datasets are specified in the command line options. (In client-server mode, particle datasets are specified in **Particle panel**.)

$ pbvr_client -shading P,0.6,0.6,0.6,30 –pin1 ./particle/p1 –pin2 ./particle/p2 –pin3 ./particle/p3

After launching, p1~p3 are loaded in **Particle panel**. By turning on the **Display particle** check box for p1, the volume rendering is shown as in Figure 41. In addition, by turning on the **Display particle** check boxes for p2 and p3, all three particle datasets are integrated as in Figure 42. The integrated particle data can be stored as a single particle dataset via **Particle File panel**. It is noted that in order to obtain correct integrated images, all particle datasets have to be generated by using the same “particle density” and “particle limit” parameters, which are specified by the command line options, “-pd” and “-plimit”, or by the **Main panel**.

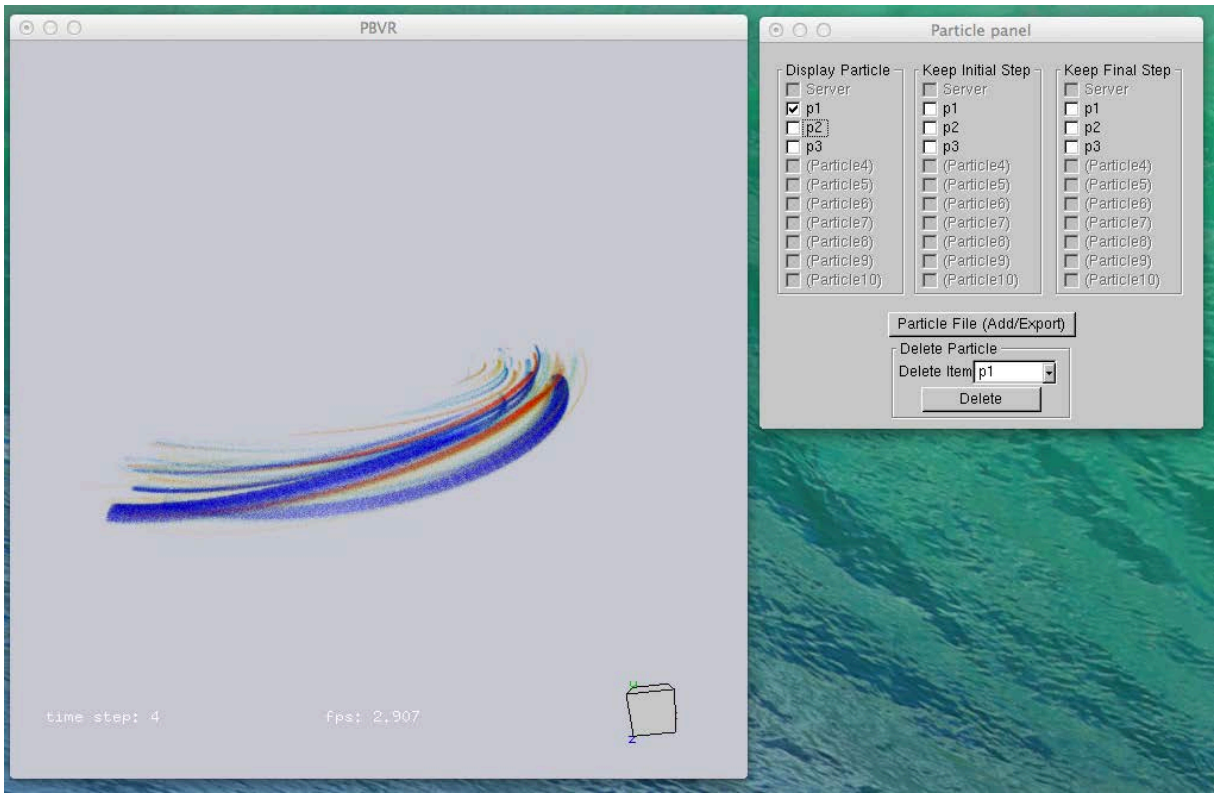


Figure 41 Particle dataset p1

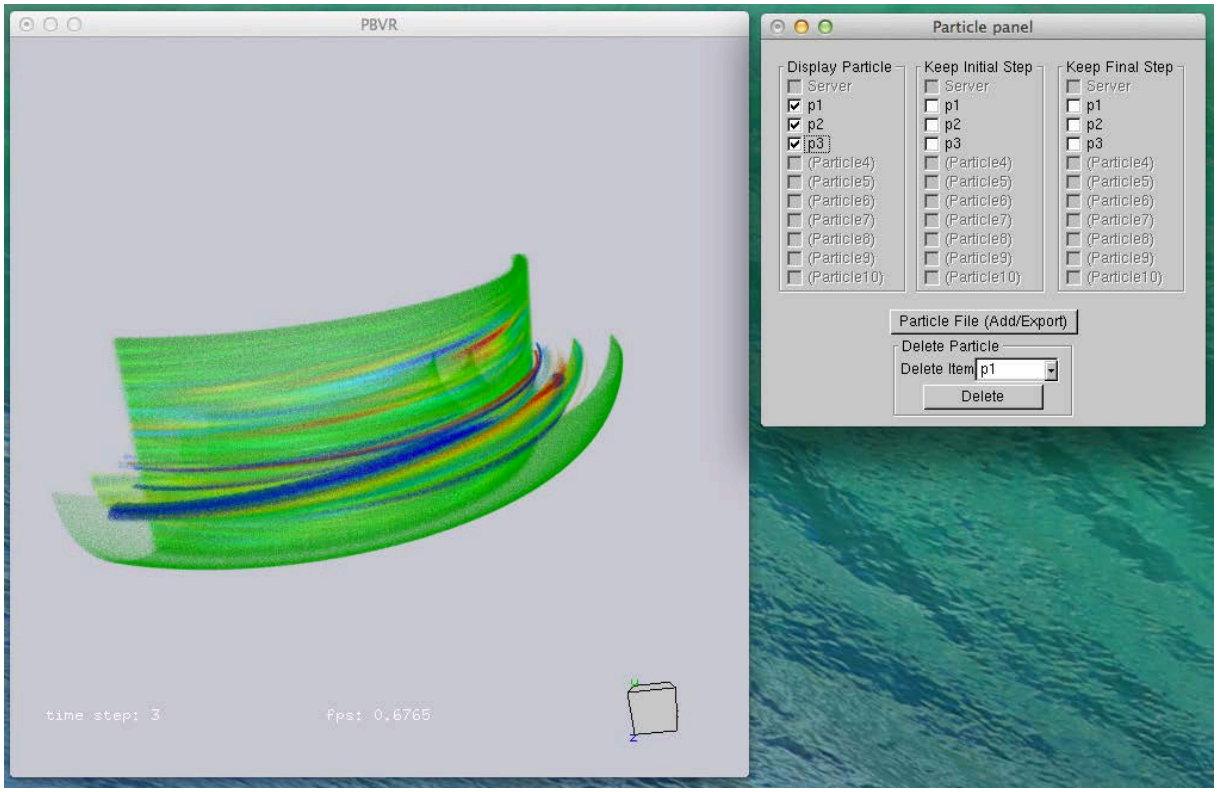


Figure 42 Integration of p1, p2, and p3

6.4 Saving Results

After designing the transfer function, PBVR can save the resulting image and parameters in following three ways.

1) Image output (5.4.6)
   In order to save the results as images, select **on** from the **capture** drop down menu in **Animation panel**. The bitmap image files (PBVR_image.xxxxx.bmp) are generated.
2) Transfer function file (5.4.3)
   In order to generate the transfer function file, write file name of the transfer function in **File Path** field of **Transfer Function Editor** and press **Export File**. Later, this file can be loaded by hitting **Import File**.
3) Visualization parameter file (5.4.2)
   In order to run PBVR Server in batch mode, all the visualization parameters including the transfer function can be exported. Open **File panel** from **Main panel**, specify the parameter filename, and press **Export File**.

6.5 Example of Batch Mode

This section explains how to run PBVR Server in batch mode using the visualization parameter file exported in the previous section. This mode is developed for carrying out massively parallel processing with supercomputers. In addition, this mode is useful also for high speed rendering of time series data with PBVR Client in stand-alone mode, since the latency due to particle generation and particle data transfer can be eliminated.

[Step 1] Launch PBVR Server (of OpenMP version) in batch mode

```
$ pbvr_server –B –fin ./param.txt –pout ./output/case –pa ./param.in
```

[Step 2] Launch PBVR in stand-alone mode

```
$ pbvr_client –pin1 ./output/case -shading P,0.6,0.6,0.6,30
```