1 Constants in vismo_const.F90

Various constants such as the maximum number of cameras are defined in vismo_const.F90, so rewrite as necessary. some examples are given below.

- 1.1 integer, parameter :: vsmImgKind = 2 ! 0:BMP, 1:PPM, 2:PNG : image format. When this is 0 or 1, -lpng or png4vismo.c is not necessary.
- 1.2 integer, parameter :: vsmMaxCamera = 256 : maximum number of cameras
- 1.3 integer, parameter :: vsmMaxVisKind = 15 : maximum number of visualizations

2 Compiling VISMO

Fortran compiler and MPI Library are required

2.1 Options

- If isnan cannot be used with your Fortran compiler, add -DNO_ISNAN.
- If you want to visualize real (single precision) data, add -DFLOAT (vismo__AddScalar and vismo__AddVectorsscalar, vectx, etc. described later will accept real (single precision) arrays)
- If PPM or BMP images are not output properly, rewrite the open statement of vsmWriteImageBMP or PPM in visimo_image.F90 according to the environment. Or add -DCBMPPPM and replace that part with a C language function written in bmpppm4vismo.c.
- When outputting PNG, add the -DCPNG and set vsmImgKind to 2 in vismo_const.F90.
- If the arrows and streamlines are drawn very slowly, Fortran's transfer function may cause the problem(Fujitus FX100, etc.). by adding -DCTRANS, replace it with a C language function written in transfer4vismo.c.
- -DSERIAL is required when coupling with a simulation code that is not MPI parallel. (the Cartesian version only)
- -DCLANG is required when coupling with a simulation code written in C language. You also need to compile cvismo.c. In addition, -DUSE_GFORT for gfortran, or -DUSE_IFORT for intel should be added. Other compilers can also be used by rewriting cvismo.c.

2.2 Makefile

Please refer to Makefile and test8.f90 for simulation codes written in Fortran, MakefileC and test8c.c for codes in C langurage, and MakefileS and test1.f90, test1c.c for serial codes (not parallelized by MPI).

3 Coupling with simulation codes

3.1 Coupling with MPI parallelized simulation codes written in Fortran

Please refer to test8.f90, test8rectl.f90 and Makefile.

First, you should write at the beginning of your program

```
use vismo
```

Call the following subroutines from the appropriate location in your simulation code.

```
call vismo__init(mpi_comm_world, myrank, psize, "configtest.vsm") ! Initialization of vismo.
```

The arguments are MPI communicator, process rank(integer number), process size (integer number), and vismo visualization config file name.

```
call vismo__addScalar(scalar, LXSIZE, LYSIZE, LZSIZE)
```

call vismo__addVector(vectx, vecty, vectz, LXSIZE, LYSIZE, LZSIZE)

scalar, vectx, vecty, vectz are arrays of data to visualize. LXSIZE, LYSIZE, LZSIZE are the dimensions of the data in the x, y, and z directions.

or

call vismo__addScalar(scalar, SIZE)
call vismo__addVector(vectx, vecty, vectz, SIZE)
in this case, SIZE is integer, dimension(3), and SIZE=[LXSIZE, LYSIZE, LZSIZE].

call vismo__addPtcl(p_x, p_y, p_z, q, n)
or
call vismo__addPtclCol(p_x, p_y, p_z, q, scal, n) :: when using color map
These variable arrays require the target attribute. If q is 0, it is drawn, otherwise it is not drawn.
n is the number of particles
scal is an array of scalar value of the particles. The colors for particles are determined using the values and the color map.

With n = 5, and the contents of the array q (integer type) are
0 0 0 1 0
the fourth sphere is not drawn.
The content of the array scal (double precision) is 0.0 0.1 0.2 0.3 0.4, and the color of each particle is determined by this value. p_x, p_y, p_z are arrays of double-precision real numbers, and are the coordinates of each particle (x, y, z)

!initialization of coordinates for the Cartesian version
call vismo__initCoords(xx) ! xx = vismo__uniform or vismo__rect

!initialization of coordinates for the Yin-Yang version
call vismo__initCoords(xx, rad) ! xx = vismo__Yin or vismo_Yang
! rad(1):min radius, rad(2):max radius, rad(3): dr

call vismo__setUniCoord(n, corner, dx)
call vismo__setLocalUniCoord(ln, lx0, rln, loccorner, dx)
!setting coordinates. Details are described below.

call vismo__preparevis
! Preparing vismo. Call before vismo__visualization.

call vismo__visualization(time) ! arg is timestep (integer)
Call this subroutine where you want to visualize data and generate image(s). The argument is used for the time stamp appearing file name.

call vismo__finalize
Call this subroutine at the end of your simulation code. This subroutine frees the memory allocated by vismo.

About setting coordinates, please see "makeCrdData" subroutine.
Uniform Grid
○vismo__setUniCoord

call vismo__setUniCoord(n, corner, dx)
n: grid size in x, y, z direction (whole data, overlap not counted)
corner: the Origin position.
dx: spacing in the x, y, z directions.

○vismo__setLocalUniCoord
call vismo__setLocalUniCoord(ln, lx0, rln, loccorner, dx)
This is the coordinate information of the local area that the process is in charge of.
ln: grid size in x, y, z directions (including overlap)
lx0: grid number at the beginning of "area used for visualization"
rln: grid size excluding overlap from ln
loccorner: The leftmost coordinates of the local data (including ○).
dx: the same as vismo__setUniCoord's dx

Illustration (□ is for adjusting the positions of ○ and ●)
○●●●●●○
□□□□□□●●●●●●○
□□□□□□□□□□□□●●●●●●○
□□□□□□□□□□□□□□□□□□●●●●●●○

○ are the overlapped grid points, ● are the grid points that determine the area used for visualization. It is assumed that data is also included in ○. If there is no data, there are some inconveniences such as holes or stains of the isosurface.

ln is the sum of ○ and ●.
rln is the number of ●.
The position of the black circle on the left end is lx0.
The leftmost coordinates of the local data are the loccorner.
As mentioned above, if you do not set the VISMO coordinates so that ●s overlap, VISMO will not be able to visualize data well.
The grid number assigned to lx0 is a number starting from 1 in each process. It is not the number of the entire data.

RectiLinear grid (the Cartesian version)
Instead of vismo__setUniCoord, vismo__setRectCoord(n, coordx, coordy, coordz) has to be used.

integer, dimension(3), intent(in) :: n
real(kind=vdp), dimension(nx(1)), intent(in) :: coordx // double precision
real(kind=vdp), dimension(nx(2)), intent(in) :: coordy // double precision
real(kind=vdp), dimension(nx(3)), intent(in) :: coordz // double precision

Instead of vismo__setLocalUniCoord, vismo__setLocalRectCoord(ln, lx0, rln, coordx, coordy, coordz) has to be used
integer, dimension(3), intent(in) :: ln, lx0, rln
real(kind=vdp), dimension(nx(1)), intent(in) :: coordx // double precision
real(kind=vdp), dimension(nx(2)), intent(in) :: coordy // double precision
real(kind=vdp), dimension(nx(3)), intent(in) :: coordz // double precision

3.2 Coupling with serial codes written in Fortran (not parallelized by MPI)
Please refer to test1.f90 and MakefileS. -DSERIAL is required.

The initialization subroutine has no arguments for mpi communicator, process number, or process size, so that

```
call vismo__init ("configtest.vsm")
```

To set the coordinates, set n = lx = rln, lx0 = 1, corner = loccorner.

3.3 Coupling with codes written in C language

Please refer to test8c.c, test1c.c, MakefileC, MakefileS. -DCLANG is required.

```
#incude "vismo.h"
```

is required like usual C langurage programs. You also need to compile cvismo.c.
The function name is the same as the Fortran subroutines above. An example of an array of data is one-dimensional, but I think three-dimensional arrays are accepted if they are contiguous and cast to one-dimensional. -DSERIAL is required if MPI is not parallel.
I have made it possible to compile with gcc + gfortran (-DUSE_GFORT) and Intel C + Intel Fotran (-DUSE_IFORT), but other compilers can be used. Please execute nm vismo.o to find out what the name of the fortran subroutines are, and rewrite cvismo.c. For example, in gfortran, vismo__visualization is included as ___vismo_MOD_vismo__visualization.

4 Configuration file of VISMO

4.1 Common

The delimiter is a space, not TAB. Please refer to configtest.vsm and configtest2.vsm.
Sphere rendering, and semi-transparent isosurface and slices are supported only for the Cartesian version.

4.2 Keywords

Lowercase letters are places to write numbers.
Reading another file
INCLUDE FIlename

Image pixel size
IMAGESIZE width height //width×height

Thickness of frame
FRAMEWIDTH width //width=thickness

Background color
BGCOLOR r g b //r,g,b 0.0-1.0

Viewing point, direction etc
CAMERA
[PERSPECTIVE or ORTHOGONAL] // If omitted, Perspective
POSITION x y z // Camera position
FRONT x y z // Front direction of camera
UP x y z // up direction of camera
FOVY f // Required for perspective. The unit is degree
NEARFAR near far // Do not draw volume closer to near, and farther than far
WIDTH w // Required for Ortho. w is the horizontal length of the screen. Vertical length is calculated from IMGESIZE
END_CAMERA

If you want to create multiple cameras automatically. If you set the axis, radius, center and the

number of cameras, multiple cameras will be automatically generated.
MULTIPLECAMERAS
[PERSPECTIVE or ORTHOGONAL] // If omitted, Perspective
AXIS x y z // direction of rotation axis
RADIUS r // radius of rotation
CENTER x y z // center of rotation
NVAMERA n //number of cameras
FOVY f // Required for perspective. The unit is degree
NEARFAR near far // Do not draw volume closer to near, and farther than far
WIDTH w // Required for Ortho. w is the horizontal length of the screen. Vertical length is calculated from IMGESIZE
END_MULTIPLECAMERAS

lighting
LIGHT
[PARALLEL or POINT] // if omitted, parallel
POSITION x y z //position of light
AMB r g b //intensity of ambient light
DIF r g b // intensity of diffusion light
SPEC r g b // intensity of specular light
SHIN s //parameter of specular light
END_LIGHT
The order of POSITION, AMB, etc do not have to be this way. The range of RGB is from 0.0 to 1.0

Depth cueing
DEPTHCUE kind near far
kind: 1:linear, 2 exp(-x), 3:exp(-x*x)
The volume between near and far will be darken gradually.

colormaps
SCAL_COLORMAP
SCALNUM n //n=the scalar number
EQUALLY_SPACED // if the values are equally spacing
value1 r g b a // the range of R,G,B,A is from 0.0 to 1.0
value2 r g b a
• • • •
END_SCAL_COLORMAP

VECT_COLORMAP
VECTNUM n //n=the vector number
EQUALLY_SPACED// if the values are equally spacing
value1 r g b // there is no "a" unlike SCAL_COLORMAP
value2 r g b
• • • •
END_VECT_COLORMAP

PTCL_COLORMAP
PTCLNUM n //n=the particle number
EQUALLY_SPACED // if the values are equally spacing
value1 r g b rad // the range of R,G,B is from 0.0 to 1.0. RAD is radius of the particle
value2 r g b rad

・ ・ ・ ・
END_PTCL_COLORMAP

Visualization parameters
enclose
VISUALIZATION
and
END_VISUALIZATION

4.3 Description: VISUALIZATION～END_VISUALIZATION

ISOSURFACE
SCALNUM n //the scalar number
LEVEL level //isosurface level
AMB r g b //users determines the materials (colors) of isosurface
DIF r g b
SPES r g b
SHIN shin
[ALPHA alpha] // opacity. If omitted, alpha=1.0
END_ISOSURFACE

SLICE
SCALNUM n
EQ a b c d // f(x,y,z) = ax+by+cz+d=0
AMB amb // users determine only reflection retio
DIF dif //
SPEC spec
SHIN shin
[ALPHA alpha] // opacity. If omitted, alpha=1.0
END_SLICE

VOLUME_RENDERING
SCALNUM n
AMB amb
DIF dif
SPEC spec
SHIN shin
END_VOLUME_RENDERING

STREAM_LINES
VECTNUM n
AMB [r g b or amb] // if r,g,b are specified and MONO is set, it is the color of streamlines
DIF [r g b or dif] // if only reflection ratio is specified, the streamlines are colored
SPEC [r g b or spec] //by the magnitude of vector field
SHIN shin
RADIUS rad //radius of stream tubes
[MONO or COLOR] //if omitted, COLOR
SEED // seeds of stream lines
x y z
x y z
…
END_SEED
END_STREAM_LINES

ARROWS
VECTNUM n
EQ a b c d // f(x,y,z) = ax+by+cz+d=0
DENSITY d // density of arrows
RADIUS r //radius of arrows
LENGTH l //length of arrows
END_ARROWS

PARTICLE
PTCLNUM n
COLOR //when using colormap
RADIUS r // radius of particle. When using colormap, this is not necessary.
AMB [r g b or amb] // when using colormap, only reflection ratio has to be specified.
DIF [r g b or dif]
SPEC [r g b or spec]
SHIN s
END_PARTICLE

[DEPTHCUE]
Turn on depth cueing.

LIGHT n
Turn of the light number n

INNERSPHERE
(only the YinYang version)

OUTERSPHEREFRAME
(only the YinYang version)