Voxelised geometries intersected with a volume

Pedro Arce

CIEMAT, Madrid

2010 Geant4 Workshop
ESA, 4-8 October 2010

Use case

- A DICOM voxelised geometry of a human or animal taken by a CT or MRI is used in a PET/SPECT or radiotherapy treatment

- DICOM geometry is a box, that encompass the human/animal plus the surrounding air
  - ☹ This box does not fit into the accelerator head or detector bore of the detector: **the air overlaps with it**

Regular navigation today

- G4RegularNavigation/G4PhantomParameterisation with skipping voxel boundaries is the fastest navigation option
  - 4/6 times faster that any other for a realistic phantom (see poster at NSS/MIC 2008)
  - ☹ But it can only be used for box geometries: voxels are inside a container volume filing it completely
    - When a track enters the container it always enters one voxel, which can be easily identified, as the geometry is very regular
    - If container would not be filled, when a track is in the space between the container walls and the voxels G4Navigator::LevelLocate has to calculate the positions of many voxels to know in which one the track is going to enter (usually there are millions of voxels)
      - ✓ No need to look into them all if 3D optimisation is used

Solution

1. Extend G4PhantomParemeterisation for cases where voxels do not form a box and do not fill completely the mother volume
   - Option 1: do not allow discontinuous geometries (no empty voxel between two filled voxels)
   - Option 2: allow discontinuous geometries

   Option 1 faster and less memory than option 2

2. Implement a tool that calculates automatically the intersection between a DICOM geometry and any volume (existing in the Geant4 geometry or created ´ad hoc´ by the user)

Old DICOM format

- **Old DICOM Geant4 text file format** (binary file format is also supported)
  - See examples/extended/medical/DICOM

```
2                    // number of materials
0 G4_AIR       // material list
1 G4_WATER
128 128 10     // number of voxels in X/Y/Z
-196 196        // Extension in X , Y , Z
-196 196
0 90
// index of material for each voxel
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 …
// densities for each voxel
0 0.0110367 0.0130433 0.01204 0.01204 0.0100333 0.0110367 0.0100333
0.00903 0.01204 0.0110367 0.00903 0.00802667 0.00903 0.0100333
…
```

New DICOM format

Same format but add list of which voxels are filled:

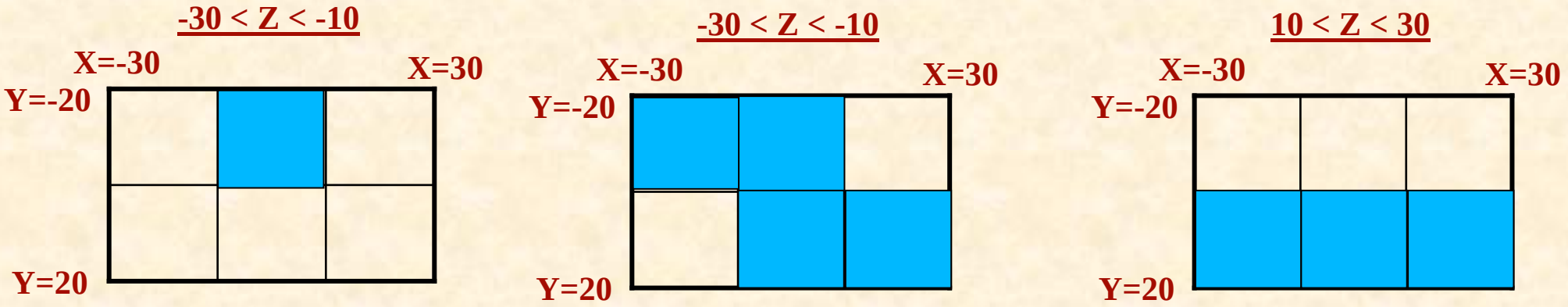
(Standard practice is to define a phantom in Z-slices)

- Loop to Z slices,
  - Loop to Y rows
    - Option 1:
      - Minimum and maximum X of the filled voxels
    - Option 2:
      - Number of discontinuous X voxel sets
      - Minimum and maximum X of each voxel set

List of material indices for each voxel

List of densities for each voxel

New DICOM format




```
2                   // number of materials
0 G4_AIR        // material list
1 G4_WATER
3 2 3     // number of voxels in X/Y/Z
-30 30       // Extension in X , Y , Z
-20 20
-30 30
// list of which voxels are filled
1 1
-1 -1
0 1
1 2
-1 -1
0 2
0 0 0 0 0 1 1 1 // material indices
0.001 0.001 0.001 0.001 0.001 1. 1. 1. // densities
```

New DICOM reader

- Create a new DICOM reader
  - Instantiates a **G4PartialPhantomParameterisation** with **fRegularStructureID = 2** (G4PhantomParameterisation has fRegularStructureID = 1)

☺ No change in G4Navigator or G4RegularNavigation needed

- Only

if(getRegularStructureID() != 1) ⇨ if(getRegularStructureID() == 0)

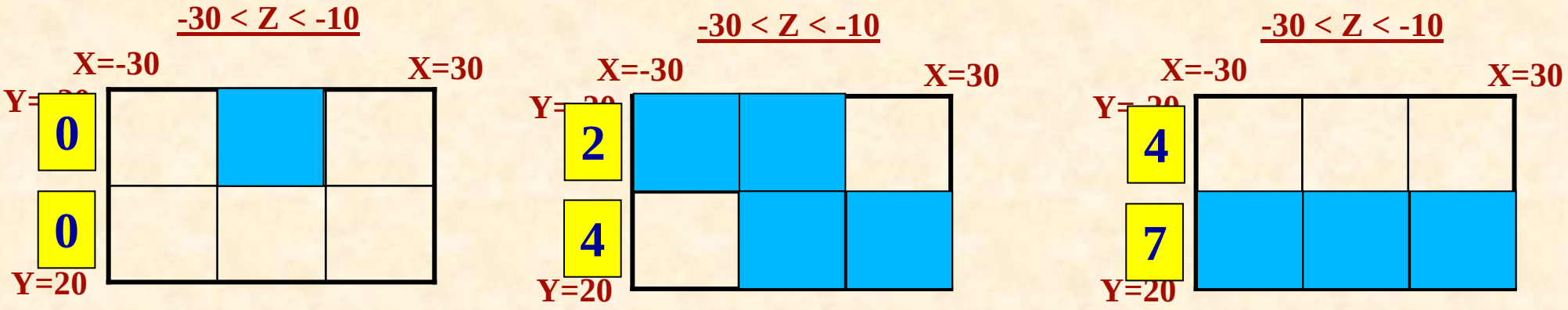
G4PartialPhantomParameterisation

Two problems to solve

1. Given a copyNo, which is the corresponding voxel position
2. Given a position in phantom local coordinates, in which voxel it is in

1. Voxel position from copyNo

- Build a std::multiset<G4int> with an entry per each ZY row (i.e. looping in Z and then in Y)
  - Store accumulated number of voxels filled -1



G4PartialPhantomParameterisation

Option 1:

- Build a G4int* minXVoxels = new G4int[totalNcopies];

Minimum ID of X voxels for each ZY row

Option 2:

- Build a std::multiset<G4int>* minXVoxels = new std::multiset<G4int>

Set of minimum IDs of X voxels in each discontinuous set for each ZY row

void G4PartialPhantomParameterisation::ComputeVoxelIndices(

```
 std::multimap<G4int,G4int>::const_iterator ite =
fFilledIDs.lower_bound(size_t(copyNo));
 G4int dist = std::distance( fFilledIDs.begin(), ite
);
 nz = size_t(dist/fNoVoxelY);
 ny = size_t( dist%fNoVoxelY );
 G4int ifmin = (*ite).second;
```

```
G4int nvoxXprev;
if( dist != 0 ) {
   ite--;
   nvoxXprev = (*ite).first;
  } else {
   nvoxXprev = -1; }
 nx = ifmin+copyNo-nvoxXprev-1;
```

G4PartialPhantomParameterisation

2. copyNo from track position

```
G4int nx = G4int( (localPoint.x()+fContainerWallX+kCarTolerance)/(fVoxelHalfX*2.) );
G4int ny = G4int( (localPoint.y()+fContainerWallY+kCarTolerance)/(fVoxelHalfY*2.) );
G4int nz = G4int( (localPoint.z()+fContainerWallZ+kCarTolerance)/(fVoxelHalfZ*2.) );

G4int nyz = nz*fNoVoxelY+ny;
std::multimap<G4int,G4int>::iterator ite = fFilledIDs.begin();
advance(ite,nyz);
std::multimap<G4int,G4int>::iterator iteant = ite; iteant--;
G4int copyNo = (*iteant).first + 1 + ( nx - (*ite).second );
```

Automatic phantom-volume intersection

User writes a user command:

1. Intersection is calculated
2. Partial DICOM file is written
3. Job ends

/readDICOM/intersectWithG4Volume VOLUME_NAME

- Extract G4VSolid from existing volume

/readDICOM/intersectWithUserVolume POS_X POS_Y POS_Z ROTANG_X ROTANG_Y ROTANG_Z SOLID_TYPE SOLID_PARAM1 (SOLID_PARAM2, ….)

- Build a G4VSolid using the utilities of ASCII geometry format
  - All Geant4 solid shapes available (booleans, tellesated, twisted, …)

Tests

Take a 200x200x10 voxel geometry and intersect it with a volume

- Read the new DICOM file
- Send tracks everywhere
- Plot for each Z layer X vs Y position when an interaction occurred in the "phantom" volume

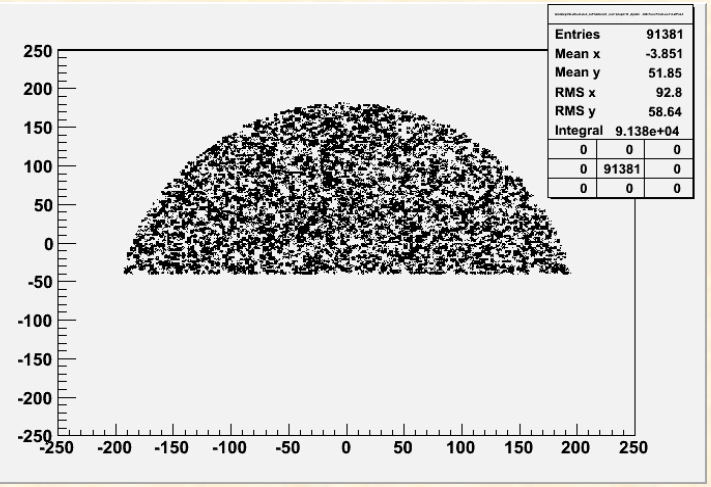
```
# only steps in phantom
/gamos/filter InPhantomF GmInLogicalVolumeFilter phantom
# classify by the Z position (a different histogram for each Z interval)
/gamos/classifier ZposC GmClassifierByNumericData FinalPosZ  -100.  100.  200./10
# select what to plot
/gamos/setParam GmStepDataHistosUA_InPhantomF_ZposC:DataList FinalPosX.vs.FinalPosY
# build histograms each step using filter and classifier
gamos/userAction GmStepDataHistosUA InPhantomF ZposC
```

Tests

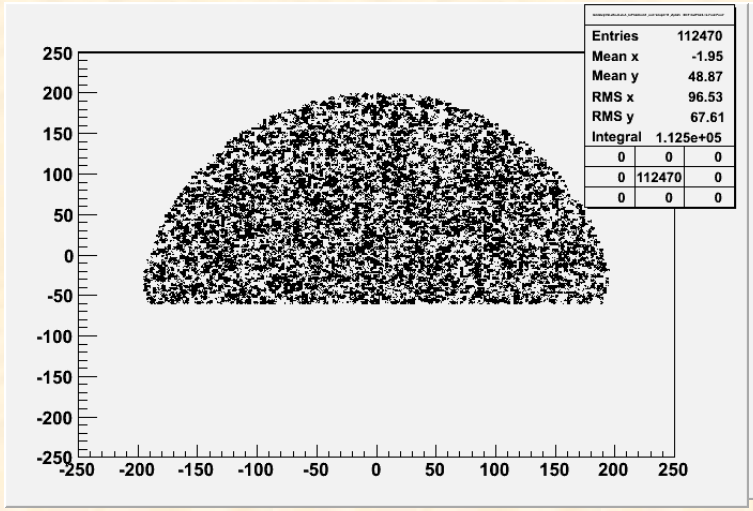
Intersect with a **rotated cylinder**

/gamos/readDICOM/intersectWithUserVolume 0. 0. 0. 45.*deg 0. 0. TUBE 0. 200. 100.

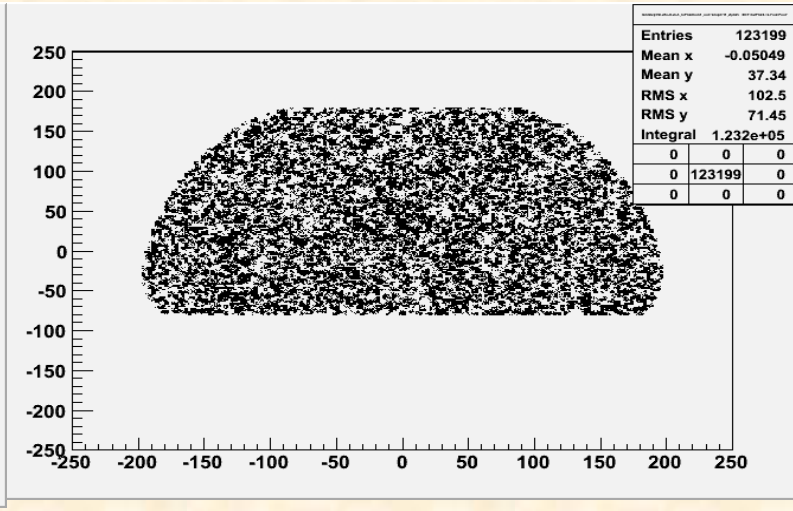
-100 < Z < -80



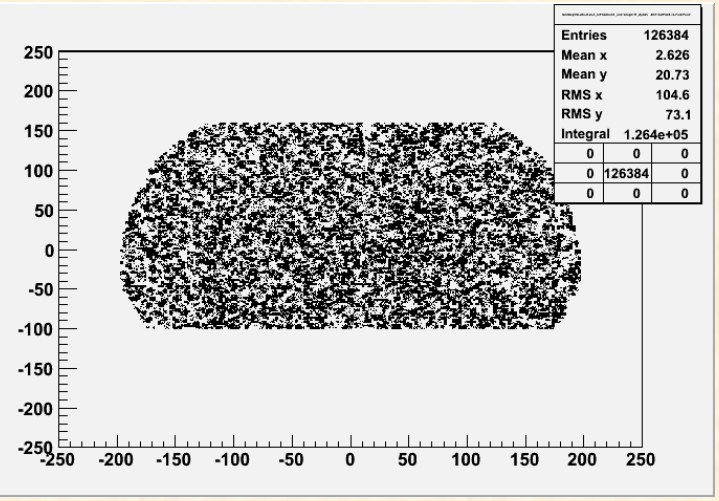
-80 < Z < -60



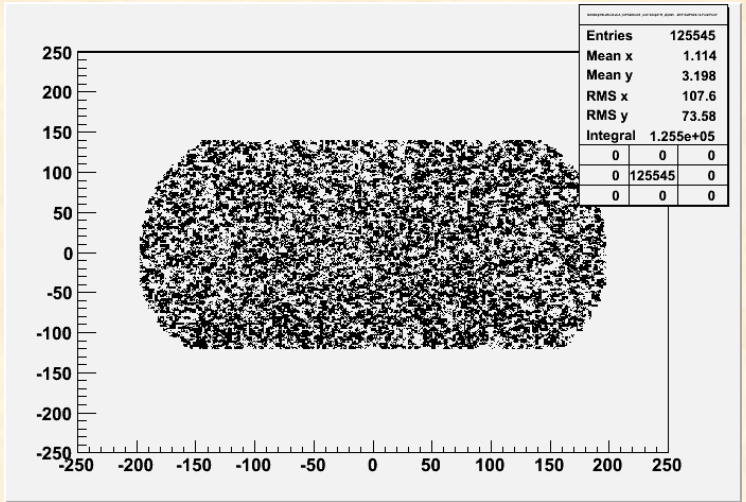
-60 < Z < -40



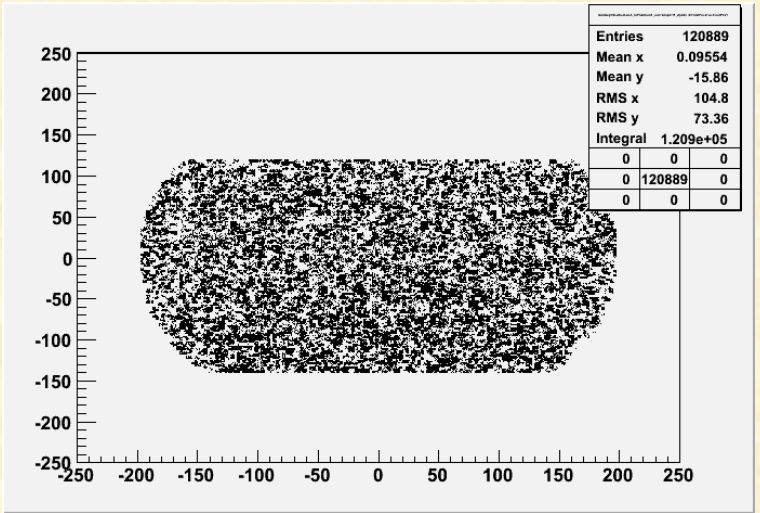
-40 < Z < -20



-20 < Z < 0



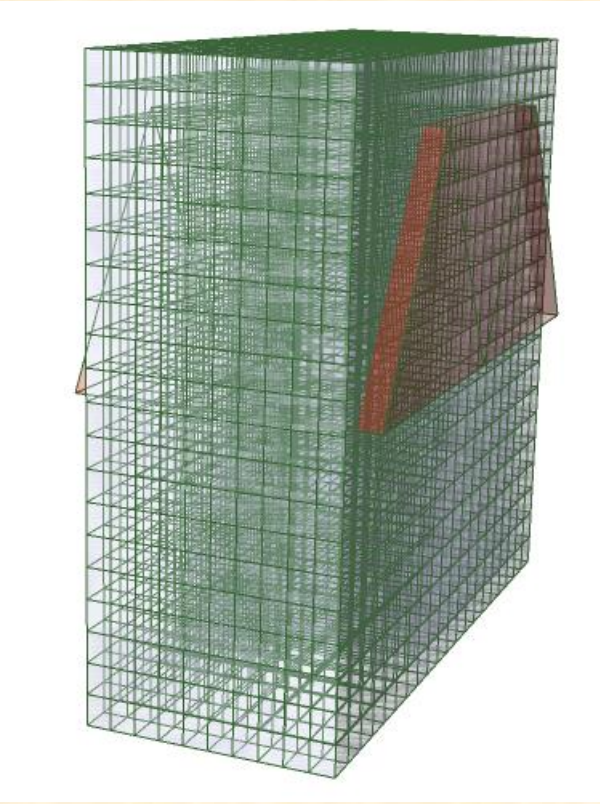
-0 < Z < 20

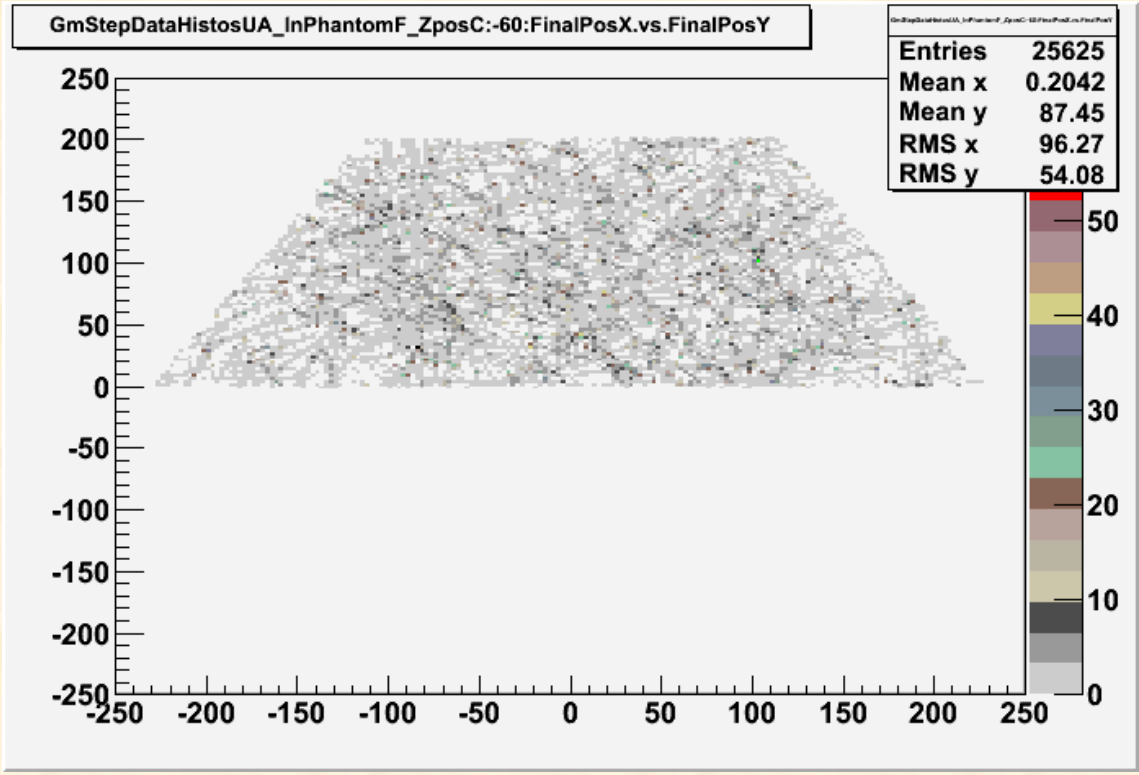


Tests

Intersect with a **polyhedra**

/gamos/readDICOM/intersectWithUserVolume 0. 0. 0. 0.*deg 0. 0. POLYHEDRA 0. 180.*deg 3
2 -100. 0. 200. 100. 0. 200.





Status

☺ G4PartialPhantomParameterisation option 1 (does not support non-contiguous voxels)

☺ Automatic intersection utility for any solid shape in Geant4 (option 1)

☹ Test for CPU and memory performance

- Test performance of containers (std::multimap<G4int,G4int>)
  - 10 M entries std::multimap<G4int,G4int>
    - 45 sec to fill
    - 307 Mg memory
    - But very fast search

😐 G4PartialPhantomParameterisation option 2 (support non-contiguous voxels)

- As a separate class G4PartialNonContiguousPhantomParameterisation ?