Visualization

The visualization comprises the visualization of the instrument and the visualization of trajectories.

General

In principle, there are 4 different ways to run the simulation: normal run, instrument visualization, visualization of trajectories and a test run – Table 1 summarizes the main features of these options.

	bVisInstr	bVisTraj	bTest	instrument.inf	features
normal run	TRUE	FALSE	FALSE	written	
instrument visualization	TRUE	FALSE	FALSE	written/needed*	
trajectory visualization	TRUE	TRUE	FALSE	needed	
check/test	TRUE	FALSE	TRUE	written	no treatment of trajectories

Table 1: Main features of the 4 options to run a VITESS simulation. At present, instrument.inf is needed also for the instrument visualization, if a cylindrical detector is included.

In the test run no trajectories are treated, but all checks are executed, all input files for the simulation are written and *instrument.inf* is generated. It can be used to start the trajectory visualization, because this needs *instrument.inf* to run. Writing this file only works at the end of the simulation, but the visualization of trajectories needs it during the simulation.
For the moment, instrument visualization is performed during a normal run, so that there are only 3 options realized.

The following global parameters record the position and orientation of the component in the absolute co-ordinate system:

```
double   Length,           /* length of beamline from source to origin of this module [cm] */
         RotZ, RotY;       /* hor. and vert. rotation of the local co-ordinate system relative
to the absolute one  */
double   RotMatrixM[3][3], /* matrix to rotate between these co-ordinate systems */
         RotMatrixS[3][3]; /* matrix to rotate from abs. co-ordinate system to co-ordinate
system of last section   */
VectorType BegPosM,        /* end position of prev. module = origin of this module in absolute
co-ordinate system  [cm]  */
           BegPosS;        /* end position of prev. section = origin of this section in
absolute co-ordinate system [cm] */
```

BegPosS and *RotMatrixS* are only used in Guide, because this is the only module that treats elements in local co-ordinate systems. *RotMatrixM* is currently needed for a correct visualization of the cylindrical detector.

The unit for positions in the output files is meter.

Test or Check run

Writing the intersection points requires that *instrument.inf* already exists up to the preceding module, i.e. that a calculation of the whole instrument had been executed before, but of course a simulation of trajectories is not necessary. This is controlled by the parameter *bTest*. It is declared in *init.c* and set to *FALSE* by default. If the parameter --t is set, it is changed to

TRUE in *Init()*. *bTest* prevents reading of trajectories in *ReadNeutrons()*. In this way, only checks are executed and the input and inf files are generated.
This also allows the user to check the generated input files, e.g. the guide shape resulting from the input for an elliptic or parabolic guide, before he starts the simulation.

Visualization of the Instrument.

The instrument visualization is controlled by the parameter *bVisInstr*. It is declared in *init.c* and set to TRUE by default, the default geometry file name is *geometry.inf*. If the parameter --v*instr-file* is set, the file name is set to *instr-file* in *Init()*.

The structure that has to be filled is *VtModGeom stGeometry*. *VtModGeom* and all structures that it contains are defined in *general.h as*. *stGeometry* is declared in *init.c* as a global variable. *VtModGeom* contains pointers to all geometrical elements that can be visualized and their numbers:

```
typedef struct
{
  VtModID      eModule;
  VtLine*      pLine;
  int          nLines;
  VtRectangle* pRectangle;
  int          nRectangles;
  VtTriangle*  pTriangle;
  int          nTriangles;
  VtOpenRect*  pOpenRect;
  int          nOpenRects;
  VtCircle*    pCircle;
  int          nCircles;
  VtCuboid*    pCuboid;
  int          nCuboids;
  VtHull*      pHull;
  int          nHulls;
  VtCylinder*  pCylinder;
  int          nCylinders;
  VtHolCyl*    pHolCyl;
  int          nHolCyls;
  VtEllipsoid* pEllipsoid;
  int          nEllipsoids;
  VtSphere*    pSphere;
  int          nSpheres;
  VtCylSlice*  pCylSlice;
  int          nCylSlices;
  const char*  pDescr;   /* description   */
}
VtModGeom;
```

VtHolCyl is a hollow cylinder. *VtHull* has rectangular top and bottom (or entrance and exit), with the normal through the centers coinciding and the edges parallel; the corners of the rectangles are connected by straight lines. For details of any geometrical element see structure definitions in *general.h*.

Each module has to fill this structure at the end of the program, but before *Cleanup()* is called, e.g. in *OwnCleanup()*. Here all positions and orientations of the elements describing the component are given in a relative coordinate system.
The file *instr-file* is then opened and one line appended in the function *WriteGeomData()* after writing the instrument file *instrument.inf* in *WriteInstrData()*, both called in the function *Cleanup()*. *WriteGeomData()* takes the absolute position of the end of the previous module from *instrument.inf*, which is read in *ReadInstrData()*, and calculates the absolute values of the elements from these positions and the relative positions given in *stGeometry*.

Therefore, writing the geometry file works in the same way as writing *instrument.inf* and should therefore not be a problem.

The output files look like this:

```
DEF red=<Material diffuseColor='.9 .01 .01' emissiveColor='.9 .01 .01' transparency='.4'/>
DEF green=<Material diffuseColor='.01 .9 .01' emissiveColor='.01 .9 .01' transparency='.4'/>
DEF blue=<Material diffuseColor='.01 .01 .9' emissiveColor='.01 .01 .9' transparency='.4'/>
DEF yellow=<Material diffuseColor='.9 .6 .01' emissiveColor='.9 .6 .01' transparency='.3'/>
DEF orange=<Material diffuseColor='.9 .4 .01' emissiveColor='.9 .4 .01' transparency='.4'/>
DEF cyan=<Material diffuseColor='.0 .99 .99' emissiveColor='.0 .99 .99' transparency='.4'/>
DEF magenta=<Material diffuseColor='.9 .01 .6' emissiveColor='.9 .01 .6' transparency='.4'/>
DEF grey=<Material diffuseColor='.6 .6 .6' emissiveColor='.6 .6 .6' transparency='.4'/>
DEF black=<Material diffuseColor='.01 .01 .01' emissiveColor='.01 .01 .01' transparency='.4'/>
DEF white=<Material diffuseColor='.99 .99 .99' emissiveColor='.99 .99 .99' transparency='.4'/>
#
#units
#  [m]  position, length, width, height, radius
# [deg] angels
#
Rectangle       0.00000   0.00000   0.00000     1.00000   0.00000   0.00000      0.12000   0.12000    0.00000 Source:blue
Rectangle       2.00000   0.00000   0.00000     1.00000   0.00000   0.00000      0.08700   0.08700    0.00000 Source:blue
Hull            4.00000   0.00000   0.00000     1.00000   0.00000   0.00000      4.00000   8.67000    2.00000     8.67000    2.00000
0.00000   elliptic guide:yellow
Circle          6.00000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.32500   0.01000  359.99000   disc chopper:white
Circle          6.00000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000  88.73590   69.71010   :white
Circle          6.00000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000  30.51585    4.43815   :white
Circle          6.00000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 333.29585  298.16615   :white
Circle          6.00000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 275.57580  232.39420   :white
Circle          6.00000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 217.85580  166.62220   disc chopper:white
Circle          6.54000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.32500   0.01000  359.99000   disc chopper:white
Circle          6.54000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000  69.70485   55.20315   :white
Circle          6.54000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 363.93345  340.25655   :white
Circle          6.54000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 298.16105  265.30895   :white
Circle          6.54000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 232.38865  190.36135   :white
Circle          6.54000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 166.61745  115.41655   disc chopper:white
Circle          7.40000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.32500   0.01000  359.99000   disc chopper:white
Circle          7.40000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000  63.91870   47.64530   :white
Circle          7.40000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 351.97525  326.52675   :white
Circle          7.40000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 280.03185  245.40815   :white
Circle          7.40000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 208.08745  164.28855   :white
Circle          7.40000  -0.33750   0.00000     1.00000   0.00000   0.00000      0.35000 136.14400   83.18400   disc chopper:white
Hull            9.55000   0.00000   0.00000     1.00000   0.00000   0.00000      4.30000   2.00000    9.00000     2.00000    9.00000
0.00000   elliptic guide:yellow
Hull           14.85000   0.00000   0.00000     1.00000   0.00000   0.00000      6.30000   9.00000    9.00000     9.00000    9.00000
0.00000   elliptic guide:yellow
Hull           23.00000   0.00000   0.00000     1.00000   0.00000   0.00000     10.00000   9.00000    9.00000     9.00000    9.00000
0.00000   elliptic guide:yellow
Hull           39.00000   0.00000   0.00000     1.00000   0.00000   0.00000     22.00000   9.00000    9.00000     9.00000    9.00000
0.00000   elliptic guide:yellow
Rectangle      60.00000   0.00000   0.00000     1.00000   0.00000   0.00000      0.15000   0.15000    0.00000 monitor1D:grey
```

The colors that are defined in the beginning of each geometry file are used by instrument components in a way that improves the visibility. The color is written in the component discription after colon. The component label only appears for the first and the last of a group of the same geometrical elements.

The parameter *bVisInstalled* handles if the visualization option is included in a module or not. The initial value is FALSE; therefore all modules that have visualization implemented have to set:

```
bVisInstalled = TRUE;
```

For the case *bVisInstalled* = FALSE, a square perpendicular to the beamline is drawn for modules of length zero, and a cylinder with the beamline axis as symmetry axis is drawn for modules with length greater zero. (This means that for some modules the implementation can be omitted.)

Visualization of Trajectories

The visualization of the trajectories is controlled by the parameter *bVisTraj*. It is declared in *init.c* and set to *FALSE* by default. If the parameter --V*traj-file* is set, it is changed to TRUE in *Init()*. At the same time the trajectory file is opened and the file pointer *TrajFilePtr* filled.

Writing of interaction points into the trajectory file *traj-file* is carried out in the function *WriteWWP(Neutron *pNeutron, VtReason eReason)* in init.c. *VtReason* is defined in general.h and can 8 different values (*VT_CREATED, VT_OUTSIDE, VT_OUT_OF_WND, VT_PASSED, VT_ENTERED , VT_TRANSIT, VT_REFLECTED, VT_SCATTERED, VT_ABSORBED, VT_EXITED*); it expresses what has happened with the neutron at the intersection point.
In order not to slow down the simulation, it has been decided to use the parameter *bVisTraj* in the calling program, i.e. only to call *WriteWWP()*, if trajectories are requested:

```
if (bVisTraj) WriteWWP(pNeutron, eReason);
```

This can be written shorter using a macro (defined in init.h):

```
WriteIAP(pNeutron, eReason);
```

WriteWWP()

- writes a header in the first call
- counts the number of trajectories and limits them to MAX_TRAJ
- calculates the neutron position in the absolute co-ordinate system
- writes ID, color, wavelength, intensity loss, position, spin-state and reason into the file

The intensity loss is supposed to be used later on by another tool to estimate the radiation background.

WriteWWP() has to be called at each point of entering, exiting, reflection, scattering or absorption, and also if the entrance area is not hit. VT_PASSED is supposed to be used if entrance and exit point are identical (disc chopper, slit, etc.), VT_TRANSIT for the transition from one segment to the next (guide).
It is essential that the definition of the new co-ordinate system (Position[0]=0.0) is executed **after** writing all interaction points.

The output file looks like this:

```
#  ID       color  lambda      pos x      pos y      pos z spin reason
# ------------------------------------------------------------------
AA000000001     0  5.74683    350.000     -1.173      2.146  -1  2
AA000000002     0  4.14582    350.000      3.894     -0.990  -1  0
AA000000003     0  3.54095    350.000     -7.399     -4.174  -1  0
AA000000004     0  6.90066    350.000     -7.807     -0.058   1  0
AA000000005     0  2.64564    350.000     -5.606      1.265   1  0
AA000000006     0  7.83605    350.000      4.681     -6.721   1  0
AA000000007     0  1.35295    350.000      3.168      8.873   1  0
AA000000008     0  2.95180    350.000    -14.700     12.990   1  0
AA000000009     0  3.17460    350.000    -10.202     11.749  -1  0
AA000000010     0  7.05914    350.000     10.942     -2.399   1  0
```

The *traj-file* is closed again in *Cleanup()*. So far a dedicated trajectory treatment is realized for the modules Source, Slit, Chopper_disc, Guide, Elliptic_Guide, SM_Ensemle, Space_Window, Detector and Radial_Collimator. Apart from this, an interaction point is written to the trajectory file every time a neutron is read in the ReadNeutrons() function, thus also modules without a specific trajectory implementation contribute to the visualization.

The case of cloned trajectories

In ReadNeutrons() the case, in which a previous module has created a number of new trajectories with the same neutron ID, is treated by assigning new IDs to these neutrons. Independent of the way how other modules might assign new IDs to such cloned trajectories, in ReadNeutrons() the first letter of the ID string is incremented, the previous module number is added to the digits following the first two characters and the integer representing the trajectory number is changed in an unique way. The resulting output can look like this:

```
#  ID         color  lambda      pos x      pos y      pos z spin reason
# ------------------------------------------------------------------------
AA0000001111     3  1.51903   0.000e+00    3.91128   -1.75252   -0.01608  -1  4
BA1600001112     3  1.51903   0.000e+00    3.91098   -1.75320   -0.01703  -1  4
BA1600001113     3  1.51903   0.000e+00    3.91138   -1.75228   -0.01631  -1  4
BA1600001114     3  1.51903   0.000e+00    3.91145   -1.75211   -0.01606  -1  4
BA1600001115     3  1.51903   0.000e+00    3.91144   -1.75213   -0.01537  -1  4
BA1600001116     3  1.51903   0.000e+00    3.91030   -1.75227   -0.01475  -1  4
BA1600001117     3  1.51903   0.000e+00    3.91177   -1.75131   -0.01606  -1  4
BA1600001118     3  1.51903   0.000e+00    3.91236   -1.74968   -0.01278  -1  4
BA1600001119     3  1.51903   0.000e+00    3.91057   -1.75162   -0.01536  -1  4
BA1600001120     3  1.51903   0.000e+00    3.91075   -1.75117   -0.01577  -1  4
BA1600001121     4  1.51903   0.000e+00    3.91062   -1.75396   -0.01548  -1  4
BA1600001122     3  1.51903   0.000e+00    3.91108   -1.75298   -0.01608  -1  4
BA1400001112     3  1.51903   0.000e+00    3.86148   -1.76890   -0.08149  -1  4
CA3000001124     3  1.51903   0.000e+00    3.86028   -1.76817   -0.08590  -1  4
CA3000001125     3  1.51903   0.000e+00    3.86117   -1.76872   -0.08130  -1  4
CA3000001126     3  1.51903   0.000e+00    3.86424   -1.76925   -0.08254  -1  4
CA3000001127     3  1.51903   0.000e+00    3.85685   -1.76568   -0.08306  -1  4
CA3000001128     3  1.51903   0.000e+00    3.86238   -1.76827   -0.08165  -1  4
CA3000001129     3  1.51903   0.000e+00    3.85920   -1.76745   -0.07939  -1  4
CA3000001130     3  1.51903   0.000e+00    3.86109   -1.76867   -0.07968  -1  4
CA3000001131     3  1.51903   0.000e+00    3.86200   -1.76805   -0.08137  -1  4
CA3000001132     3  1.51903   0.000e+00    3.86188   -1.76798   -0.08103  -1  4
CA3000001133     4  1.51903   0.000e+00    3.86187   -1.76913   -0.08092  -1  4
CA3000001134     3  1.51903   0.000e+00    3.86167   -1.76901   -0.08196  -1  4
BA1400001113     3  1.51903   0.000e+00    3.91118   -1.72461   -0.02559  -1  4
CA3000001136     3  1.51903   0.000e+00    3.91110   -1.72444   -0.02569  -1  4
CA3000001137     3  1.51903   0.000e+00    3.91050   -1.72557   -0.02611  -1  4
CA3000001138     3  1.51903   0.000e+00    3.91022   -1.72493   -0.02727  -1  4
```

```
CA3000001139     3  1.51903   0.000e+00    3.91105   -1.72433   -0.02576  -1  4
CA3000001140     3  1.51903   0.000e+00    3.91035   -1.72521   -0.02509  -1  4
```