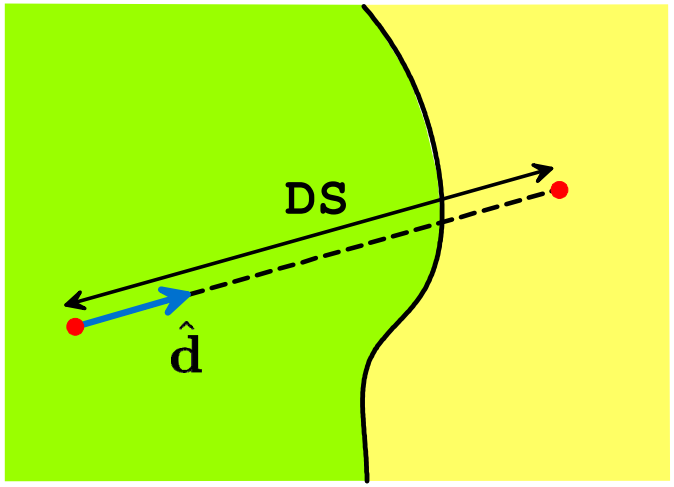
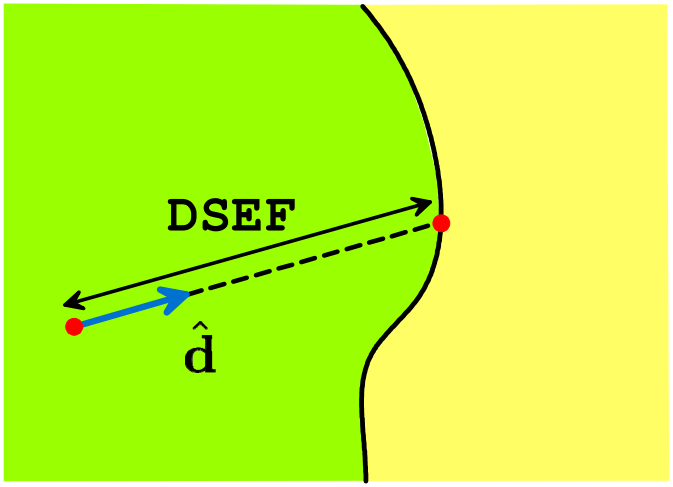
PENGEOM: Constructive quadric geometry

UNIVERSITAT DE BARCELONA

Tracking particles through complex geometries



Physics subroutines determine the distance that particles would travel in an infinite medium



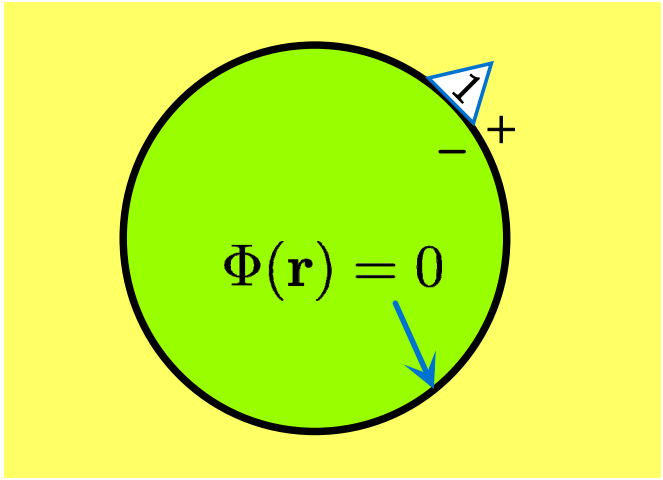
Geometry subroutines move the particle and stop it when the trajectory intersects an interface

- **The job of geometry subroutines:**
  1) Determine where (body and material) is the particle
  2) Move particles along straight trajectory segments
  3) Find intersections with the surfaces that limit the body
  4) Identify the body an material behind the interface

- **Bodies** are defined by their limiting surfaces

  Example: Sphere of radius R

  $$\Phi(\mathbf{r}) = x^2 + y^2 + z^2 - R^2 = 0$$



- **A surface divides space into three mutually exclusive regions:**
  1) $\Phi(\mathbf{r}) = 0$ → the surface itself (a set of null measure)
  2) $\Phi(\mathbf{r}) > 0$ → the "outside" of the surface, **side pointe SP=+1**
  3) $\Phi(\mathbf{r}) < 0$ → the "inside" of the surface, **side pointer SP=−1**

- **The side pointer of a region is not an absolute property:**
  The equation $-\Phi(\mathbf{r}) = 0$ defines the same surface but with the inner and outer regions exchanged

- **Intersections** of the (straight) trajectory of a particle, $\mathbf{r} = \mathbf{r}_0 + s\hat{\mathbf{d}}$ with a surface $\Phi(\mathbf{r}) = 0$ occurs after a path length s determined by the equation

  $$\Phi(\mathbf{r}_0 + s\hat{\mathbf{d}}) = 0$$

  This operation must be performed **many** times.

Quadric surfaces

- To make the tracking code efficient, we must limit the complexity of the surface functions
- A convenient solution is to use only **quadric surfaces** of the type

$$\Phi(x,y,z) = A_{xx}x^2 + A_{yy}y^2 + A_{zz}^2 + A_{xy}xy + A_{xz}xz + A_{yz}yz + A_x x + A_y y + A_z z + A_0 = 0$$

Path lengths to intersections are obtained by solving a quadratic eq.

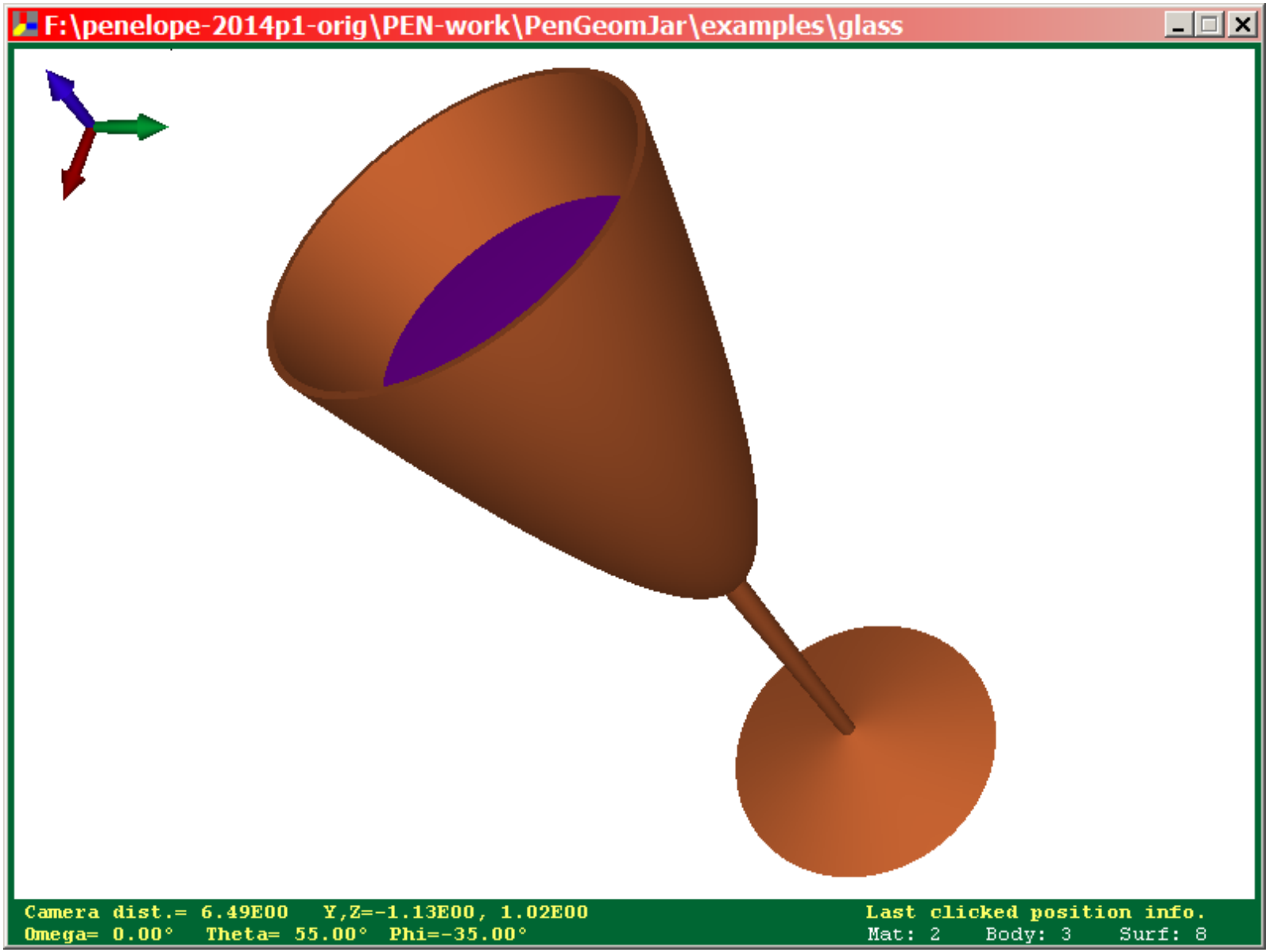
$$\Phi(\mathbf{r}_0 + s\hat{\mathbf{d}}) = 0 \quad \Rightarrow \quad as^2 + bs + d = 0$$

- Quadric surfaces allow a certain control of numerical round-off errors (fuzzy surfaces: shrink or swell slightly; effective protection against under- and over-shots)
- Some non-quadric surfaces give intersection equations that are also solvable analytically, but require much more CPU time. For instance, the torus

$$\left(\sqrt{x^2+y^2} - a\right)^2 + z^2 - b^2 = 0$$

gives a 4th order algebraic equation for the distance to the interface

- Quadric surfaces are versatile enough to describe many real objects … approximately



Definition of quadric surfaces

- **Implicit form:**

$$\Phi(x,y,z) = A_{xx}x^2 + A_{yy}y^2 + A_{zz}^2 + A_{xy}xy + A_{xz}xz + A_{yz}yz + A_x x + A_y y + A_z z + A_0 = 0$$

- **Reduced form:** Any quadric surface can be described by means of its primitive "standard" shape and the following **transformations**:
  - **Scaling** along the directions of the coordinate axes (the surface is expanded by factors a, b, c in the directions x, y, z)

$$x \to ax, \qquad y \to by, \qquad z \to cz$$

  - **Rotation** (Euler angles)

$$\mathbf{r} \to \mathcal{R}(\omega,\theta,\phi)\mathbf{r}$$

  - **Translation**

$$\mathbf{r} \to \mathbf{r} + \mathbf{a}$$

Applied in this order

- **Reduced quadric surfaces** are "standard" forms

Any quadric surface after appropriate translation, rotation, and scaling can be reduced to the form

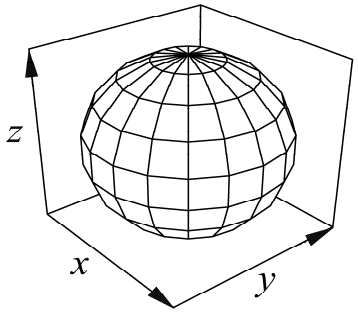
$$\Phi(x, y, z) = I_1 x^2 + I_2 y^2 + I_3 z^2 + I_4 z + I_5 = 0$$

where the indices I_i take the values $0, +1$, or -1

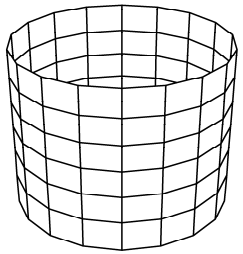
Reduced form	Indices					Quadric
$z - 1 = 0$	0	0	0	1	−1	plane
$z^2 - 1 = 0$	0	0	1	0	−1	pair of parallel planes
$x^2 + y^2 + z^2 - 1 = 0$	1	1	1	0	−1	sphere
$x^2 + y^2 - 1 = 0$	1	1	0	0	−1	cylinder
$x^2 - y^2 - 1 = 0$	1	−1	0	0	−1	hyperbolic cylinder
$x^2 + y^2 - z^2 = 0$	1	1	−1	0	0	cone
$x^2 + y^2 - z^2 - 1 = 0$	1	1	−1	0	−1	one sheet hyperboloid
$x^2 + y^2 - z^2 + 1 = 0$	1	1	−1	0	1	two sheet hyperboloid
$x^2 + y^2 - z = 0$	1	1	0	−1	0	paraboloid
$x^2 - z = 0$	1	0	0	−1	0	parabolic cylinder
$x^2 - y^2 - z = 0$	1	−1	0	−1	0	hyperbolic paraboloid

. . . and permutations of x and y.

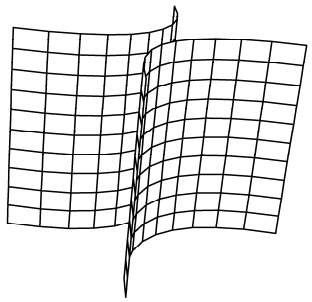
Non-planar reduced quadric surfaces



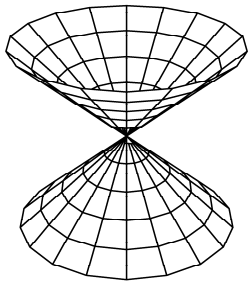
1, 1, 1, 0, −1
sphere



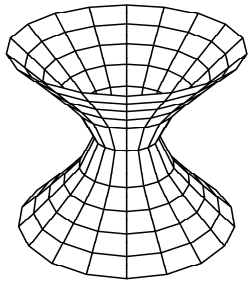
1, 1 , 0 , 0 , −1
cylinder



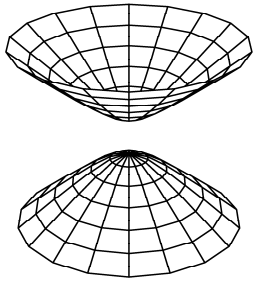
1, −1, 0, 0, −1
hyperbolic cylinder



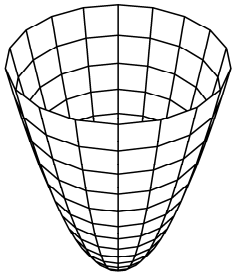
1, 1, −1, 0, 0
cone



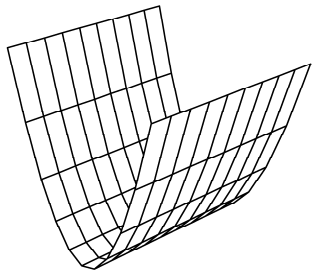
1, 1, −1, 0, −1
one sheet hyperboloid



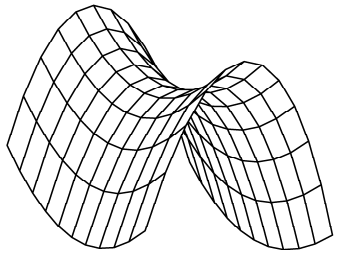
1, 1, −1, 0, 1
two sheet hyperboloid



1, 1, 0, −1, 0
paraboloid



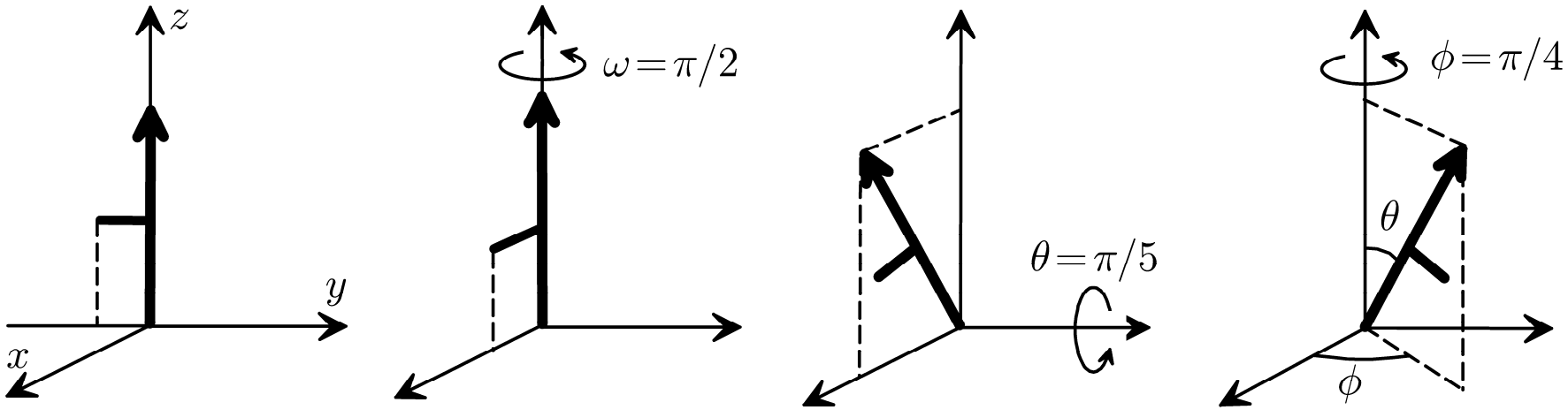
1, 0, 0, −1, 0
parabolic cylinder

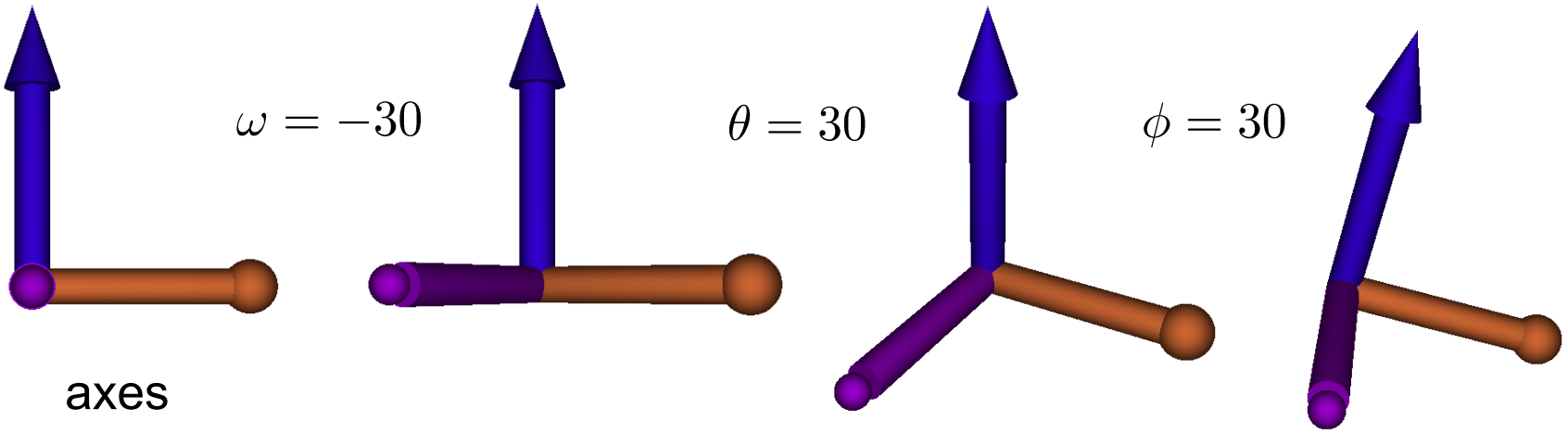


1, −1, 0, −1, 0
hyperbolic paraboloid

Rotations and Euler angles

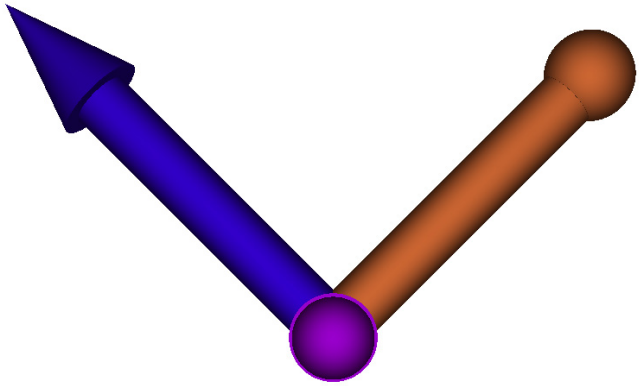
$$\mathcal{R}(\theta, \omega, \phi) = \mathcal{R}(\phi\hat{\mathbf{z}})\,\mathcal{R}(\theta\hat{\mathbf{y}})\,\mathcal{R}(\omega\hat{\mathbf{z}})$$



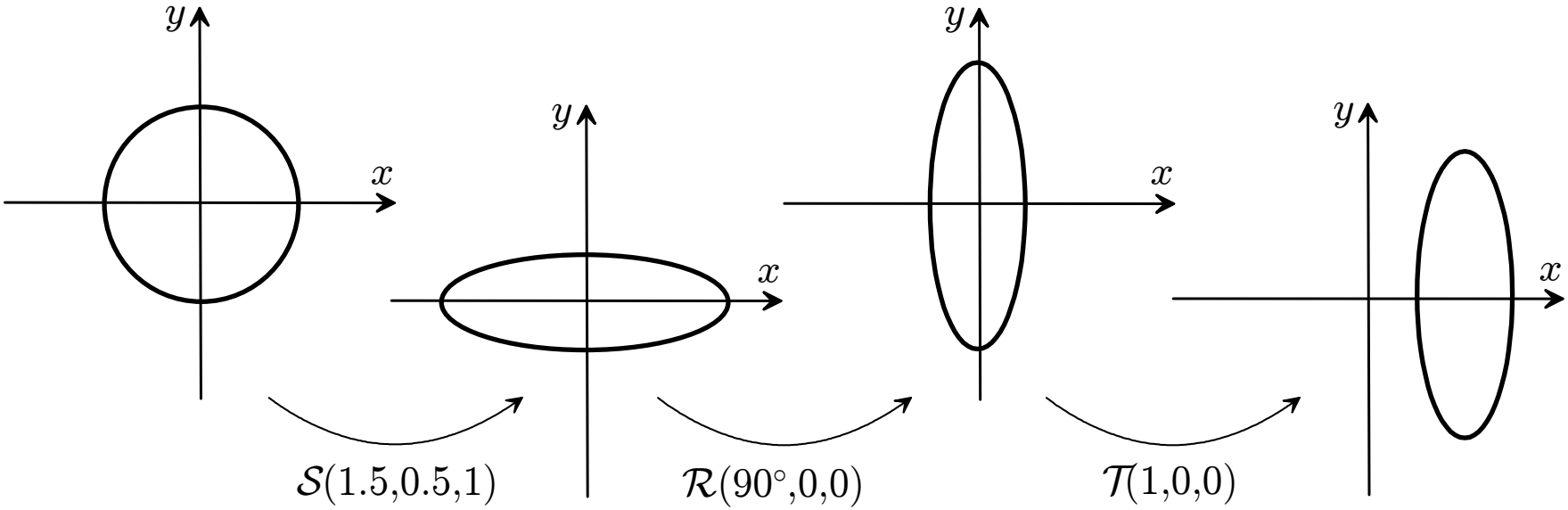


Rotation of 45° about the *x* axis:

$$R_x(\theta) = R_z(-90°)\ R_y(\theta)\ R_z(90°)$$



- **Complete definition process:**



Syntax

NOTE: The length unit is arbitrary! In PENELOPE all lengths are in cm

- **Surfaces in implicit form**

```
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (   1)   implicit form
INDICES=( 0, 0, 0, 0, 0)
    AXX=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
    AXY=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
    AXZ=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
    AYY=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
    AYZ=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
    AZZ=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
     AX=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
     AY=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
     AZ=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
     A0=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
1111111111111111111111111111111111111111111111111111111111111111
  OMEGA=(+0.00000000000000E+00,   0) DEG             (DEFAULT=0.0)
  THETA=(+0.00000000000000E+00,   0) DEG             (DEFAULT=0.0)
    PHI=(+0.00000000000000E+00,   0) RAD             (DEFAULT=0.0)
X-SHIFT=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
Y-SHIFT=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
Z-SHIFT=(+0.00000000000000E+00,   0)                (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000
```

- Lines defining default values can be removed from the definition file

Syntax

- **Surfaces in reduced form**

```
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (   1)   reduced form
INDICES=( 1, 1, 1, 1, 1)
X-SCALE=(+1.0000000000000E+00,   0)                (DEFAULT=1.0)
Y-SCALE=(+1.0000000000000E+00,   0)                (DEFAULT=1.0)
Z-SCALE=(+1.0000000000000E+00,   0)                (DEFAULT=1.0)
  OMEGA=(+0.0000000000000E+00,   0) DEG            (DEFAULT=0.0)
  THETA=(+0.0000000000000E+00,   0) DEG            (DEFAULT=0.0)
    PHI=(+0.0000000000000E+00,   0) RAD            (DEFAULT=0.0)
X-SHIFT=(+0.0000000000000E+00,   0)                (DEFAULT=0.0)
Y-SHIFT=(+0.0000000000000E+00,   0)                (DEFAULT=0.0)
Z-SHIFT=(+0.0000000000000E+00,   0)                (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000
```

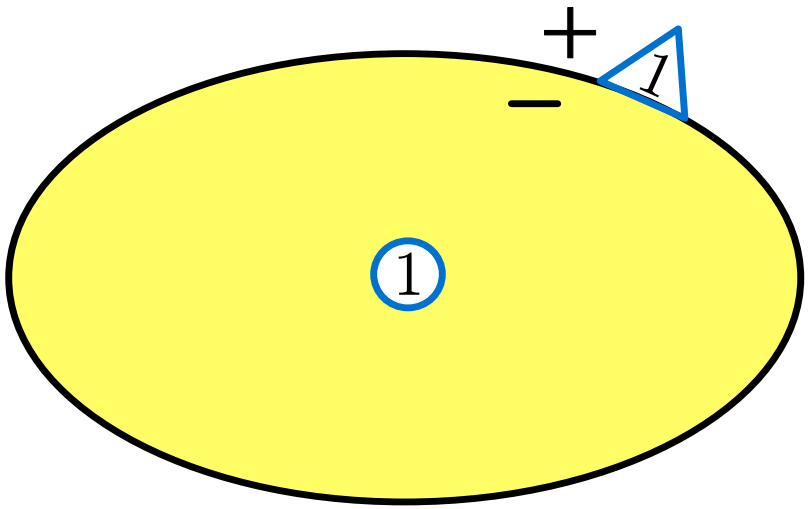
- Rotations are described by means of the **Euler angles**
- Lines defining default values can be removed from the definition file

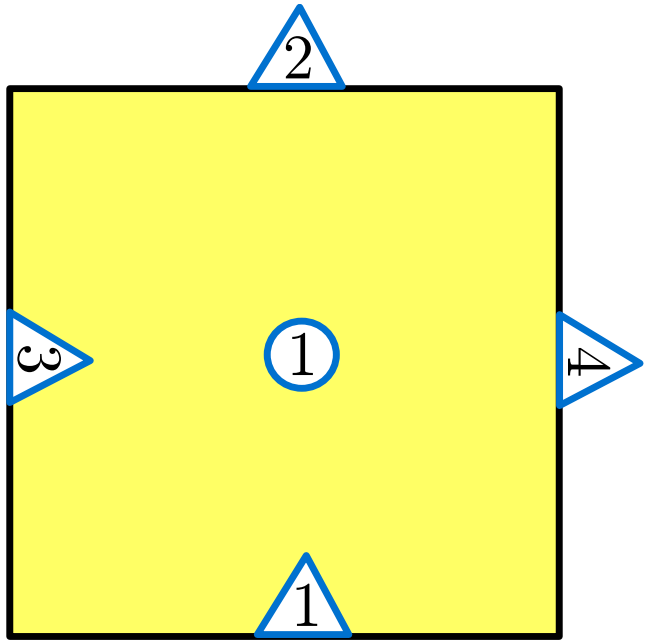
NOTE: The integer label following each parameter

```
X-SCALE=(+1.0000000000000E+00,   0)                (DEFAULT=1.0)
```

allows the parameter value to be modified from the main program (selected geometry elements can be made mobile)

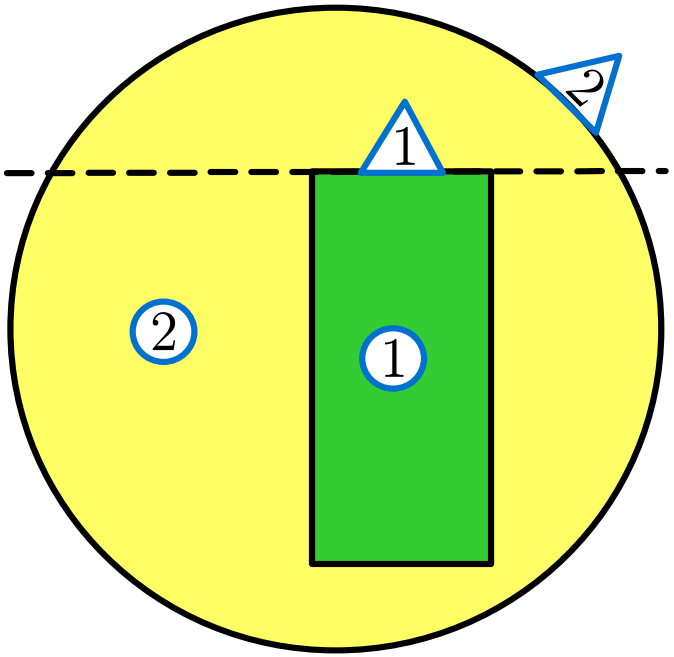
- Definition of bodies

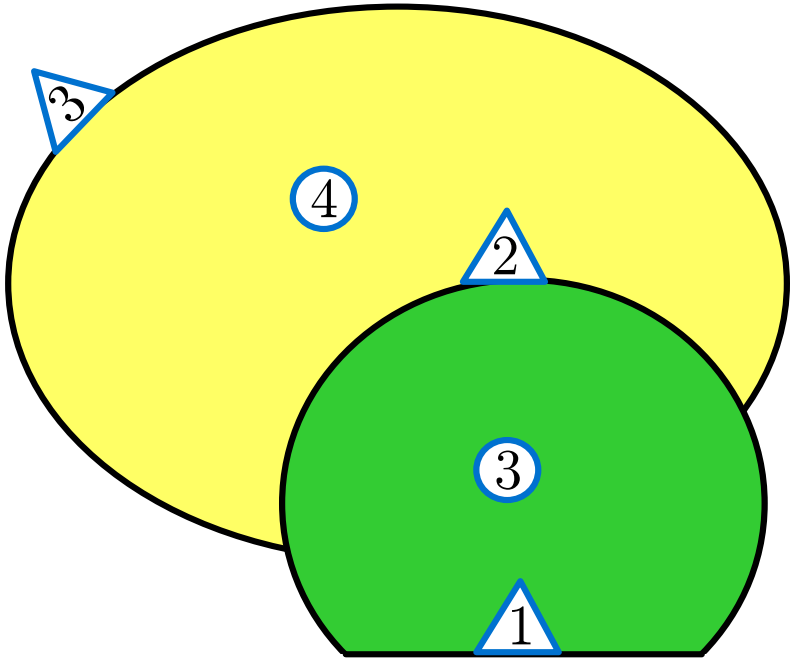





```
0000000000000000000000000000000000000000000000000000000000000000
BODY     (    1)    text
MATERIAL(    1)
SURFACE  (    1), SIDE POINTER=( 1)
SURFACE  (    2), SIDE POINTER=(-1)
SURFACE  (    3), SIDE POINTER=( 1)
SURFACE  (    4), SIDE POINTER=(-1)...
0000000000000000000000000000000000000000000000000000000000000000
```

Constructive geometry. Bodies and modules





Some bodies can not be defined with only surfaces

... but avoid using limiting bodies if you can do with only surfaces

```
0000000000000000000000000000000000000000000000000000000000000000
BODY     (    2)    text
MATERIAL(    2)
SURFACE  (    2), SIDE POINTER=(-1)
BODY     (    1)
0000000000000000000000000000000000000000000000000000000000000000
```

Constructive geometry. Bodies and modules

- **Module** ≡ Connected volume limited only by quadrics

  -- The particle can enter or leave the module only through the limiting surfaces (this is not true for bodies)

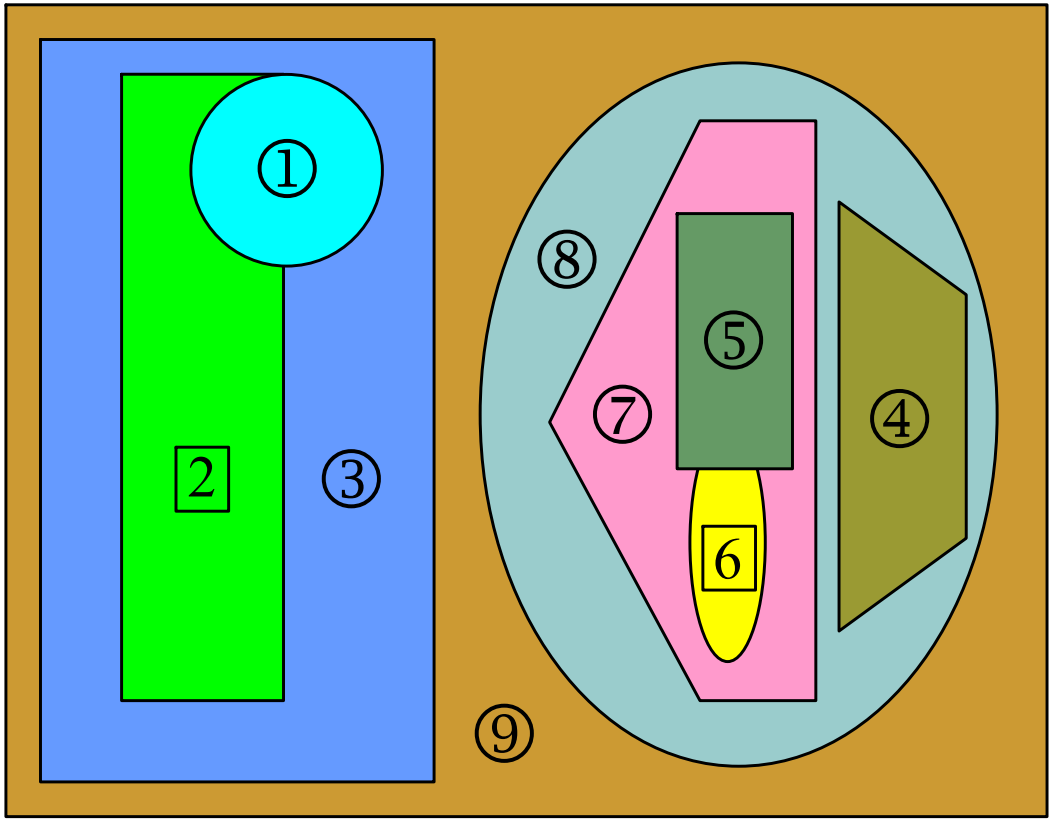
  -- A module may contain bodies and other modules (daughters). These have to be **fully contained** in the parent module
  The "cavities" of a module can be filled with a given material

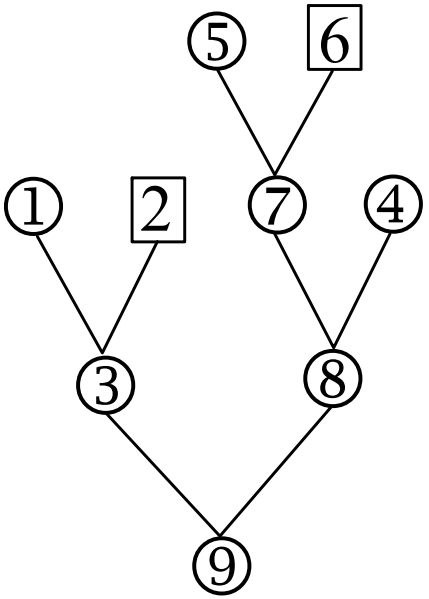
  -- A module cannot overlap other modules or bodies other than its daughters

  -- A module can be rotated/translated as a whole

```
0000000000000000000000000000000000000000000000000000000000000000
MODULE   (   3)    text
MATERIAL(   2)
SURFACE  (   5), SIDE POINTER=( 1)
BODY     (   1)
MODULE   (   2)
1111111111111111111111111111111111111111111111111111111111111111
  OMEGA=(+0.000000000000000E+00,   0) DEG           (DEFAULT=0.0)
  THETA=(+0.000000000000000E+00,   0) DEG           (DEFAULT=0.0)
    PHI=(+0.000000000000000E+00,   0) RAD           (DEFAULT=0.0)
X-SHIFT=(+0.000000000000000E+00,   0)               (DEFAULT=0.0)
Y-SHIFT=(+0.000000000000000E+00,   0)               (DEFAULT=0.0)
Z-SHIFT=(+0.000000000000000E+00,   0)               (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000
```

- **Genealogical tree:**



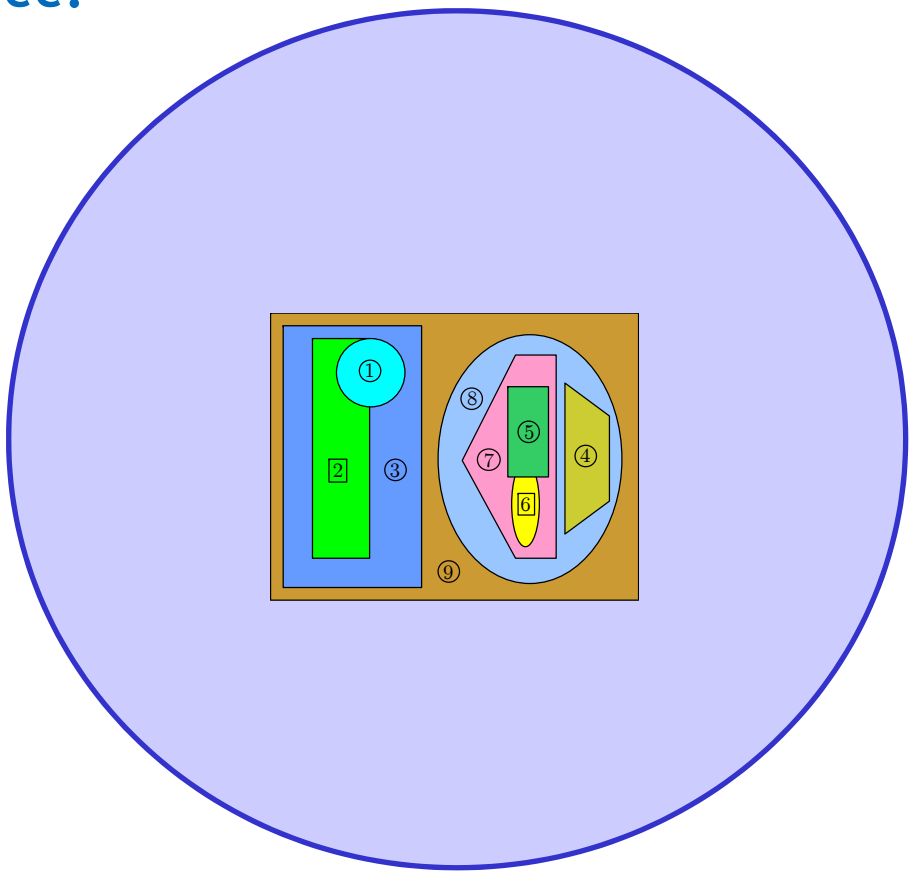


○ → modules (may have descendants, or not)

□ → bodies

To speed up the simulation, each module should have a small number of daughters (small number of branches in each node).
With a properly defined genealogical tree, simulation speed is almost independent of the complexity of the geometry.

- **Genealogical tree:**



Enclosure: The root module, must be convex.

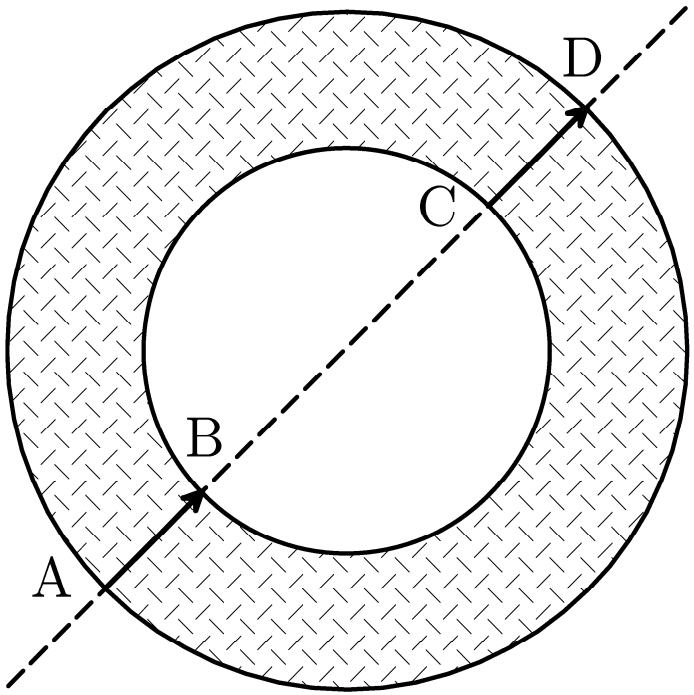
If a root module is not defined (i.e., if there are motherless bodies or multiple modules) the whole geometry is assumed to be contained within a sphere of 10^7 units radius (root module).

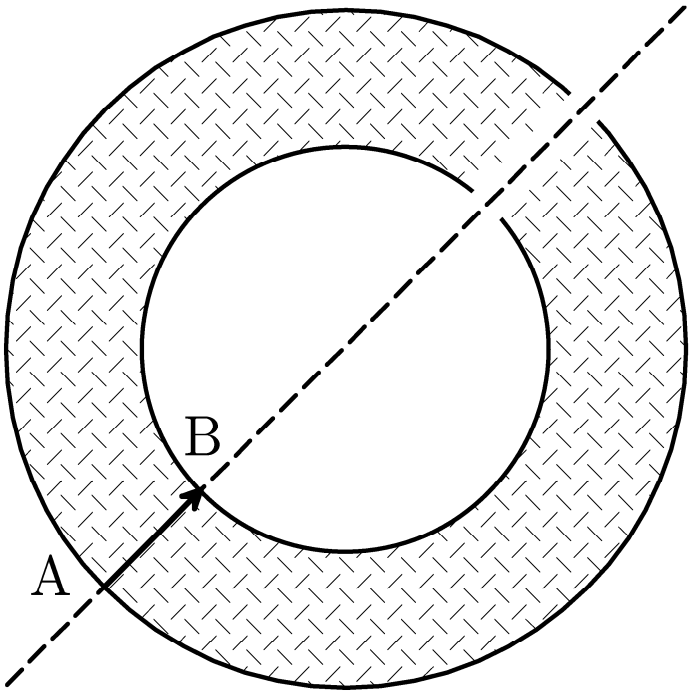
Bodies that extend beyond the enclosure are “truncated”

- **Genealogical tree:**
  -- Particles that exit through a limiting surface of the root module are assumed to leave the system

The **root module must be convex**, because

The program will not work properly if the root module is concave:





The root module may be infinite (e.g. a planar foil)

Other geometry options

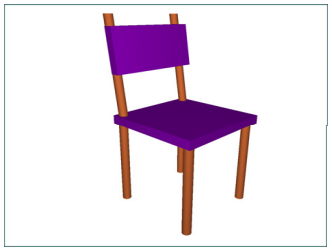
- **CLONE:** copies one module and moves it
  (very useful to define repetitive structures)

```
0000000000000000000000000000000000000000000000000000000000000000
CLONE    ( yyy)    copies one module and moves it
MODULE   ( xxx)
1111111111111111111111111111111111111111111111111111111111111111
  OMEGA=(+0.0000000000000E+00,    0) DEG           (DEFAULT=0.0)
  THETA=(+0.0000000000000E+00,    0) DEG           (DEFAULT=0.0)
    PHI=(+0.0000000000000E+00,    0) RAD           (DEFAULT=0.0)
X-SHIFT=(+0.0000000000000E+00,    0)               (DEFAULT=0.0)
Y-SHIFT=(+0.0000000000000E+00,    0)               (DEFAULT=0.0)
Z-SHIFT=(+0.0000000000000E+00,    0)               (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000
```

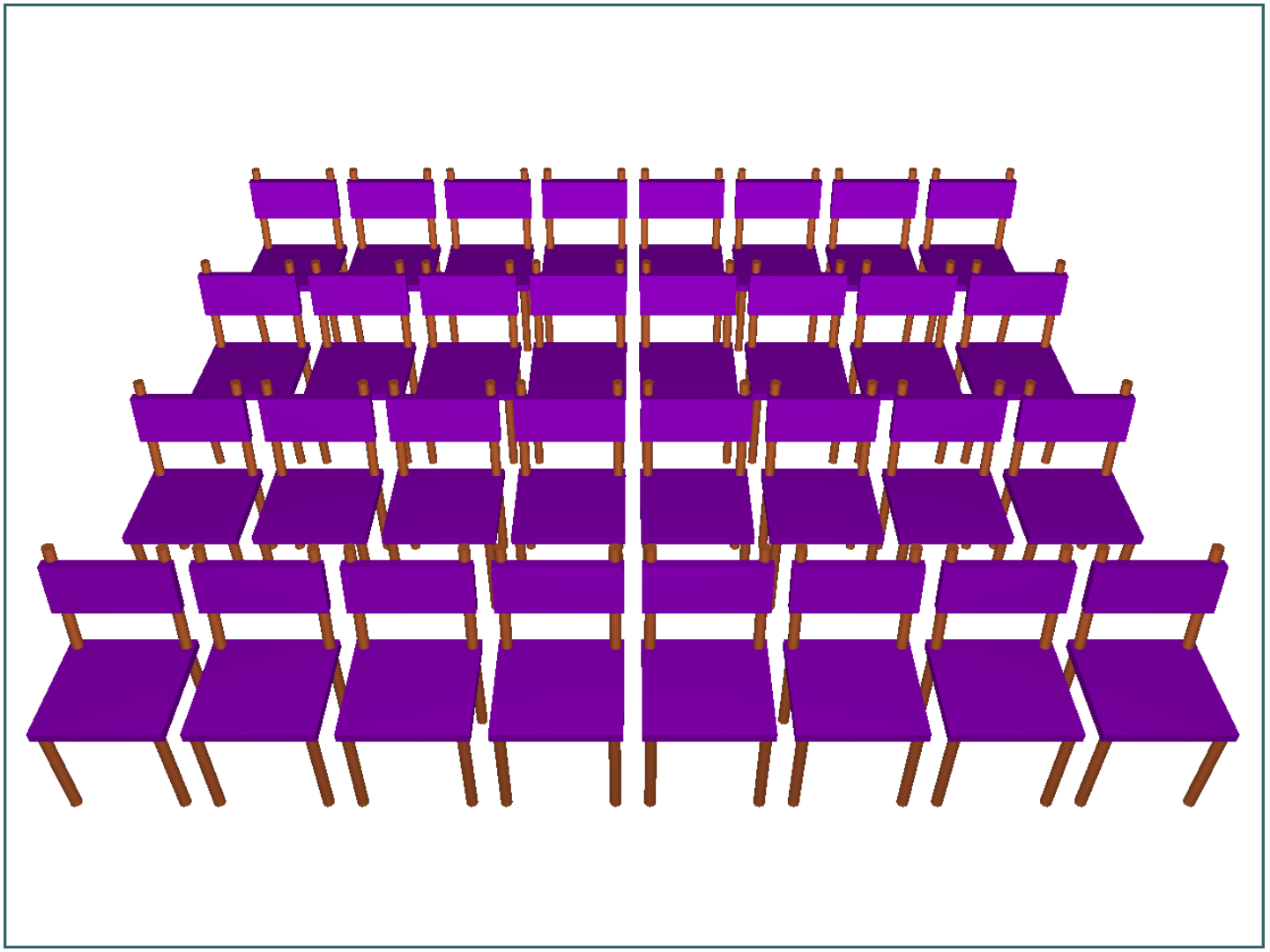
- **INCLUDE:** inserts an existing geometry definition file
  (useful to split a long geometry definition file into smaller pieces)

```
0000000000000000000000000000000000000000000000000000000000000000
INCLUDE
FILE=    (filename.ext)
0000000000000000000000000000000000000000000000000000000000000000
```

Repeated cloning



After cloning five times...



Fortran module PENGEOM_mod

```
      MODULE PENGEOM_mod
C  ****  Geometry definition parameters and I/O quantities.
      SAVE  ! Saves all items in the module.
C  ----  Geometry array sizes.
C        Maximum numbers of surfaces, bodies, and limiting elements.
      INTEGER*4, PARAMETER :: NS=10000, NB=5000, NXG=250
C        Number of bodies in the material system (given by PENGEOM).
      INTEGER*4 :: NBODY
C  ----  Body aliases (user labels).
      CHARACTER*4 :: BALIAS(NB)='    '
C  ----  Body materials. MATER(KB) is the material in body KB.
      INTEGER*4 :: MATER(NB)=0
C  ----  Detector definition.
C        KDET(KB)=ID if body KB is part of detector ID.
      INTEGER*4 :: KDET(NB)=0
C  ****  Warning messages for accidental undershots or round-off errors
C        are issued when LVERB=.TRUE.
      LOGICAL :: LVERB=.FALSE.
C  ****  Last step features (output from subroutine STEP).
C  ----  Travelled path length, including segments in void volumes.
      DOUBLE PRECISION :: DSTOT
C  ----  Label of the last surface crossed by the particle before
C        entering a material body (defined only when NCROSS /= 0).
      INTEGER*4 :: KSLAST
      END MODULE PENGEOM_mod
```

The subroutine package PENGEOM

User-callable routines:

```
SUBROUTINE GEOMIN(PARINP,NPINP,NMAT,NBOD,IRD,IWR)
```

Reads geometry data from the input file, initializes the geometry package and prints the file `geometry.rep`

```
SUBROUTINE LOCATE
```

Determines the body **IBODY** that contains a point **(X,Y,Z)** and its material **MAT** (values delivered though module **TRACK_mod**)

```
SUBROUTINE STEP(DS,DSEF,NCROSS)
```

Performs the geometrical part of the track simulation. Moves the particle and changes body and material numbers as necessary. New values of the variables **X,Y,Z,IBODY,MAT** are delivered though **TRACK_mod**

DS	...	(input) path length to travel
DSEF	...	(output) travelled path length before leaving the initial material or completing the jump
NCROSS	...	(output) number of interfaces crossed

Impact detectors

SUBROUTINE STEP does not stop particles when they cross an interface between adjacent bodies of the same composition. That is, these **interfaces are not visible from the main program**

KDET(NB) in module **PENGEOM_mod**

KDET(KB) is initialised to zero. Body **KB** is made part of impact detector **IDET** by setting **KDET(KB)=IDET** (in main program). When particles enter body **KB** coming from a body that is not part of detector **IDET**, they are halted at the surface of the active body and control is returned to the main program

Thus, detector surfaces are made visible from the main program

Impact detectors are useful for visualising bodies (**GVIEW2D**), for simulating real detectors and for generating phase-space files (**PENMAIN**)

Making geometry parts "mobile"

Input values of pre-selected geometry parameters can be modified from the main program

```
SUBROUTINE GEOMIN(PARINP,NPINP,NMAT,NBOD,IRD,IWR)
DIMENSION PARINP(NPAR)        NPINP ≤ NPAR
```

Example: The input file element

```
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (   1)   sphere, R=1, centered at (0,0,0)
INDICES=( 1, 1, 1, 0,-1)
X-SHIFT=(+0.0000000000000000E+00,   1)                (DEFAULT=0.0)
Y-SHIFT=(+0.0000000000000000E+00,   2)                (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000
```

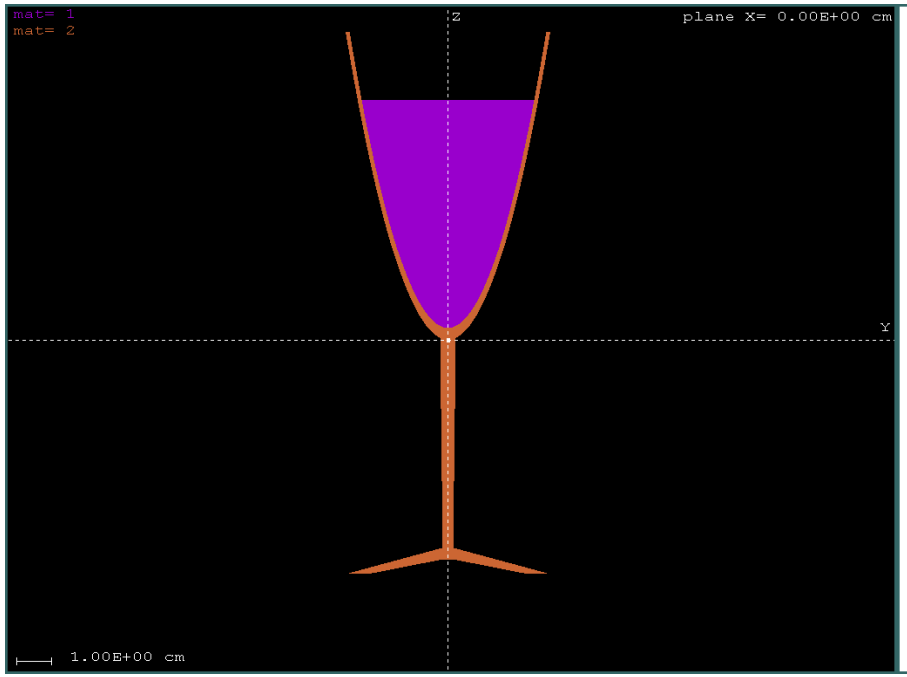
with **NPINP=2, PARINP(1)=1., PARINP(2)=3.**
is transformed (internally) into

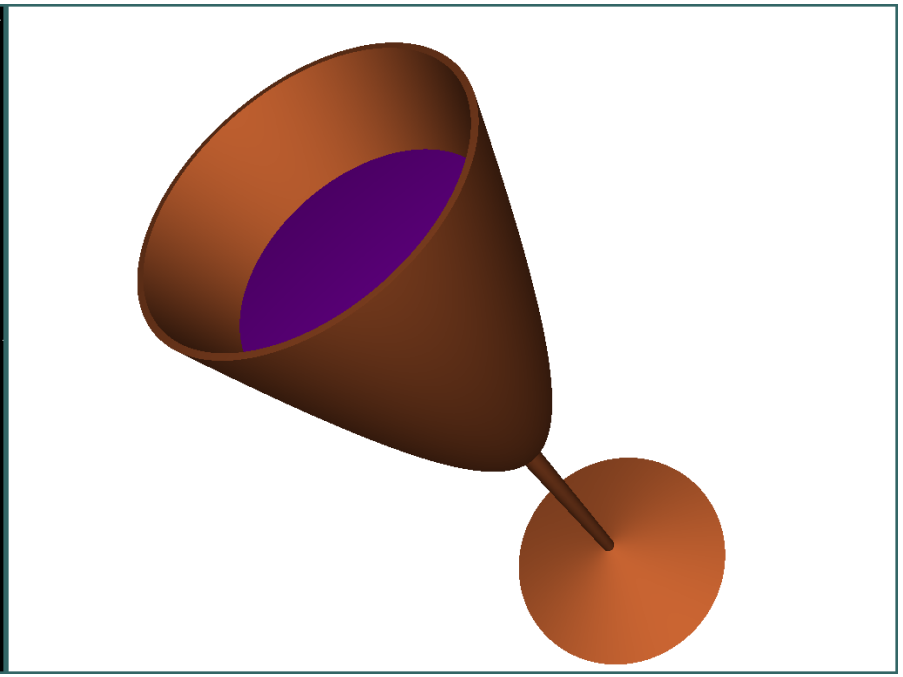
```
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (   1)   sphere, R=1, centered at (1,3,0)
INDICES=( 1, 1, 1, 0,-1)
X-SHIFT=(+1.0000000000000000E+00,   0)                (DEFAULT=0.0)
Y-SHIFT=(+3.0000000000000000E+00,   0)                (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000
```

PENELOPE geometry viewers

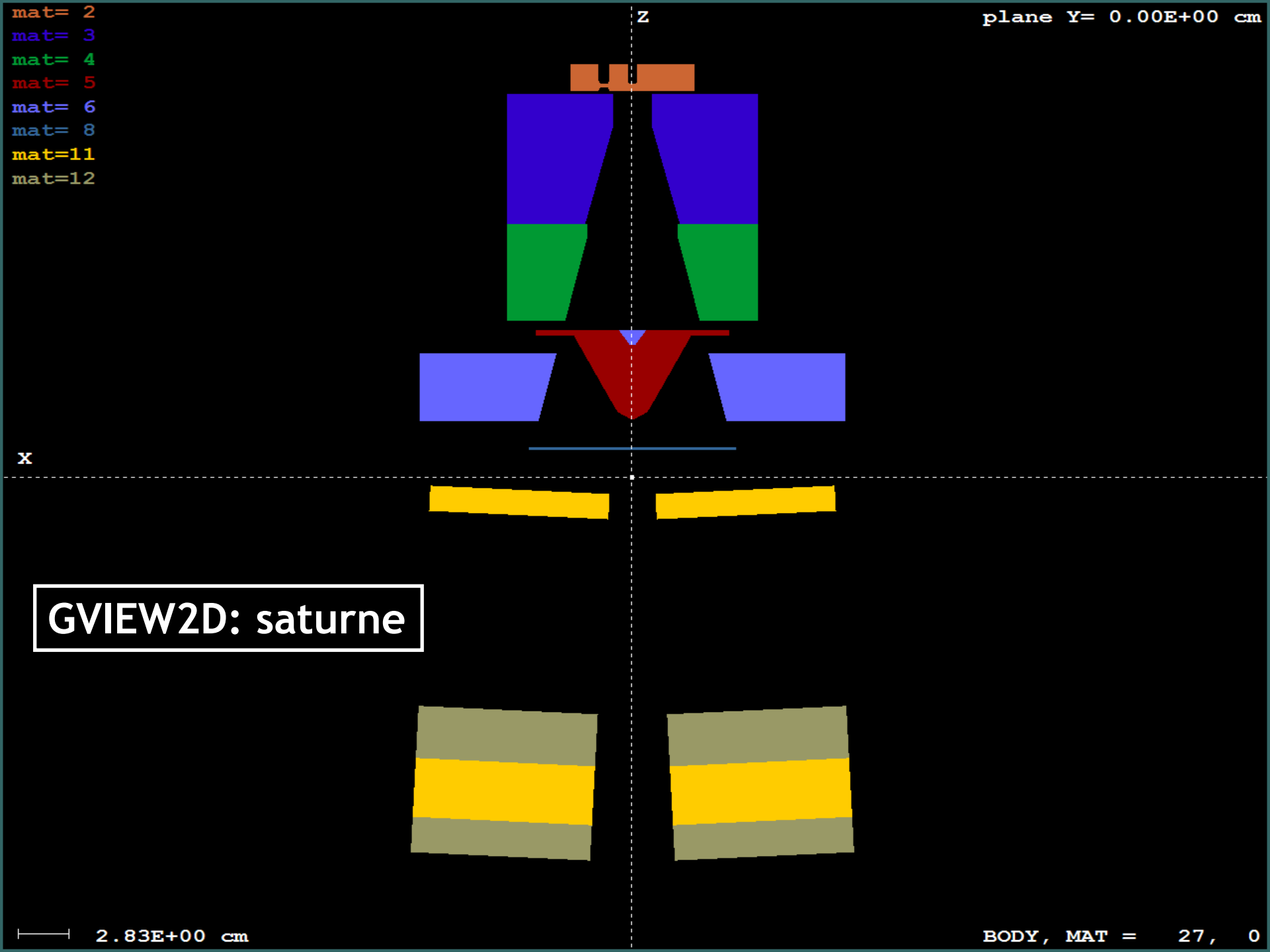
- **GVIEW2D** two-dimensional (material and body maps)
- **GVIEW3D** three-dimensional images (quite realistic)

Generated with Compaq Visual Fortran, run only under MS Windows.

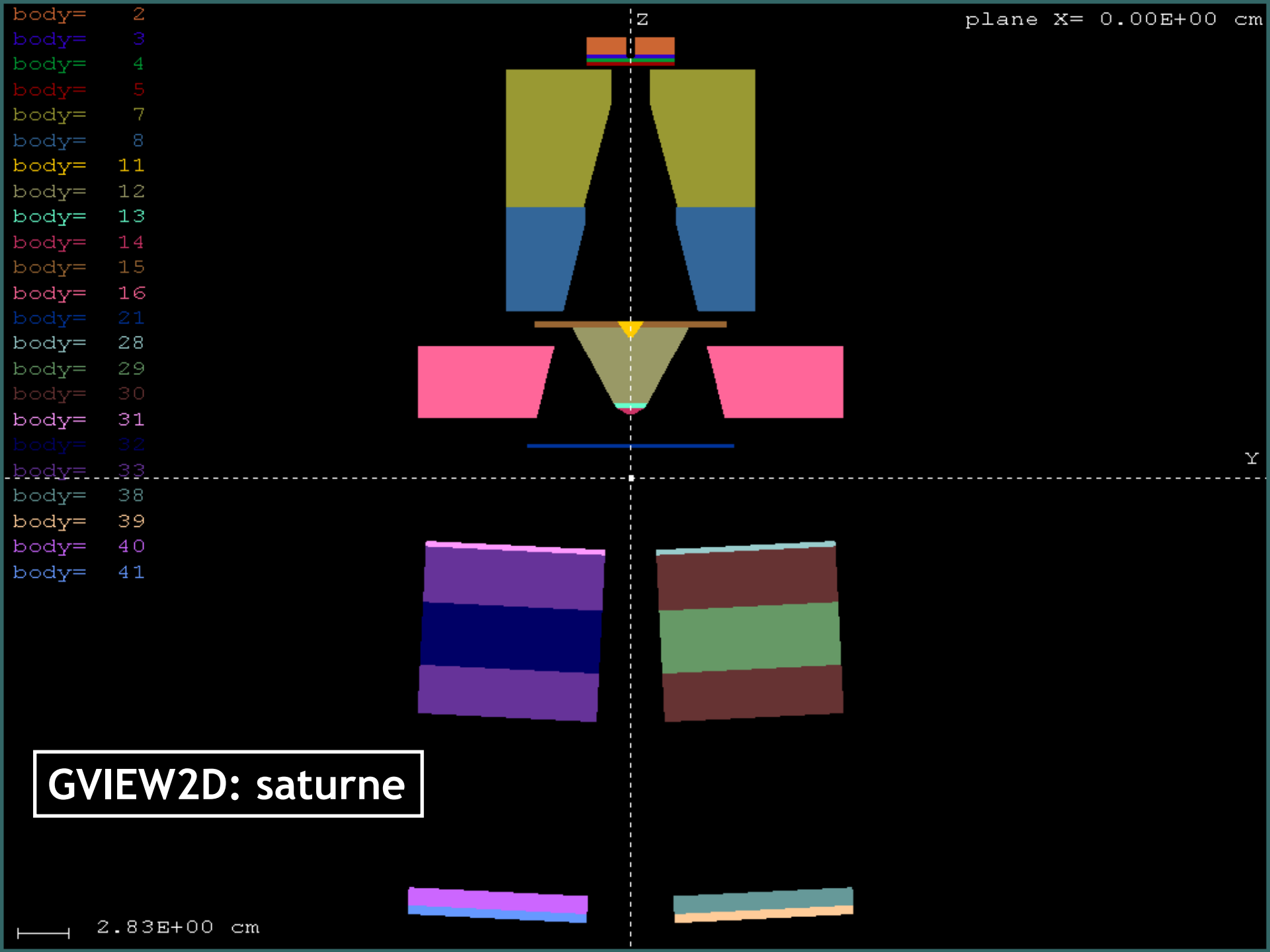




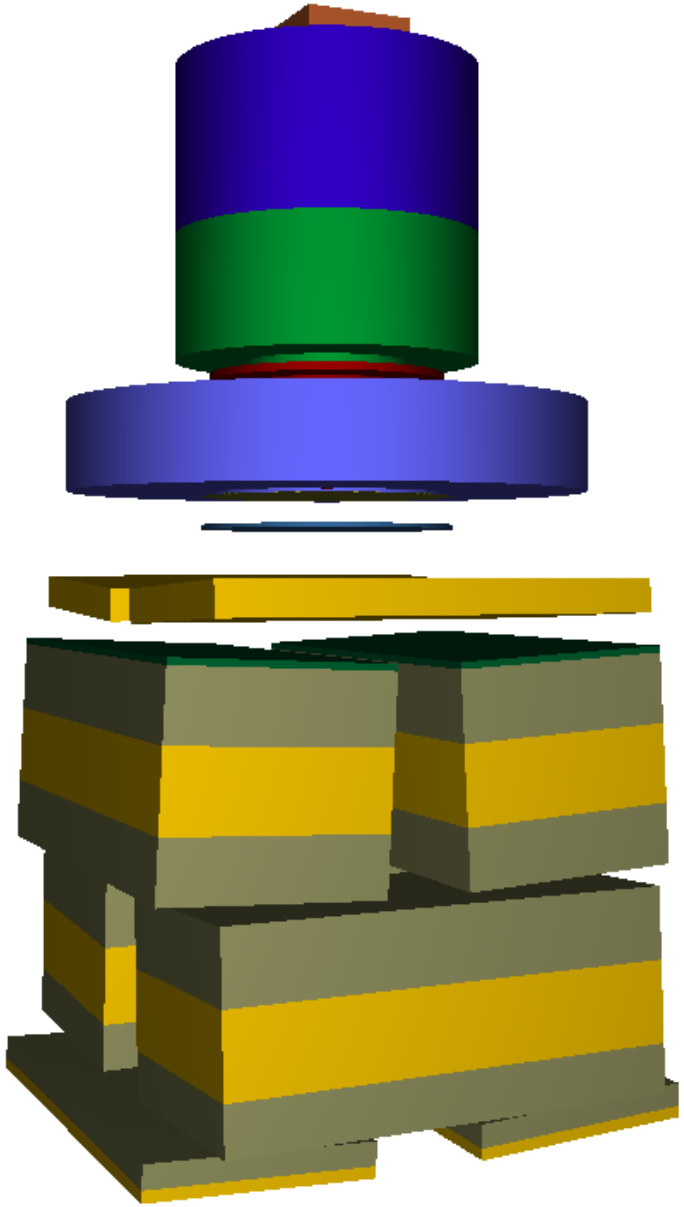
Images are rendered by using the **PENGEOM** tracking subroutines
➔ what we see is what is passed to the simulation code

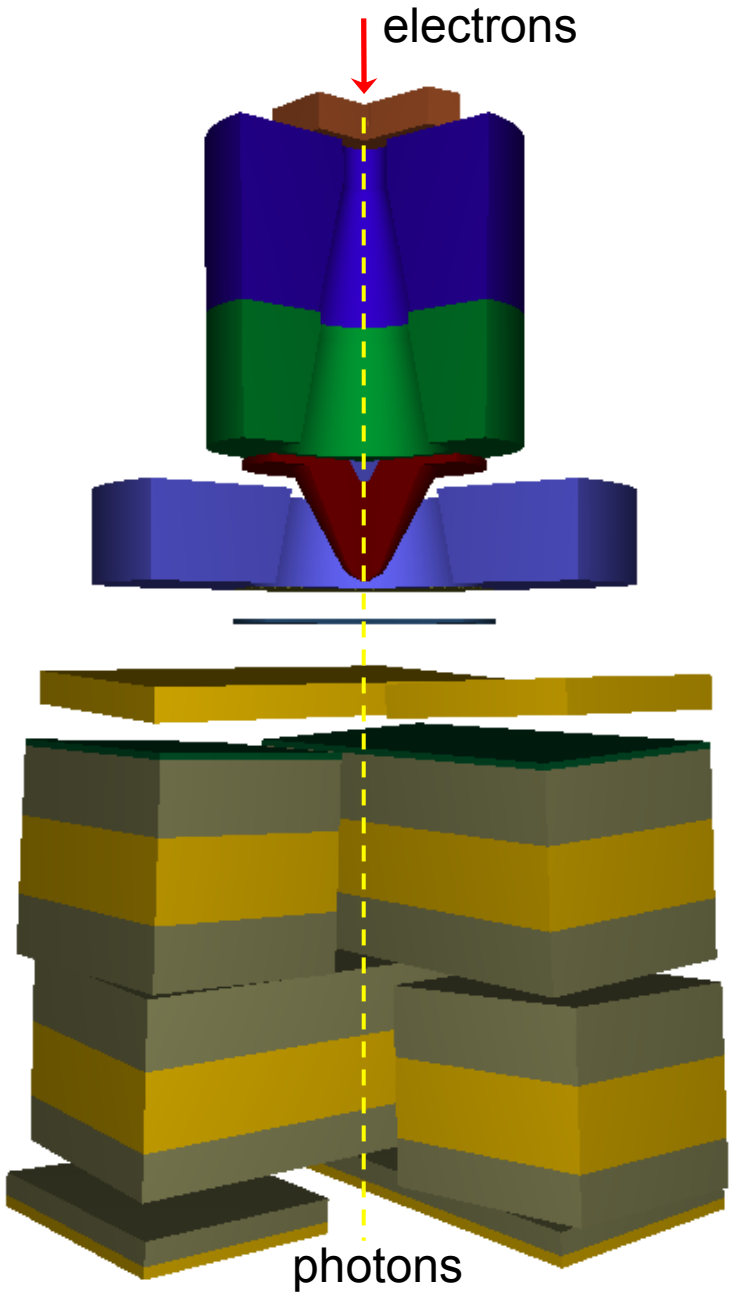


GVIEW2D: saturne


body= 2
body= 3
body= 4
body= 5
body= 7
body= 8
body= 11
body= 12
body= 13
body= 14
body= 15
body= 16
body= 21
body= 28
body= 29
body= 30
body= 31
body= 32
body= 33
body= 38
body= 39
body= 40
body= 41
Z
Y
plane X= 0.00E+00 cm
2.83E+00 cm
GVIEW2D: saturne

GVIEW3D: saturne





Debugging

- **GVIEW** generates a debugging file named "**geometry.rep**", which contains a duplicate of the geometry definition file followed by details on the genealogical tree structure

- **GVIEW** stops when a syntax error or an obvious inconsistency is found. The offending input datum usually appears in the last printed lines of the debugging file

- Errors or inconsistencies in the definition of the geometry are usually identified by visual inspection of the images created by **GVIEW**

WARNING: **PENGEOM** assigns its own **labels** to all geometry elements. To specify a body or module, we need to know its "internal" label

- The surface and body labels used by **PENGEOM** during simulation are printed in file "**geometry.rep**". The body labels can also be identified by running **GVIEW2D** in mode bodies

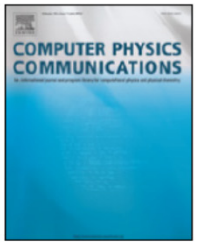
Editing and debugging geometry files: PenGeomJar

- Geometry definition files can be edited and debugged by using the Java GUI **PenGeomJar**, which is distributed by the OCDE NEA as a separate package. Also available from the CPC Program Library.

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

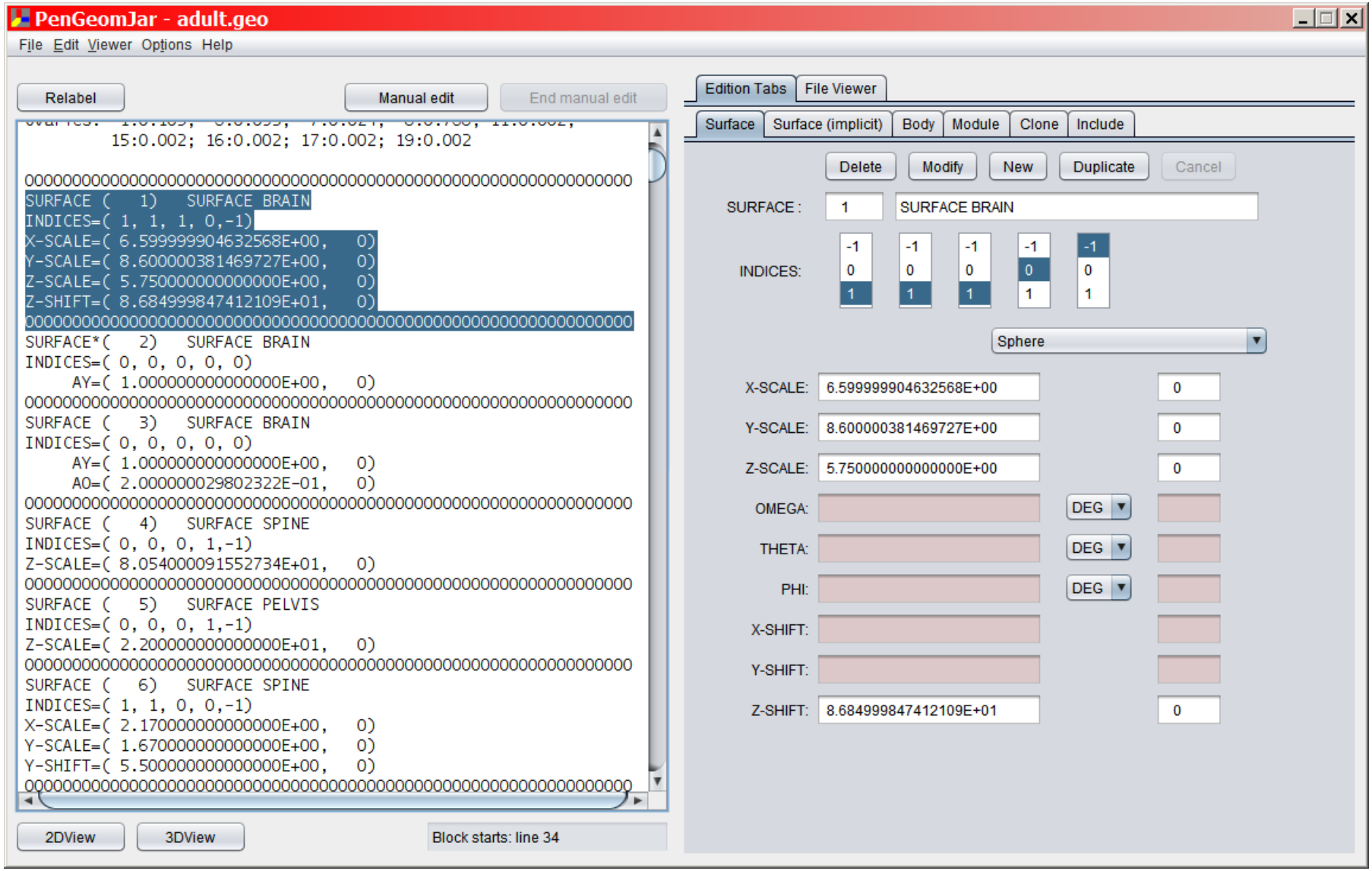


PENGEOM—A general-purpose geometry package for Monte Carlo simulation of radiation transport in material systems defined by quadric surfaces☆

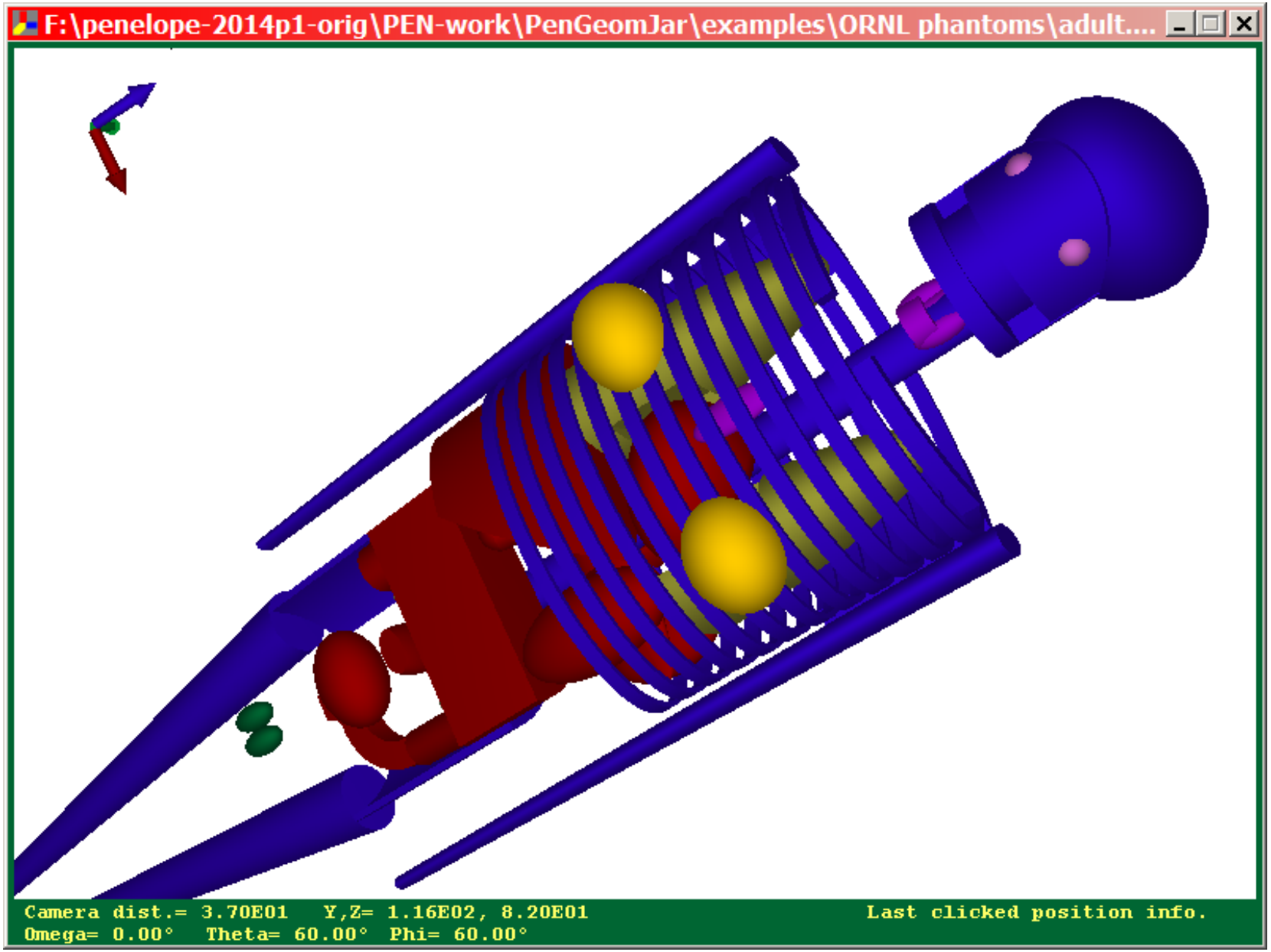
Julio Almansa[a], Francesc Salvat-Pujol[b], Gloria Díaz-Londoño[c], Artur Carnicer[d], Antonio M. Lallena[e], Francesc Salvat[d,*]

- It contains 2D and 3D viewers that operate in the same way as the codes **gview2d** and **gview3d**. Versions are provided for Windows, Linux and generic UNIX platforms

PenGeomJar


PenGeomJar - adult.geo
File Edit Viewer Options Help
Relabel
Manual edit
End manual edit
15:0.002; 16:0.002; 17:0.002; 19:0.002
SURFACE (1) SURFACE BRAIN
INDICES=(1, 1, 1, 0,-1)
X-SCALE=(6.599999904632568E+00, 0)
Y-SCALE=(8.600000381469727E+00, 0)
Z-SCALE=(5.750000000000000E+00, 0)
Z-SHIFT=(8.684999847412109E+01, 0)
SURFACE*(2) SURFACE BRAIN
INDICES=(0, 0, 0, 0, 0)
AY=(1.000000000000000E+00, 0)
SURFACE (3) SURFACE BRAIN
INDICES=(0, 0, 0, 0, 0)
AY=(1.000000000000000E+00, 0)
AO=(2.000000029802322E-01, 0)
SURFACE (4) SURFACE SPINE
INDICES=(0, 0, 0, 1,-1)
Z-SCALE=(8.054000091552734E+01, 0)
SURFACE (5) SURFACE PELVIS
INDICES=(0, 0, 0, 1,-1)
Z-SCALE=(2.200000000000000E+01, 0)
SURFACE (6) SURFACE SPINE
INDICES=(1, 1, 0, 0,-1)
X-SCALE=(2.170000000000000E+00, 0)
Y-SCALE=(1.670000000000000E+00, 0)
Y-SHIFT=(5.500000000000000E+00, 0)
2DView
3DView
Block starts: line 34
Edition Tabs
File Viewer
Surface
Surface (implicit)
Body
Module
Clone
Include
Delete
Modify
New
Duplicate
Cancel
SURFACE :
1
SURFACE BRAIN
INDICES:
Sphere
X-SCALE:
6.599999904632568E+00
Y-SCALE:
8.600000381469727E+00
Z-SCALE:
5.750000000000000E+00
OMEGA:
DEG
THETA:
DEG
PHI:
DEG
X-SHIFT:
Y-SHIFT:
Z-SHIFT:
8.684999847412109E+01

Java GUI PenGeomJar



Limitations of PENGEOM

- **PENGEOM** uses fuzzy quadric surfaces (i.e., surfaces that swell or shrink when a particle crosses them), thus reducing the effect of round-off errors. Avoid “touching” bodies

- However, the effective resolution is finite: small structures at large distances from the origin may not be properly resolved. Use the viewers to make sure that the geometry is described correctly

- **PENGEOM** admits up to 10,000 surfaces and 5000 bodies (and modules). The maximum number of bodies in a module or the number of surfaces defining a body is set to 250. These parameters are defined in module **PENGEOM_mod** can be changed by editing the source file **pengeom.f**

- Note that the speed of the simulation depends strongly on the structure of the genealogical tree. **The responsibility of setting it appropriately rests with the user**