1 Introduction to material model

The material inside tkLayoutis present inside the modules (i.e. silicon and cooling blocks), and outside the modules (i.e. cables or cooling pipes). Is possible to specify, in the configuration files, the material inside:

- modules;
- rods (the ladders);
- barrel's layers;
- endcap's disks;
- custom cylinders and disks (i.e. for the supports).

Is also possible to set material from every element:

- locally,
- exiting.

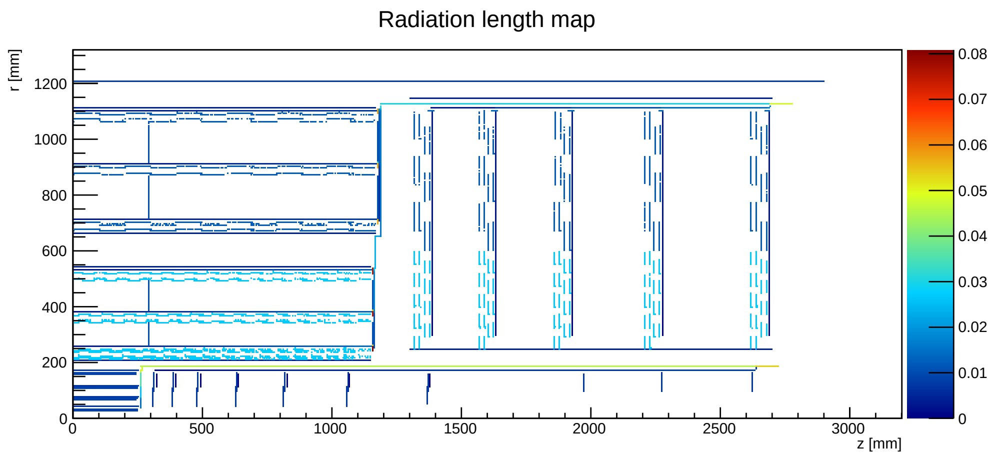

Figure 1: Material routing

Apart from the material is possible also to define *conversions* in special points: at the end of the layers, at the end of the disks, and in custom positon along the way.

In figure 1 is visible the material for the modules and the *sections* for the routing of the material, section 2 deal with the description in detail of how to build the configuration files and how the material is distributed, from an user point of view.

The material *routing algorithm* builds automatically all the necessary structures for keeping all the materials that not belongs to the modules. It builds *sections* on top of the layers, and on the right of the disks, then connect those sections for each barrel and endcap, then connect barrels and endcaps with the cable exit point (upper right point of tracker). Section 3 deal with the description in detail of the routing algorithm from a developer point of view.

2 User manual

2.1 Material routing effects

The first column is where the material is defined and if is defined with *service* true or false. The other columns are the different effect regarding the unit of measure of the material, for each cell: the first field is the destination volume; the second is the amount of material in grams; the third specify if the material is accumulating along the layer/disk; the fourth specify if the material is converted after the layer/disk or only stay in the layer/disk; the fifth if is possible or not to set the scaling on channels.

	Unit=g/m	Unit=mm	Unit=g
Module Service=false	Module $\times moduleLength$ No accumulation No conversion Scaling possible	Module $\times moduleSurface \times \rho$ (sensor surface) No accumulation No conversion Scaling possible	Module $\times 1$ No accumulation No conversion Scaling possible
Module in ring R[1] Service=true[2]	Following supports $S_{R+1} \ldots S_i \ldots S_N$ $\times numModules_R \times supportLength_i$ Accumulation Conversion(1:1 by default, with warning) Scaling possible	Following supports $S_{R+1} \ldots S_i \ldots S_N$ $\times numModules_R \times supportSurface_i \times \rho$ Accumulation Conversion(1:1 by default, with warning) Scaling possible Deprecated warning	Error
Rod (barrel[3]) Service=false	All supports $S_1 \ldots S_i \ldots S_N$ $\times numModules_1 \times supportLength_i$ No accumulation No conversion Scaling not possible	All supports $S_1 \ldots S_i \ldots S_N$ $\times supportSurface_i \times \rho$ No accumulation No conversion Scaling not possible	All supports $S_1 \ldots S_i \ldots S_N$ $\times numModules_1 \times \frac{supportLength_i}{\sum_{j=1}^{N} supportLength_j}$ No accumulation No conversion Scaling not possible
Rod (barrel) Service=true[2]	All supports $S_1 \ldots S_i \ldots S_N$ $\times numModules_1 \times supportLength_i$ No accumulation Conversion Scaling not possible	All supports $S_1 \ldots S_i \ldots S_N$ $\times supportSurface_i \times \rho$ No accumulation Conversion Scaling not possible Deprecated warning	Error
Layer/Disk Service=false	All supports $S_1 \ldots S_i \ldots S_N$ $\times supportLength_i$ No accumulation No conversion Scaling not possible	All supports $S_1 \ldots S_i \ldots S_N$ $\times supportSurface_i \times \rho$ No accumulation No conversion Scaling not possible	All supports $S_1 \ldots S_i \ldots S_N$ $\times \frac{supportLength_i}{\sum_{j=1}^{N} supportLength_j}$ No accumulation No conversion Scaling not possible
Layer/Disk Service=true[2]	All supports $S_1 \ldots S_i \ldots S_N$ $\times supportLength_i$ No accumulation Conversion Scaling not possible	All supports $S_1 \ldots S_i \ldots S_N$ $\times supportSurface_i \times \rho$ No accumulation Conversion Scaling not possible Deprecated warning	Error

	Modules	Cylind. service sections	disk service section
Length	Local y	Δz	Δr
Surface	Sensor surface	$2\pi r \Delta z$	$\pi({r_2}^2 - {r_1}^2)$

[1]of N rings
[2]may be converted by station
[3]line of one module per ring with same ϕ

3 Developer manual

3.1 General algorithm description

3.2 Classes description

3.2.1 namespace insur

- **MaterialBudget**
  This class integrates information from a *Tracker* and an *InactiveSurface* instance with a collection of *ModuleCap* instances to provide the full material budget of a tracker.
  Its main function accepts an instance of a material calculator as an input parameter in order to use that calculator's more specialised functions to assign mixtures of materials to the different categories of volumes in the tracker geometry. Afterwards, each individual volume is ready to calculate its total mass, its radiation length and its interaction length, which is also taken care of by the calculator class. A material budget that has been filled in this way can then be passed on to be analysed by an instance of an *Analyzer* class.

- **MaterialProperties**
  This is the base class for collections of properties related to the material budget.
  It encapsulates the main parameters of interest, namely the overall density, radiation length and interaction length of a tracker element, as well as the influence of the materials that make up this building block. Access functions are provided where appropriate. But unless the object is cloned the overall parameters should typically be calculated from a list of materials and their properties rather than set explicitly. Some of the access functions for individual materials may throw exceptions if the requested material does not appear on the list.

- **MaterialTable**
  Essentially, this is a collection class for *MaterialRow* instances.
  It provides access functions for individual entries and a couple of bookkeeping calls.
  The access functions may trow exceptions if an entry doesn't exist or if an index is out of range.

- **BaseMaterial**

- **MaterialTable2**

- **ElementaryMaterial : public BaseMaterial**
- **CompositeMaterial : public BaseMaterial**
- **InactiveElement : public MaterialProperties**
  This is the base class for the elements that make up the inactive surfaces.
  Since all inactive elements are simplified to tube shapes in the geometrical model, the geometry parameters have been packed into the base class. All of these parameters apply to descendants of a different shape as well, though, because they describe relations between the object and the origin, not between points within the object.
- **InactiveRing : public InactiveElement**
  The only thing that this class adds to its parent is a check that it is a ring rather than a tube.
- **InactiveSurfaces**
  This is the top-level container class for the inactive surfaces.
  It contains lists of all subgroups of inactive volumes: the services list and the supporting parts list.
  It provides access functions to them or their individual elements that typically copy a new element to its place at the end of the vector or return a reference to a requested element. It also stores the type of configuration (UP or DOWN) in a boolean flag. Some of the access functions to individual elements may throw an exception if the requested index is out of range.
- **InactiveTube : public InactiveElement**
  The only thing that this class adds to its parent is a check that it is a tube rather than a ring.
- **MatCalc**
  The MatCalc class provides the core material assignment algorithm for a given tracker geometry.
  Once its internal data structures have been initialised from the material config file by the *MatParser* class, it uses that information, combined with the geometry and position of an individual tracker element, to set the local and exiting materials vector that element before getting it to calculate its overall mass, its radiation length and its interaction length. The tracker geometry is ready for further study afterwards. Some of the access functions for various internal list elements may return an exception if the requested element does not exist on the list.

3.2.2 namespace material

- **MaterialObject : public PropertyObject**
- **MaterialObject::ReferenceSensor : public PropertyObject**
- **MaterialObject::MaterialObjectKey**
- **MaterialObject::Element : public PropertyObject**
- **MaterialObject::Component : public PropertyObject**
- **MaterialObject::Materials : public PropertyObject**
- **MaterialSection : public MaterialObject**
- **MaterialStation : public MaterialSection**
- **MaterialTab : public MaterialTabType**
- **Materialway**
- **Materialway::RodSectionsStation**
- **Materialway::Section**
- **Materialway::Station : public Materialway::Section**
- **Materialway::Boundary**
- **Materialway::OuterUsher**
- **Materialway::InnerUsher**
- **Materialway::ModuleUsher**
- **ConversionStation : public MaterialObject**
- **ConversionStation::Inoutput : public PropertyObject**
- **ConversionStation::Conversion : public PropertyObject**
- **WeightDistributionGrid : public WeightDistributionGridMap-Type**