연세대학교
YONSEI UNIVERSITY

Medical Example:

Computational Human Phantoms

Yeon Soo Yeom

Presenter: Sungho Moon

12th International Geant4 Tutorial in Korea 2025

Feb. 3-7, 2025@ Pohang Accelerator Laboratory, Pohang

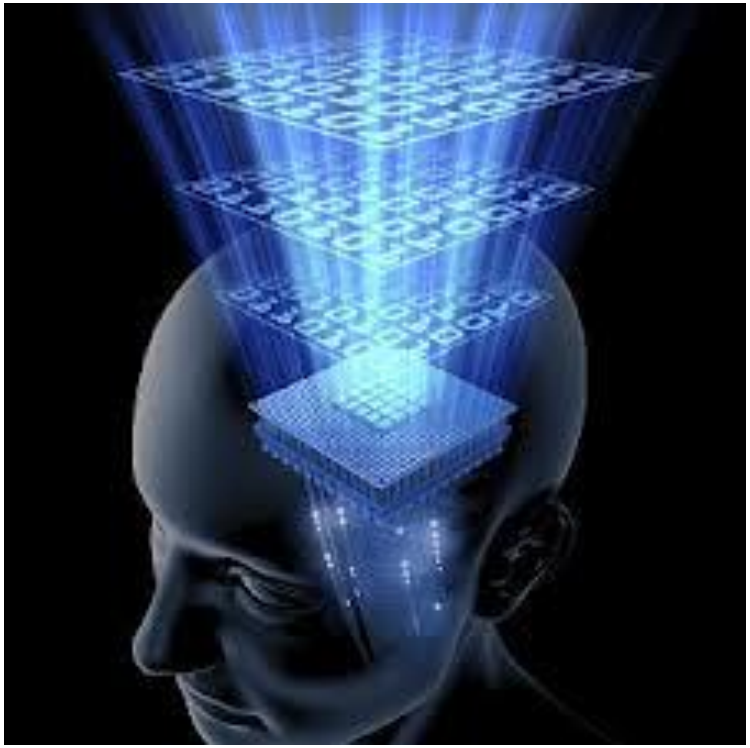
12th International Geant4 Tutorial in Korea 2025

Date 3-7 Feb 2025

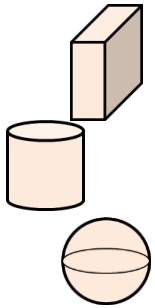
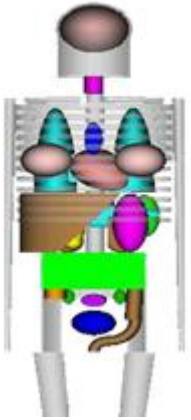
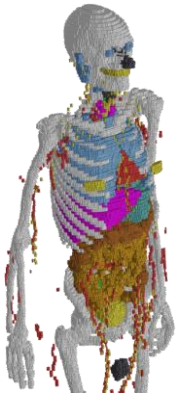
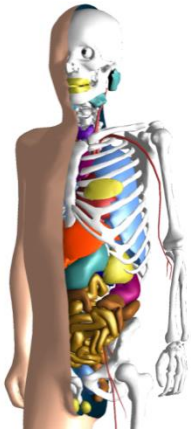
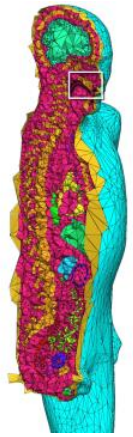
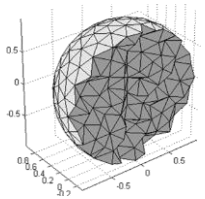
Place Pohang Accelerator Res. Bld-2-201

What Is Computational Human Phantoms?

Model of the human body for computational analysis
(e.g., for MC dose calculation or radiation imaging simulation)

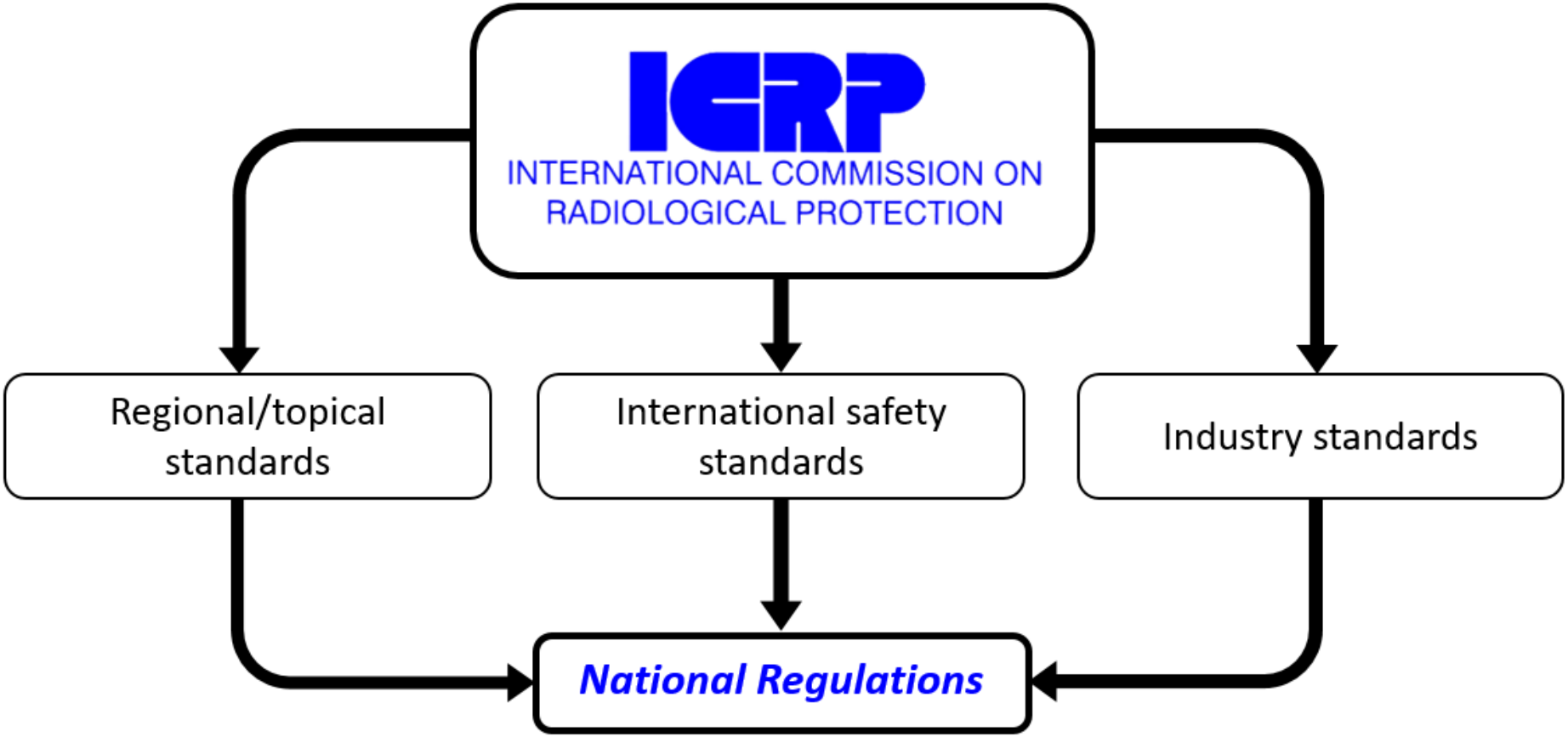


Advances in Phantom Geometry

1950s~1960s	1st Generation (1960s~)	2nd Generation (1980s~)	3rd Generation (2000s~)	4th Generation (2010s~)
Simplified	Stylized (Equations)	Voxel	NURBS/Polygon Mesh (Surface Mesh)	Tetrahedron Mesh (Volume Mesh)
				 
MIRD5, ADAM/EVA, KMIRD, …	ICRP 110, VIP-MAN, HDRKs, …	PSRKs, UF/NCI Family, RPI-AM,AF 4D XCAT, …	THRKs, MRCPs, MRKPs	
• Not realistic	• Stair-stepped surface • No thin/tiny tissue • Not deformable	• Smooth surface • Thin/tiny tissue • Deformable • Voxelization • No sub-organ density variation modeling	• Smooth surface • Thin/tiny tissue • Deformable • No voxelization • Sub-organ density variation modeling • Fast computation	

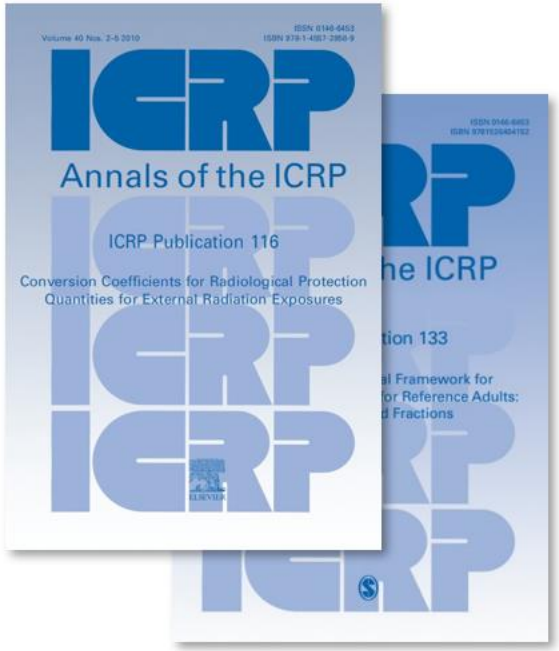
ICRP (International Commission on Radiological Protection)

- The main mission of the ICRP is to provide ***recommendations*** and ***guidance*** on radiation protection.



ICRP Reference Dose Coefficients

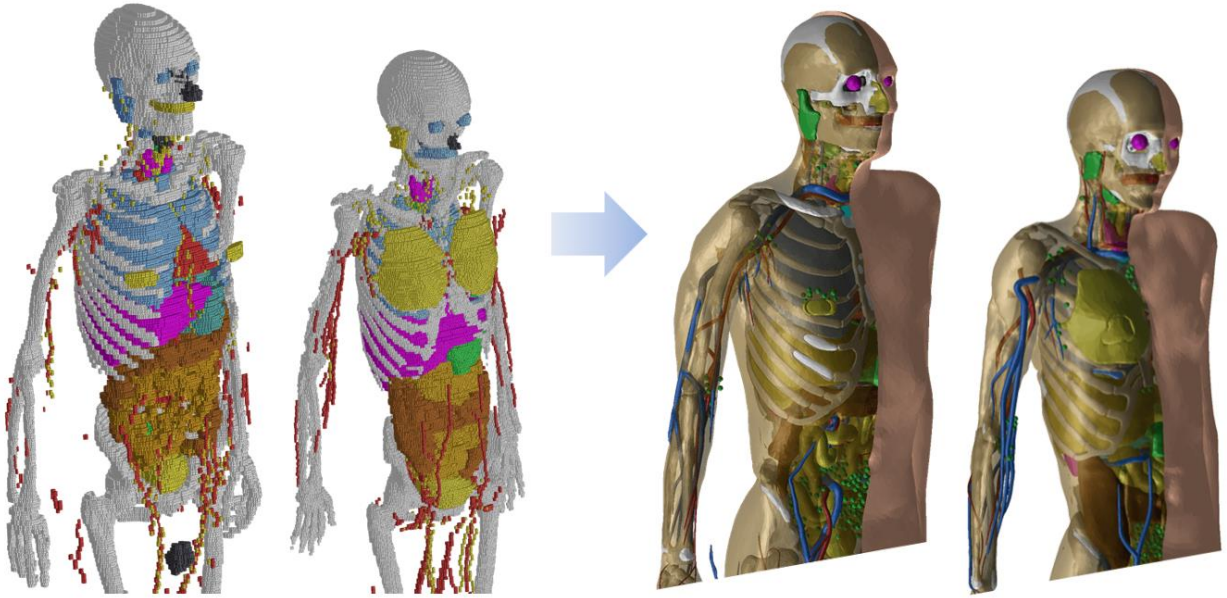
- The ICRP provides ***reference dose coefficients*** for external and internal exposures to assess the health risk from ionizing radiation by using ***computational phantoms***.



ICRP publication 116/133

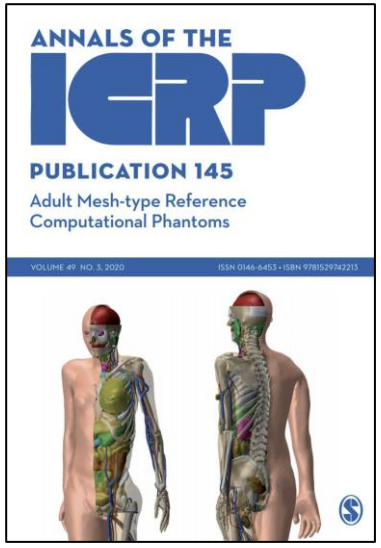
- **ICRP reference phantoms**
  - **Adult phantoms (2)**
    - Adult M/F
  - **Pediatric phantoms (10)**
    - 0, 1, 5, 10, 15 years M/F
  - **Pregnant phantoms (8)**
    - 8, 10, 15, 20, 25, 30, 35, 38 weeks

ICRP145: Adult Mesh-type Reference Computational Phantoms (MRCPs)



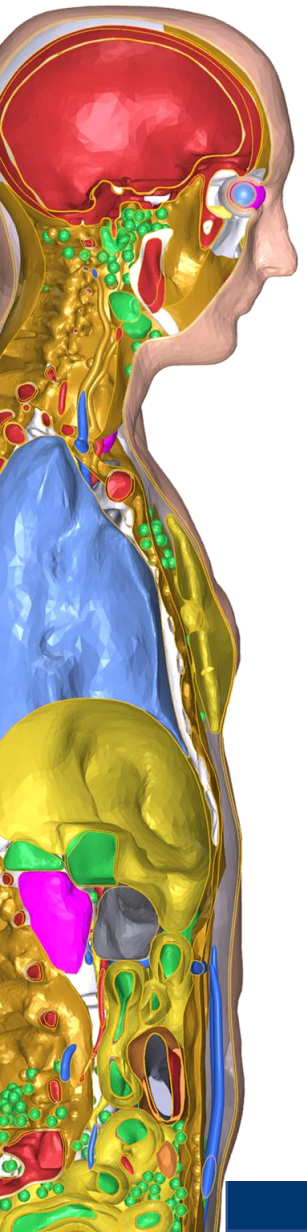
ICRP 110 Reference Phantoms

MRCPs

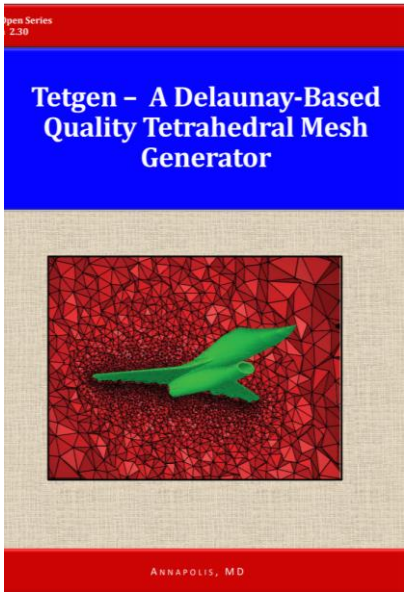


- Conversion from the ICRP110 voxel-type reference computational phantoms.
- **Elaborate modeling of microscopic radiosensitive target and source layer of organs/tissues**.
- **<u>High deformability</u>**.
- Two-format
  - Polygonal-mesh format
  - **Tetrahedral-mesh format.**

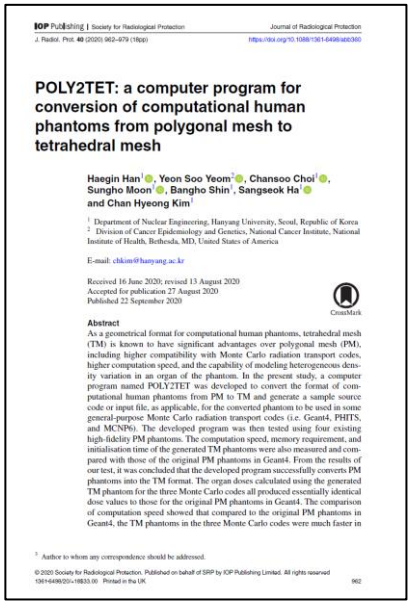
ICRP145: Adult Mesh-type Reference Computational Phantoms (MRCPs)



Tetrahedralization



TETGEN



POLY2TET: a computer program for conversion of computational human phantoms from polygonal mesh to tetrahedral mesh

Haegin Han, Yeon Soo Yeom, Chansoo Choi, Sungho Moon, Bangho Shin, Sangseok Ha and Chan Hyeong Kim

POLY2TET

Example Overview: Main

ICRP145_HumanPhantoms

Name	Contents
ICRP145phantoms.cc	source file including main() function
CMakeLists.txt	cmake file to complie and link the source files
source.mac	Source definition using *GeneralParticleSource* functions
init_vis.mac	visualization macro file
vis.mac	visualization macro file including viewer's option

include

src

build

ICRP145data

- All header and source files related to **"ICRP145standalone"** will not be covered in this lecture. While this code supports visualization, **"ICRP145phantoms"** is sufficient for running and visualization.

Example Overview: Header/Source Files

ICRP145_HumanPhantoms

include

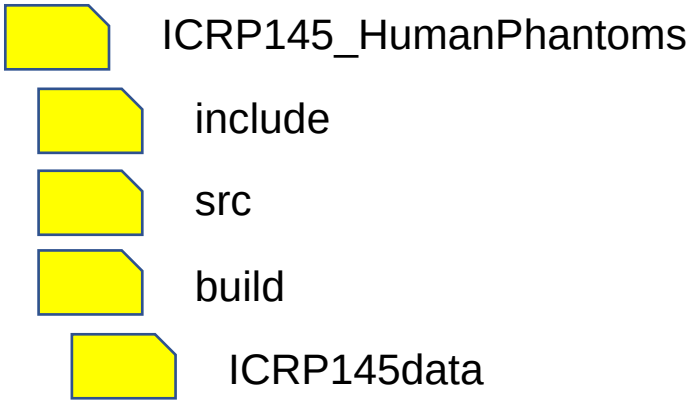
src

Name	Contents
TETModelImport.hh/cc	Read phantom data
TETDetectorConstruction.hh/cc	Install phantom
TETParameterisation.hh/cc	Install each tetrahedron
TETPSEnergyDeposit.hh/cc	Scoring energy deposition by organ ID
TETPrimaryGeneratorAction.hh/cc	Generate primary particles
TETSteppingAction.hh/cc	Stepping action to avoid particle overlapping on mesh surface
TETRun.hh/cc	Collect energy deposition data by each thread and merge them to master thread.
TETRunAction.hh/cc	Print organ dose and statistical relative err.

build

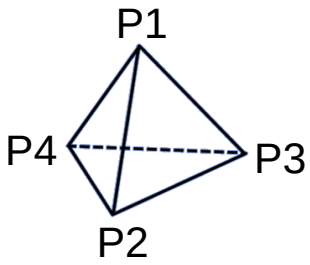
ICRP145data

Example Overview: Phantom Data



Name	Contents
colour.dat	RGBA data of the organs/tissues
MRCP_AM/AF.node	Tetrahedron vertex position data
MRCP_AM/AF.ele	Tetrahedron vertex numbering data
MRCP_AM/AF.material	Density, composition data

Phantom Data: .node and .ele files



Tetrahedron is defined by …

4 vertices	4 facets
(X_{P1}, Y_{P1}, Z_{P1}) (X_{P3}, Y_{P3}, Z_{P3})	(P1, P2, P3) (P1, P3, P4)
(X_{P2}, Y_{P2}, Z_{P2}) (X_{P4}, Y_{P4}, Z_{P4})	(P1, P4, P2) (P2, P4, P3)
MRCP_AF.node	**MRCP_AF.ele**

- MRCP_AF.node
  - First line

1279642	3	0	0
<# of vertices>	**<dimension>**	**<n/a>**	**<n/a>**

  - Remaining line list # of points

0	1.728173	0.27409899999999998	33.475464000000002
1	1.550969	0.48175099999999998	33.402968999999999
<vertex ID>	**<x>**	**<y>**	**<z>**

- MRCP_AF.ele
  - First line

8582677	4	1
<# of tet>	**<vertices per tet>**	**<# of attributes (for organ ID)>**

  - Remaining line list # of points

0	1226007	1148977	1225948	1149037	12501
1	901017	459351	901018	459350	11700
<tet ID>	**<vertex 1>**	**<vertex 2>**	**<vertex 3>**	**<vertex 4>**	**<organ ID>**

Phantom Data: .material files

ANNEX A. LIST OF ORGAN IDENTIFICATION NUMBERS, MEDIUM, DENSITY, AND MASS OF EACH ORGAN/TISSUE

Table A.1. List of organ identification (ID) number, medium, density, and mass of each organ/tissue in tetrahedral mesh (TM) phantoms.

			Density ($g\,cm^{-3}$)		Mass (g)	
Organ ID	Organ/tissue	Medium	Male	Female	Male	Female
100	Adrenal, left	1	1.036	1.035	8.683	6.817
200	Adrenal, right	1	1.036	1.035	8.683	8.649
300	ET_1, 0~8 μm	2	1.031	1.031	0.022	0.009
301	ET_1, 8~40 μm	2	1.031	1.031	0.090	0.035

ANNEX B. LIST OF MEDIA AND THEIR ELEMENTAL COMPOSITIONS

Table B.1. List of media, their elemental compositions (percentage by mass), and their densities for the adult male mesh-type reference phantom.

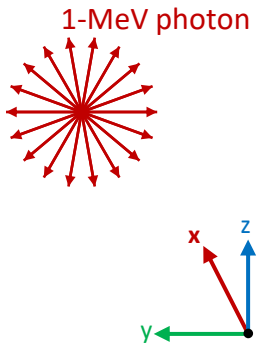
Medium no.		H	C	N	O	Na	Mg	P	S	Cl	K	Ca	Fe	I	Density ($g\,cm^{-3}$)
1	Adrenal	10.4	22.8	2.8	63.0	0.1		0.2	0.3	0.2	0.2				1.036
2	ET, trachea, BB, bb, gallbladder wall, pituitary gland, salivary glands, spinal cord, thymus, tonsils, ureter	10.5	25.1	2.7	60.7	0.1		0.2	0.3	0.2	0.2				1.031
3	Oral mucosa, tongue	10.2	14.2	3.4	71.1	0.1		0.2	0.3	0.1	0.4				1.050
4	Blood	10.2	11.0	3.3	74.5	0.1		0.1	0.2	0.3	0.2		0.1		1.060
5	Cortical bone	3.6	15.9	4.2	44.8	0.3	0.2	9.4	0.3			21.3			1.904
6	Medullary cavity	11.5	63.6	0.7	23.9	0.1			0.1	0.1					0.981
7	Humeri, upper, spongiosa	8.1	35.4	2.8	41.0	0.2	0.1	3.7	0.2	0.1	0.1	8.3			1.233

```
C Adrenal_left 1.036 g/cm3
m100     1000       -0.104
      6000       -0.228
      7000       -0.028
      8000       -0.63
      11000       -0.001
      15000       -0.002
      16000       -0.003
      17000       -0.002
      19000       -0.002
C
C Adrenal_right 1.036 g/cm3
m200     1000       -0.104
      6000       -0.228
      7000       -0.028
      8000       -0.63
      11000       -0.001
      15000       -0.002
      16000       -0.003
      17000       -0.002
      19000       -0.002
C
```

Example Exposure Scenario

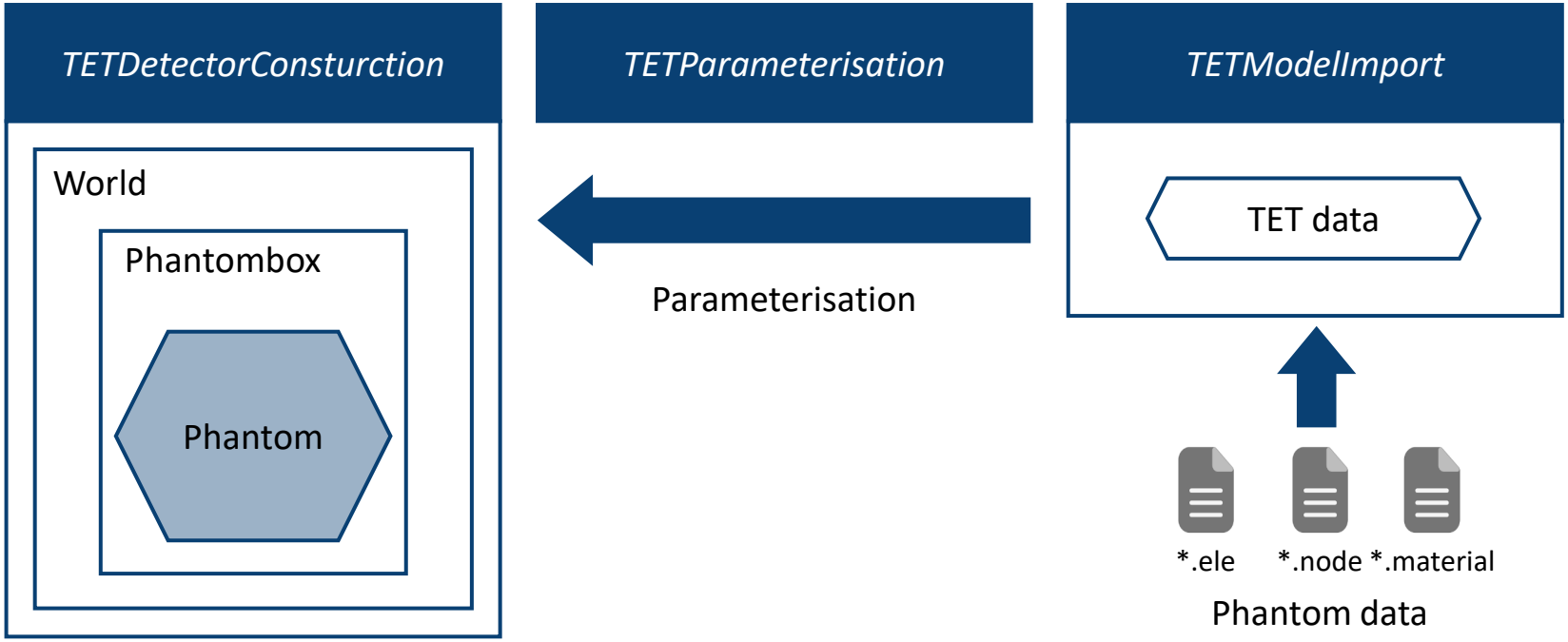
Common **external** exposure scenario

- Particle: photon
- Energy: 1 MeV
- Type and position: point source located 1 m in front of the phantom
- Direction: isotropic
- Result: average absorbed dose for each ID



Geant4 Classes: Geometries

- *TETModelImport* is to **import the phantom data** (“*.ele”, “*.node”, “*.material” files).
- *TETDetectorConstruction* is to **construct the phantom and other geometries** (phantombox, world, scorer).
- *TETParameterisation* is to **define the tetrahedral mesh phantom** by parameterisation using the phantom data which is imported in *TETModelImport*.



Geant4 Classes: Geometries (Cont'd)

- *TETModelImport* is to **import the phantom data** ("*.ele", "*.node", "*.material" files).

```
31  #include "TETModelImport.hh"
32
33  TETModelImport::TETModelImport(G4bool isAF, G4UIExecutive* ui)
34  {
35      // set path for phantom data
36      char* pPATH = getenv("PHANTOM_PATH");
37      if( pPATH == 0 ){
38          // exception for the case when PHANTOM_PATH environment variable was not set
39          G4Exception("TETModelImport::TETModelImport","",JustWarning,
40                  G4String("PHANTOM_PATH environment variable was not set.").c_str());
41          // default path for phantom data
42          phantomDataPath = "../../phantoms";
43      }
44      else {
45          // set path for phantom data as PHANTOM_PATH
46          phantomDataPath = pPATH;
47      }
48
49      // set phantom name
50      if(!isAF) phantomName = "MRCP_AM";
51      else      phantomName = "MRCP_AF";
52
53      G4cout << "===========================================================================================<<G4endl;
54      G4cout << "\t" << phantomName << " was implemented in this CODE!!   "<< G4endl;
55      G4cout << "===========================================================================================<<G4endl;
56
57      G4String eleFile      =  phantomName + ".ele";
58      G4String nodeFile     =  phantomName + ".node";
59      G4String materialFile =  phantomName + ".material";
60
61      // read phantom data files (*. ele, *.node)
62      DataRead(eleFile, nodeFile);
63      // read material file (*.material)
64      MaterialRead(materialFile);
65      // read colour data file (colour.dat) if this is interactive mode
66      if(ui) ColourRead();
67      // print the summary of phantom information
68      PrintMaterialInfomation();
69  }
```

Read ELE and NODE tetrahedral-mesh phantom files

Read Material file in MCNP format

Read organ color data for visualization

Write phantom information (e.g., phantom size and organ mass)

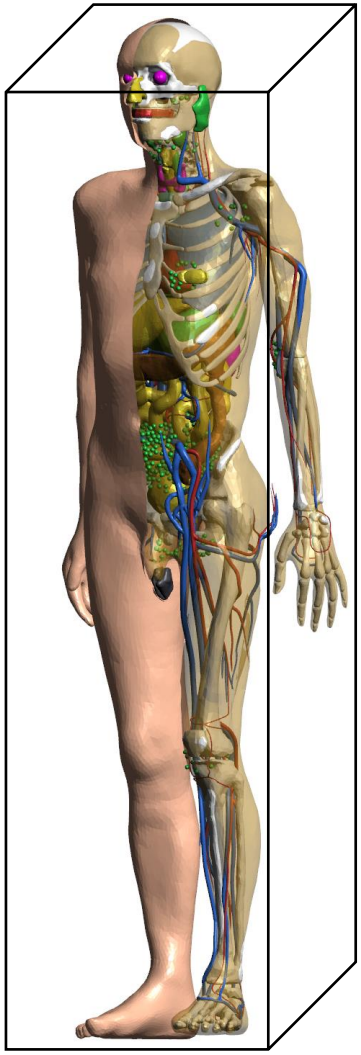
Geant4 Classes: Geometries (Cont'd)

- *TETDetectorConstruction* is to **construct the phantom and other geometries** (phantombox, world, scorer).

```
58  void TETDetectorConstruction::SetupWorldGeometry()
59  {
60      // Define the world box (size: 10*10*10 m3)
61      //
62      G4double worldXYZ = 10. * m;
63      G4Material* vacuum = G4NistManager::Instance()->FindOrBuildMaterial("G4_Galactic");
64
65      G4VSolid* worldSolid
66        = new G4Box("worldSolid", worldXYZ/2, worldXYZ/2, worldXYZ/2);
67
68      G4LogicalVolume* worldLogical
69        = new G4LogicalVolume(worldSolid,vacuum,"worldLogical");
70
71      worldPhysical
72        = new G4PVPlacement(0,G4ThreeVector(), worldLogical,"worldPhysical", 0, false,0,false);
73                                                                          Define world
74      // Define the phantom container (10-cm margins from the bounding box of phantom)
75      //
76      G4Box* containerSolid = new G4Box("phantomBox", phantomSize.x()/2 + 10.*cm,
77                                                      phantomSize.y()/2 + 10.*cm,
78                                                      phantomSize.z()/2 + 10.*cm);
79
80      container_logic = new G4LogicalVolume(containerSolid, vacuum, "phantomLogical");
81
82      new G4PVPlacement(0, G4ThreeVector(), container_logic, "PhantomPhysical",
83                        worldLogical, false, 0);
84      container_logic->SetOptimisation(TRUE);                    Define phantom box
85      container_logic->SetSmartless( 0.5 ); // for optimization (default=2)
86  }
```

Geant4 Classes: Geometries (Cont'd)

- *TETDetectorConstruction* is to **construct the phantom and other geometries** (phantombox, world, scorer).



```
// Define the phantom container (10-cm margins from the bounding box of phantom)
//
G4Box* containerSolid = new G4Box("phantomBox", phantomSize.x()/2 + 10.*cm,
                                                phantomSize.y()/2 + 10.*cm,
                                                phantomSize.z()/2 + 10.*cm);

container_logic = new G4LogicalVolume(containerSolid, vacuum, "phantomLogical");

new G4PVPlacement(0, G4ThreeVector(), container_logic, "PhantomPhysical",
                  worldLogical, false, 0);
container_logic->SetOptimisation(TRUE);
container_logic->SetSmartless( 0.5 ); // for optimization (default=2)
```

margin

smart voxel tuning

- *Phantom box functions:*
  - *make 3D-transform easy.*
  - *make particle searching process efficient.*
  - *smart voxel tuning for phantom*
- *Margin: if phantom box exactly fit with the phantom, **the particle can be stuck** at the vertex faced with the phantom box boundary.*

Geant4 Classes: Geometries (Cont'd)

- *TETDetectorConstruction* is to **construct the phantom and other geometries** (phantombox, world, scorer).
- *TETParameterisation* is to **define the tetrahedral mesh phantom** by parameterisation using the phantom data which is imported in *TETModelImport*.
- What is *parameterization*?
  - Parameterised volumes are the **repeated volumes with the multiple copies of a volume in different size, solid type, or material.**
  - The type of solid, dimensions, transformation matrix, and material can be parameterized in function of the copy number.
  - When the parameterisation is used, the Geant4 automatically computes and updates the Physical Volume at run time.

```
88  void TETDetectorConstruction::ConstructPhantom()
89  {
90      // Define the tetrahedral mesh phantom as a parameterised geometry
91      //
92      // solid and logical volume to be used for parameterised geometry
93      G4VSolid* tetraSolid = new G4Tet("TetSolid",
94                                       G4ThreeVector(),
95                                       G4ThreeVector(1.*cm,0,0),
96                                       G4ThreeVector(0,1.*cm,0),
97                                       G4ThreeVector(0,0,1.*cm));
98
99      G4Material* vacuum = G4NistManager::Instance()->FindOrBuildMaterial("G4_Galactic");
100     tetLogic = new G4LogicalVolume(tetraSolid, vacuum, "TetLogic");
101
102     // physical volume (phantom) constructed as parameterised geometry
103     new G4PVParameterised("wholePhantom",tetLogic,container_logic,
104                           kUndefined,tetData->GetNumTetrahedron(),
105                           new TETParameterisation(tetData));
106 }
```

Implementation of tetrahedral-mesh phantoms using parameterization class

(Note that G4PVParameterised has only one mother logical volume → phantom box)

Geant4 Classes: Geometries (Cont'd)

- *TETParameterisation* is to **define the tetrahedral mesh phantom** by parameterisation using the phantom data which is imported in *TETModelImport*.

```
49  TETParameterisation::~TETParameterisation()
50  {}
51
52  G4VSolid* TETParameterisation::ComputeSolid(
53                          const G4int copyNo, G4VPhysicalVolume* )
54  {
55      // return G4Tet*
56      return tetData->GetTetrahedron(copyNo);
57  }
58
59  void TETParameterisation::ComputeTransformation(
60                          const G4int,G4VPhysicalVolume*) const
61  {}
62
63  G4Material* TETParameterisation::ComputeMaterial(const G4int copyNo,
64                                                   G4VPhysicalVolume* phy,
65                                                   const G4VTouchable* )
66  {
67     // set the colour for each organ if visualization is required
68      if(isforVis){
69          G4int idx = tetData->GetMaterialIndex(copyNo);
70          phy->GetLogicalVolume()->SetVisAttributes(visAttMap[idx]);
71      }
72
73      // return the material data for each material index
74      return tetData->GetMaterial(tetData->GetMaterialIndex(copyNo));
75  }
```

Install tetrahedrons in function of the copy number

→ In this example, the tetrahedrons are saved in *TETModelImport*

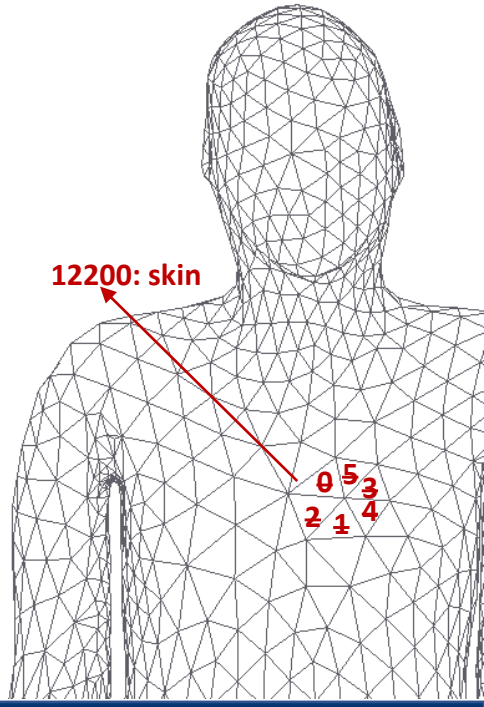
Change the material of the volume in function of the copy number

→ In this example, the material information are saved in *TETModelImport*

Geant4 Classes: Geometries (Cont'd)

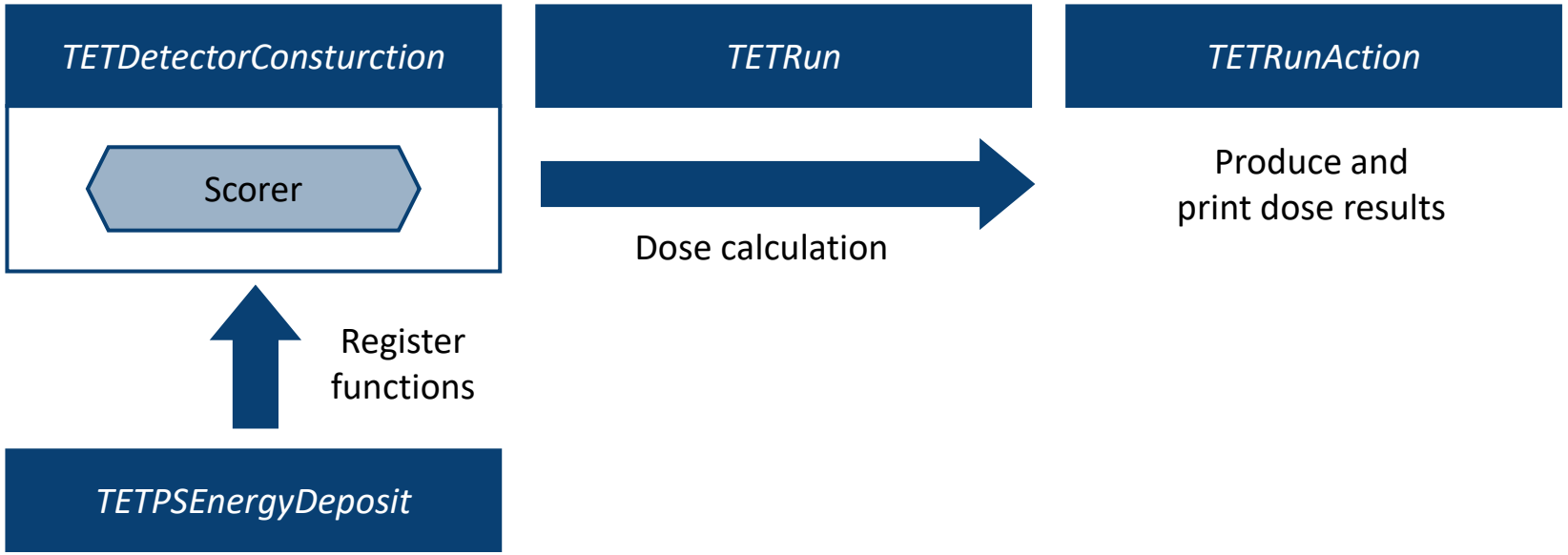
- *TETParameterisation* is to **define the tetrahedral mesh phantom** by parameterisation using the phantom data which is imported in *TETModelImport*.

```
49  TETParameterisation::~TETParameterisation()
50  {}
51
52  G4VSolid* TETParameterisation::ComputeSolid(
53                          const G4int copyNo, G4VPhysicalVolume* )
54  {
55      // return G4Tet*
56      return tetData->GetTetrahedron(copyNo);
57  }
58
59  void TETParameterisation::ComputeTransformation(
60                          const G4int,G4VPhysicalVolume*) const
61  {}
62
63  G4Material* TETParameterisation::ComputeMaterial(const G4int copyNo,
64                                                   G4VPhysicalVolume* phy,
65                                                   const G4VTouchable* )
66  {
67     // set the colour for each organ if visualization is required
68      if(isforVis){
69          G4int idx = tetData->GetMaterialIndex(copyNo);
70          phy->GetLogicalVolume()->SetVisAttributes(visAttMap[idx]);
71      }
72
73      // return the material data for each material index
74      return tetData->GetMaterial(tetData->GetMaterialIndex(copyNo));
75  }
```



Geant4 Classes: Organ Dose Calculation

- *TETDetectorConstruction* is to **construct scorer** in the phantom to get doses in organs.
- *TETPSEnergyDeposit* is to **specify scorer functions** for energy deposition.
- *TETRun* is to **calculate organ doses**.
- *TETRunAction* is to **produce and print dose results**.



Geant4 Classes: Organ Dose Calculation (Cont'd)

- *TETDetectorConstruction* is to **construct scorer** in the phantom to get doses in organs.

```
108   void TETDetectorConstruction::ConstructSDandField()
109   {
110       // Define detector (Phantom SD) and scorer (eDep)
111       //
112       G4SDManager* pSDman = G4SDManager::GetSDMpointer();
113       G4String phantomSDname = "PhantomSD";
114
115       // MultiFunctional detector
116       G4MultiFunctionalDetector* MFDet = new G4MultiFunctionalDetector(phantomSDname);
117       pSDman->AddNewDetector( MFDet );
118
119       // scorer for energy depositon in each organ
120       MFDet->RegisterPrimitive(new TETPSEnergyDeposit("eDep", tetData));
121
122       // attach the detector to logical volume for parameterised geometry (phantom geometry)
123       SetSensitiveDetector(tetLogic, MFDet);
124   }
```

Compute energy deposition in each organ using G4*PSEnergyDeposit* class

Geant4 Classes: Organ Dose Calculation (Cont'd)

- *TETPSEnergyDeposit* is to **specify scorer functions** for energy deposition.

```
40  G4int TETPSEnergyDeposit::GetIndex(G4Step* aStep)
41  {
42      // return the organ ID (= material index)
43      G4int copyNo = aStep->GetPreStepPoint()->GetTouchable()->GetCopyNumber();
44      return tetData->GetMaterialIndex(copyNo);
45  }
46
```

Energy deposition is stored in the organs where the *step* interacted

- *TETRun* is to **calculate organ doses**.

```
42  void TETRun::RecordEvent(const G4Event* event)
43  {
44      auto  fCollID
45      = G4SDManager::GetSDMpointer()->GetCollectionID("PhantomSD/eDep");
46
47      // Hits collections
48      //
49      G4HCofThisEvent* HCE = event->GetHCofThisEvent();
50      if(!HCE) return;
51
52      G4THitsMap<G4double>* evtMap =
53              static_cast<G4THitsMap<G4double>*>(HCE->GetHC(fCollID));
54
55      // sum up the energy deposition and the square of it
56      for (auto itr : *evtMap->GetMap()) {
57          edepMap[itr.first].first  += *itr.second;                    //sum
58          edepMap[itr.first].second += (*itr.second) * (*itr.second); //sum square
59      }
60  }
```

Compute organ doses from the stored data in *TETPSEnergyDeposit* class

Geant4 Classes: Organ Dose Calculation (Cont'd)

- *TETRun* is to **calculate organ doses**.

```
42  void TETRun::RecordEvent(const G4Event* event)
43  {
44      auto  fCollID
45      = G4SDManager::GetSDMpointer()->GetCollectionID("PhantomSD/eDep");
46
47      // Hits collections
48      //
49      G4HCofThisEvent* HCE = event->GetHCofThisEvent();
50      if(!HCE) return;
51
52      G4THitsMap<G4double>* evtMap =
53              static_cast<G4THitsMap<G4double>*>(HCE->GetHC(fCollID));
54
55      // sum up the energy deposition and the square of it
56      for (auto itr : *evtMap->GetMap()) {
57          edepMap[itr.first].first  += *itr.second;                    //sum
58          edepMap[itr.first].second += (*itr.second) * (*itr.second); //sum square
59      }
60  }
```

Compute organ doses from the stored data in *TETPSEnergyDeposit* class

```
62  void TETRun::Merge(const G4Run* run)
63  {
64   // merge the data from each thread
65   EDEPMAP localMap = static_cast<const TETRun*>(run)->fEdepMap;
66
67   for(auto itr : localMap){
68       fEdepMap[itr.first].first  += itr.second.first;
69       fEdepMap[itr.first].second += itr.second.second;
70   }
71
72   G4Run::Merge(run);
73  }
```

- Each worker thread execute *RecordEvent()* and store energy deposition data in variable '*edepMap*'.
- For merging those '*edepMap*' by each worker thread, *Merge()* helps to accumulation for master thread.

Geant4 Classes: Organ Dose Calculation (Cont'd)

- *TETRunAction* is to **produce and print dose results**.

Write the result file at the *EndOfRunAction!*

```
74  void TETRunAction::PrintResult(std::ostream &out)
75  {
76      // Print run result
77      //
78      using namespace std;
79      EDEPMAP edepMap = *fRun->GetEdepMap();
80
81      out << G4endl
82          << "======================================================================" << G4endl
83          << " Run #" << runID << " / Number of event processed : "<< numOfEvent     << G4endl
84          << "======================================================================" << G4endl
85          << "organ ID| "
86          << setw(19) << "Organ Mass (g)"
87          << setw(19) << "Dose (Gy/source)"
88          << setw(19) << "Relative Error" << G4endl;
89
90      out.precision(3);
91      auto massMap = tetData->GetMassMap();
92      for(auto itr : massMap){
93          G4double meanDose    = edepMap[itr.first].first  / itr.second / numOfEvent;
94          G4double squareDoese = edepMap[itr.first].second / (itr.second*itr.second);
95          G4double variance    = ((squareDoese/numOfEvent) - (meanDose*meanDose))/numOfEvent;
96          G4double relativeE   = sqrt(variance)/meanDose;
97
98          out << setw(8)  << itr.first << "| "
99              << setw(19) << fixed      << itr.second/g;
100         out << setw(19) << scientific << meanDose/(joule/kg);
101         out << setw(19) << fixed      << relativeE << G4endl;
102     }
103     out << "======================================================================" << G4endl << G4endl;
104 }
```

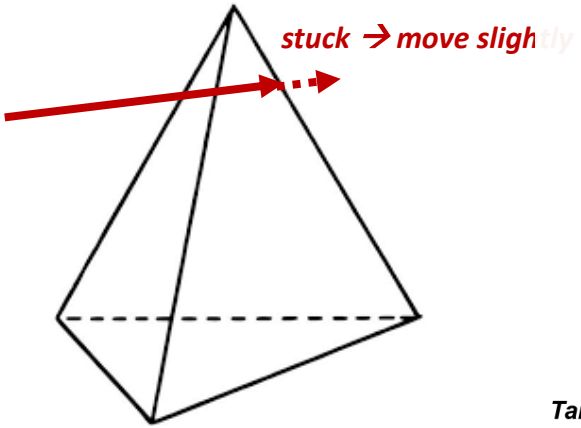
Geant4 Classes: Organ Dose Calculation (Cont'd)

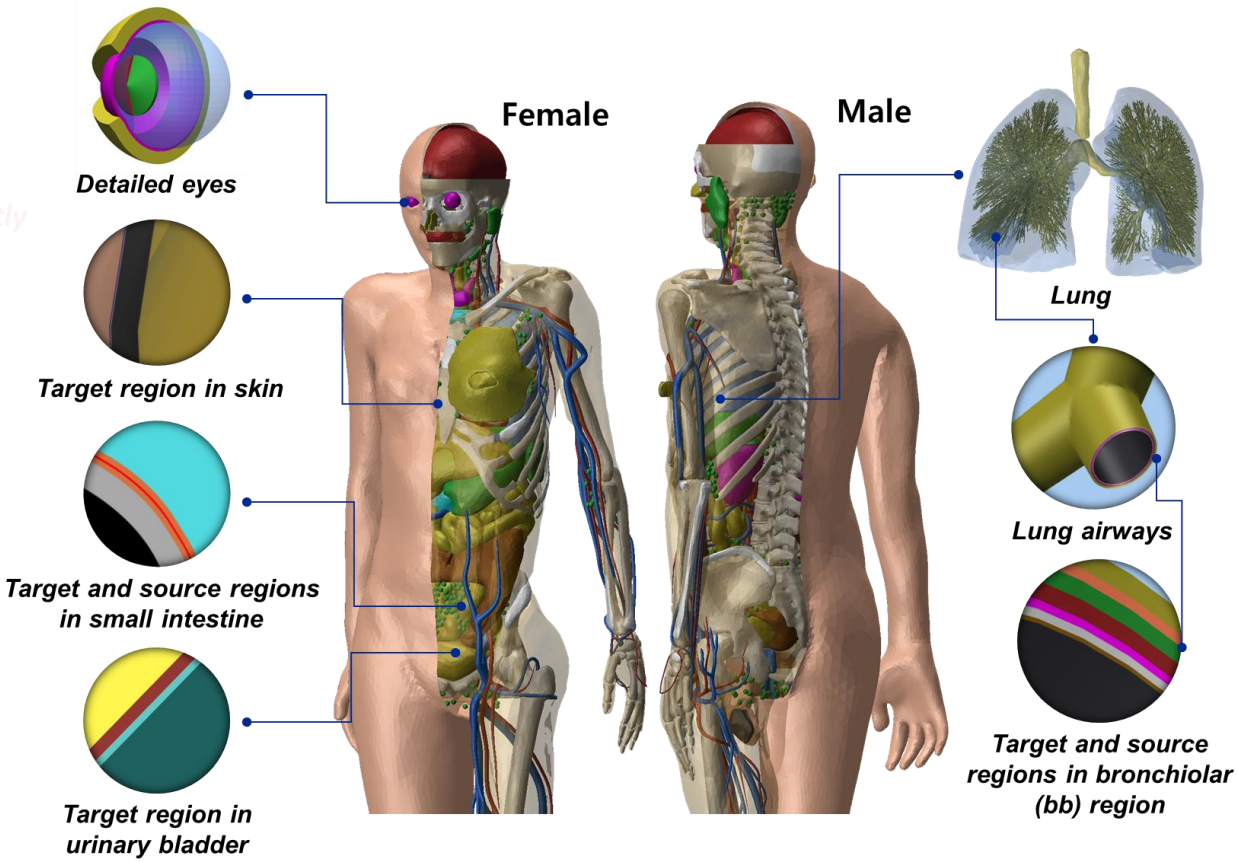
- *TETRunAction* is to **produce and print dose results**.

```
 1
 2  ============================================================================
 3   Run #0 / Number of event processed : 10000000
 4  ============================================================================
 5  organ ID|      Organ Mass (g)    Dose (Gy/source)      Relative Error
 6        100|              8.683           1.823e-17               0.214
 7        200|              8.683           2.226e-17               0.184
 8        300|              0.022           1.212e-17               0.623
 9        301|              0.090           1.298e-17               0.594
10        302|              0.028           1.613e-17               0.553
11        303|             11.291           2.867e-17               0.163
12        400|              0.141           3.189e-17               0.582
13        401|              0.390           1.916e-17               0.400
14        402|              0.098           1.795e-17               0.367
15        403|              0.049           2.251e-17               0.450
16        404|              0.098           2.337e-17               0.340
17        405|             28.808           2.232e-17               0.105
18        500|              0.086           9.438e-18               0.511
19        501|              0.024           4.264e-17               0.521
20        600|              0.023           3.064e-17               0.470
21        700|             10.364           2.425e-17               0.165
22        800|              0.025           7.156e-18               0.530
23        801|              0.031           7.956e-18               0.529
24        802|              0.052           1.636e-17               0.735
25        803|              0.130           1.912e-17               0.470
26        804|              0.026           4.134e-17               0.673
27        805|              0.052           4.320e-17               0.589
28        806|              0.052           3.789e-17               0.545
```

Geant4 Classes: etc.

- *TETActionInitialization* is to **initialize action classes** (e.g., *TETRunAction*).
- *TETSteppingAction* is to **resolve stuck particles** in each step.





microscopic scale of radiosensitive layers

Geant4 Classes: Set Source

- *TETPrimaryGeneratorAction* is to **generate sources**.
  - External exposure
    - ✓ Sources are generated through *G4GeneralParticleSource.*
- **Source information** (e.g., particle type, energy, ...) is specified in **source definition file** ("*.mac").

source.mac

Source definition file

TETPrimaryGeneratorAction

Generate sources using *G4GeneralParticleSource* or *G4ParticleGun*

- Source definition file ("Source.mac") for **external exposure**

```
# GeneralParticleSource :
# isotropic 1 MeV-gamma point source at (0, -1 m, 0)
/gps/ang/type iso
/gps/energy 1 MeV
/gps/particle gamma
/gps/pos/type Point
/gps/pos/centre 0 -1. 0 m
```

- Source can be specified by using macro commands for *G4GeneralParticleSource*.
- **Particle direction, energy, type, geometry, position** is set in sequence in the "source.mac" file.

**** For more details about G4GeneralParicleSource ("/gps/"), see Geant4 GPS manual.***

Input Macro File Example

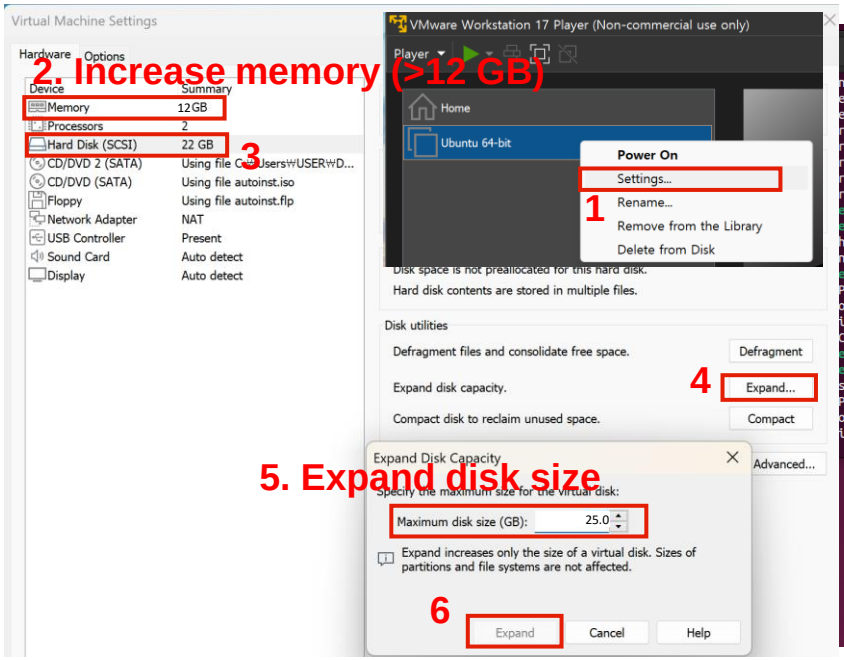
- Input macro file (“example.in”)

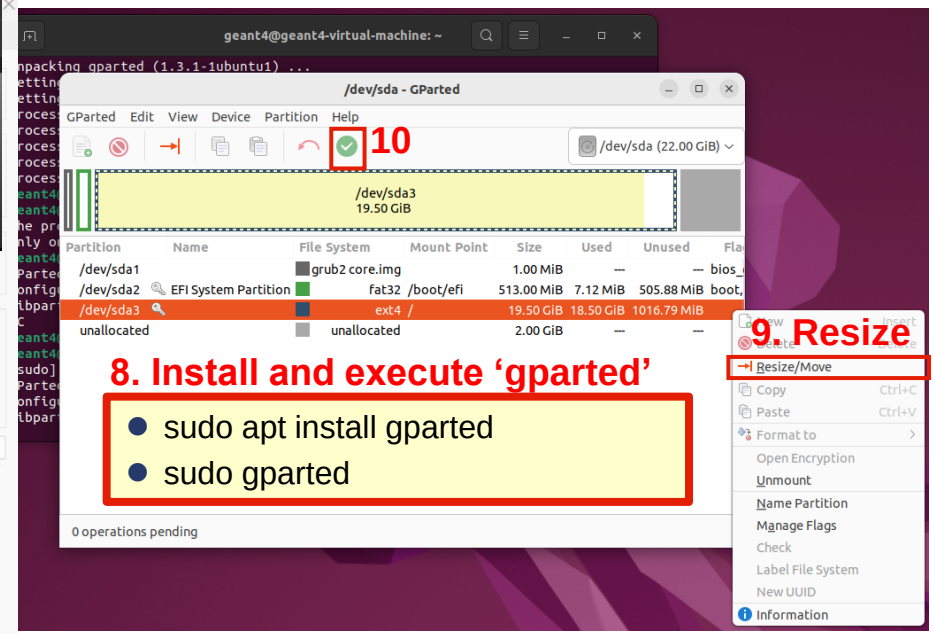
```
# Example macro file to run in the batch mode

# Set the verbose
/run/verbose 2

# Set the number of threads for multi-threading mode
#/run/numberOfThreads 1

# Initialize
/run/initialize

# source setting
/control/execute source.mac

# Set the nps
/run/beamOn 10000000
```

- Set the **number of threads** for Geant4 simulation using “/run/numberOfThreads” (Geant4 should be compiled in multi-threaded mode).
- Set the **source definition file path** using “/control/execute”.
- Set the **number of particles (NPS)** for each run using “/run/beamOn”

VM Setup and Example Build

- **If your VM disk size and memory are not enough, it should be expanded by following procedures**
  - **Phantom data: ~1.2 GB**
  - **Memory usage: ~10 GB**





- cp –r (INSTALL_PATH)/geant4-v11.3.0/examples/advanced/ICRP145_HumanPhantoms (PATH)
- cd (PATH)/ICRP145_HumanPhantoms
- mkdir build
- cd build
- cmake ..
- make

.. phantom data will be automatically downloaded.

Example Command: Batch Mode

- ./ICRP145phantoms –m example.in –o MRCP_AM.out
- ./ICRP145phantoms –m example.in –o MRCP_AF.out -f

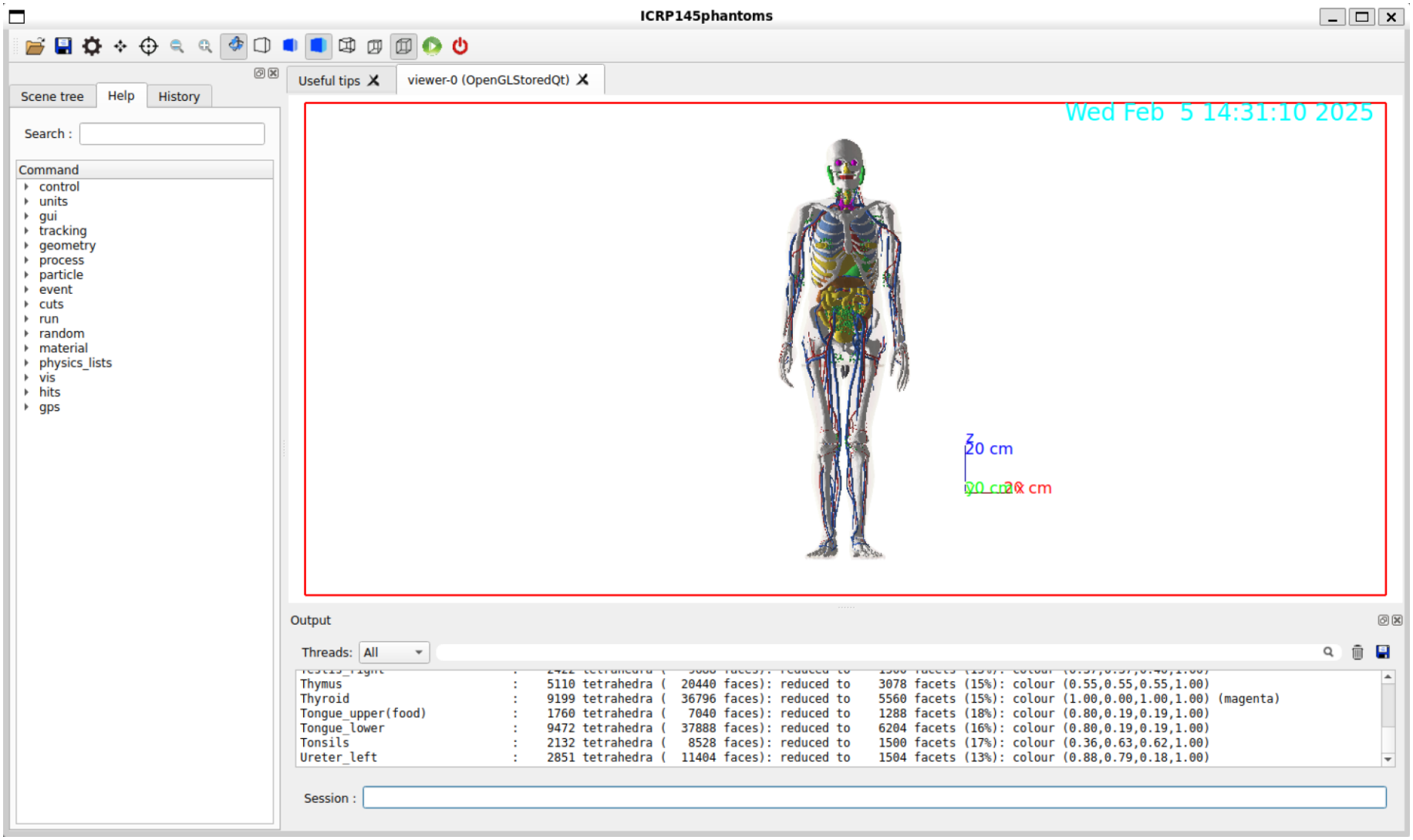
```
 1
 2  ================================================================================
 3   Run #0 / Number of event processed : 10000000
 4  ================================================================================
 5  organ ID|      Organ Mass (g)    Dose (Gy/source)      Relative Error
 6       100|                 8.683          1.823e-17                 0.214
 7       200|                 8.683          2.226e-17                 0.184
 8       300|                 0.022          1.212e-17                 0.623
 9       301|                 0.090          1.298e-17                 0.594
10       302|                 0.028          1.613e-17                 0.553
11       303|                11.291          2.867e-17                 0.163
12       400|                 0.141          3.189e-17                 0.582
13       401|                 0.390          1.916e-17                 0.400
14       402|                 0.098          1.795e-17                 0.367
15       403|                 0.049          2.251e-17                 0.450
16       404|                 0.098          2.337e-17                 0.340
17       405|                28.808          2.232e-17                 0.105
18       500|                 0.086          9.438e-18                 0.511
19       501|                 0.024          4.264e-17                 0.521
20       600|                 0.023          3.064e-17                 0.470
21       700|                10.364          2.425e-17                 0.165
22       800|                 0.025          7.156e-18                 0.530
23       801|                 0.031          7.956e-18                 0.529
24       802|                 0.052          1.636e-17                 0.735
25       803|                 0.130          1.912e-17                 0.470
26       804|                 0.026          4.134e-17                 0.673
27       805|                 0.052          4.320e-17                 0.589
28       806|                 0.052          3.789e-17                 0.545
```

Example Command: Interactive Mode

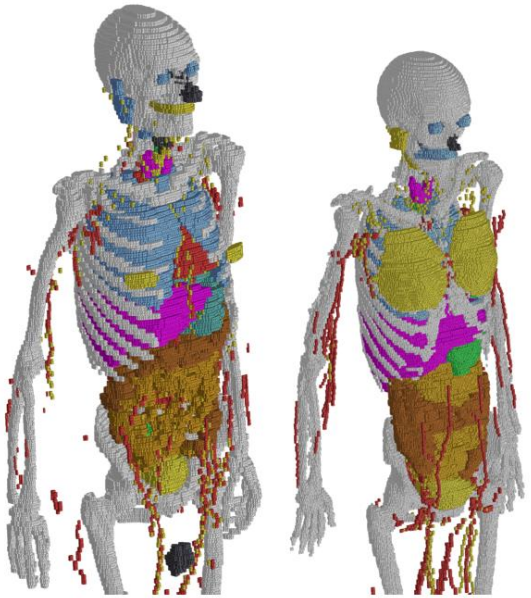
- ./ICRP145phantoms
- ./ICRP145standalone

vis.mac

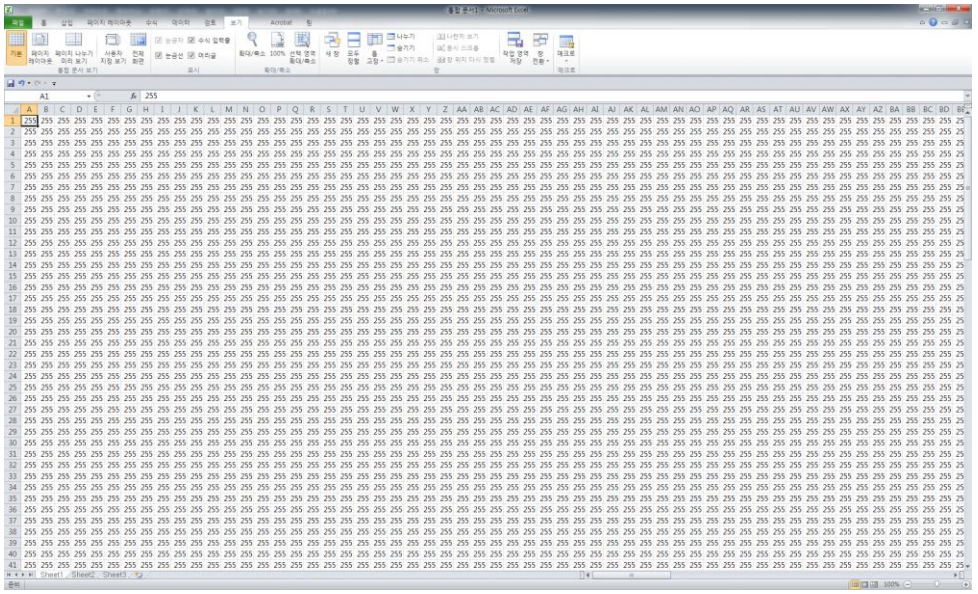
```
# Draw geometry:
/vis/viewer/set/specialMeshRendering
/vis/viewer/set/specialMeshRenderingOption surfaces
/vis/drawVolume
```

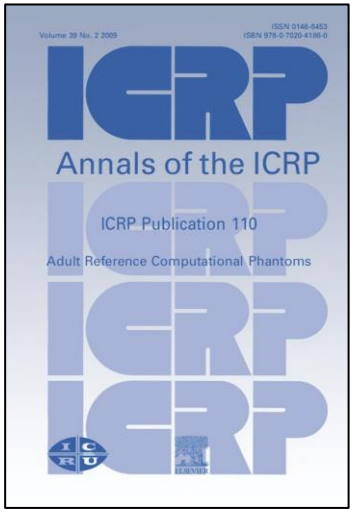


ICRP110: Adult Voxel-type Reference Computational Phantoms (VRCPs)



ICRP 110 Reference Phantoms

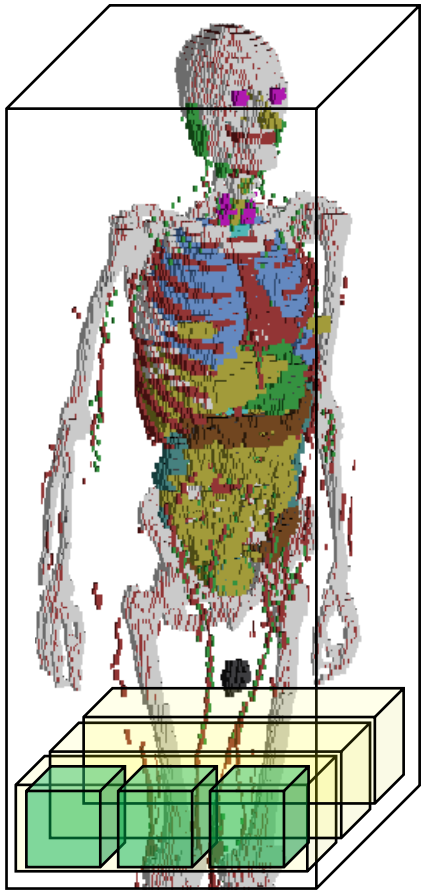




- **Highly realistic anatomical representation** based on actual patient CT images.
- **Fast computation time**.
- The masses and density of organs/tissues are consistent with the ICRP-89 reference values.

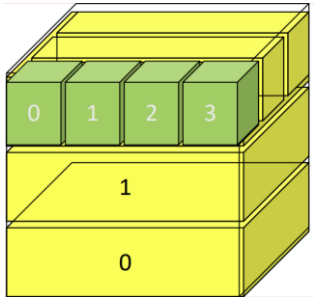
ICRP110 Example: Voxel Geometry

```
270 // Define the voxelised phantom here
271 // Replication of air Phantom Volume.
272
273 //--- Slice the phantom along Y axis
274     G4String yRepName("RepY");
275     G4VSolid* solYRep = new G4Box(yRepName,fNVoxelX*fVoxelHalfDimX,
276                                   fVoxelHalfDimY, fNVoxelZ*fVoxelHalfDimZ);
277     auto logYRep = new G4LogicalVolume(solYRep,matAir,yRepName);
278     new G4PVReplica(yRepName,logYRep,fContainer_logic,kYAxis, fNVoxelY,fVoxelHalfDimY*2.);
279
280     logYRep -> SetVisAttributes(new G4VisAttributes(G4VisAttributes::GetInvisible()));
281
282 //--- Slice the phantom along X axis
283     G4String xRepName("RepX");
284     G4VSolid* solXRep = new G4Box(xRepName,fVoxelHalfDimX,fVoxelHalfDimY,
285                                   fNVoxelZ*fVoxelHalfDimZ);
286     auto logXRep = new G4LogicalVolume(solXRep,matAir,xRepName);
287     new G4PVReplica(xRepName,logXRep,logYRep,kXAxis,fNVoxelX,fVoxelHalfDimX*2.);
288
289     logXRep -> SetVisAttributes(new G4VisAttributes(G4VisAttributes::GetInvisible()));
290
291     //----- Voxel solid and logical volumes
292     //--- Slice along Z axis
293     G4VSolid* solidVoxel = new G4Box("phantom",fVoxelHalfDimX, fVoxelHalfDimY,fVoxelHalfDimZ);
294     auto logicVoxel = new G4LogicalVolume(solidVoxel,matAir,"phantom");
295
296     logicVoxel -> SetVisAttributes(new G4VisAttributes(G4VisAttributes::GetInvisible()));
297
298     // Parameterisation to define the material of each voxel
299     G4ThreeVector halfVoxelSize(fVoxelHalfDimX,fVoxelHalfDimY,fVoxelHalfDimZ);
300
301     auto param =  new ICRP110PhantomNestedParameterisation(halfVoxelSize, pMaterials);
302
303     new G4PVParameterised("phantom",    // their name
304                           logicVoxel, // their logical volume
305                           logXRep,      // Mother logical volume
306                           kZAxis,       // Are placed along this axis
307                           fNVoxelZ,      // Number of cells
308                           param);       // Parameterisation
309
310      param -> SetMaterialIndices(fMateIDs); // fMateIDs is  the vector with Material ID associ
311      param -> SetNoVoxel(fNVoxelX,fNVoxelY,fNVoxelZ);
```



Install voxels using NestedParametersiation

→ ***repeated voxels filled with different materials***



ICRP110 Example: Run

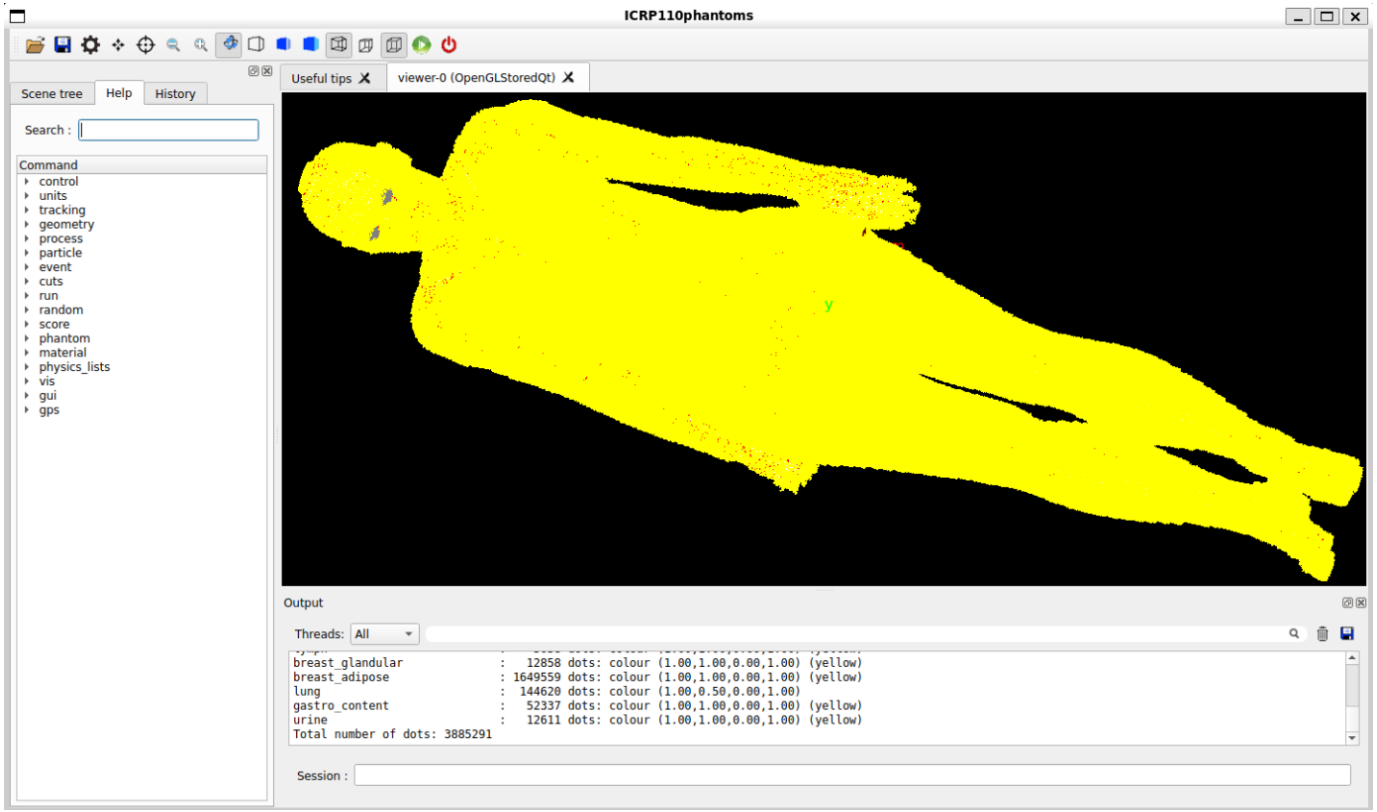
```
-------------------------------
OrganID Edep (J)        Dose (Gy)
-------------------------------
 10 |           6.09894e-12      2.24284e-11
 12 |           1.10592e-14      1.33179e-13
 20 |           3.47648e-14      1.9111e-13
 22 |           1.2511e-13       6.96061e-13
 23 |           9.23188e-15      6.61357e-14
 28 |           2.24957e-10      8.59665e-10
 29 |           4.46955e-10      9.46839e-10
 35 |           1.12742e-14      1.54573e-14
 41 |           1.14555e-09      2.87379e-09
 42 |           1.87314e-09      2.74985e-09
 44 |           2.34517e-13      4.50942e-13
 49 |           5.82756e-13      2.03348e-12
 51 |           1.45778e-12      7.82951e-12
 52 |           1.30521e-11      4.32089e-11
 53 |           5.00951e-12      4.5862e-11
 54 |           2.57771e-12      1.48563e-11
 58 |           3.94446e-12      4.44847e-11
 59 |           2.90933e-14      4.92272e-12
 74 |           7.12295e-12      1.09584e-11
 75 |           7.9356e-13       2.26719e-12
 76 |           4.66846e-14      5.18775e-13
 77 |           3.51751e-15      6.39315e-14
 82 |           7.84019e-14      8.71229e-13
 83 |           7.97285e-15      2.27796e-13
 84 |           5.39287e-12      1.34788e-10
 85 |           6.26922e-13      8.36231e-12
 86 |           7.33958e-12      2.44816e-10
 87 |           2.85918e-12      8.66472e-12
 88 |           2.29984e-12      4.5094e-12
 89 |           8.38612e-14      7.82872e-13
 95 |           1.79372e-13      9.96506e-14
103 |           2.62091e-12      2.51045e-11
105 |           3.45459e-18      3.11225e-16
107 |           2.72388e-08      1.8151e-09
108 |           3.95597e-12      1.43826e-12
109 |           1.01038e-10      1.00786e-11
115 |           3.60116e-11      2.11709e-09
117 |           5.07757e-09      4.10473e-10
118 |           3.99844e-12      2.43292e-12
119 |           5.95121e-11      1.1028e-11
123 |           3.00059e-10      2.04789e-10
124 |           4.20347e-12      6.5779e-12
125 |           4.45666e-12      3.34476e-12
129 |           2.22872e-14      1.27501e-12
130 |           5.69775e-12      3.25214e-10
135 |           8.24081e-15      9.68368e-13
```

Batch mode

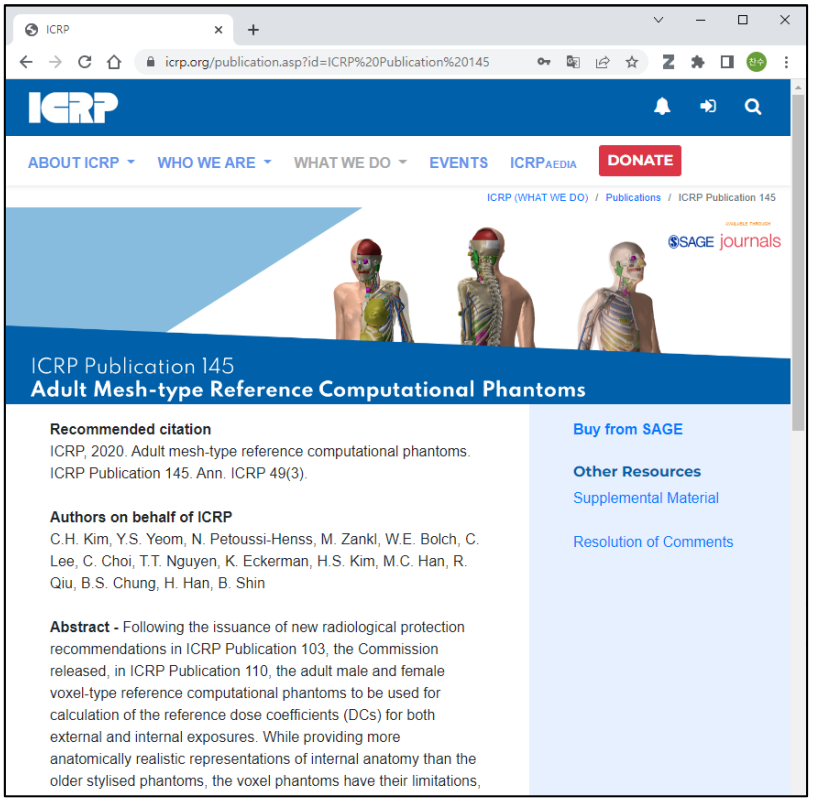
- ./ICRP110phantoms male.in
- ./ICRP110phantoms female_in

Interactive mode

- ./ICRP110phantoms
- ./ICRP110standalone



References



WHAT WE DO → PUBLICATIONS → ICRP Publication 145 or 110

https://mesh-phantom.com/