New GEAR Classes (1)

ModularTPCParameters

Inherits GearParameters ⇒ User can add his/her own parameters

virtual const **TPCModule** &	**getModule** (int ID) const =0 *Returns module with given moduleID.*
virtual int	**getNModules** () const =0 *Returns number of modules in this TPC (endplate).*
virtual const **TPCModule** &	**getNearestModule** (double c0, double c1) const =0 *Returns nearest module to given coordinates.*
virtual double	**getMaxDriftLength** () const =0 *The maximum drift length in the TPC in mm.*
virtual bool	**isInsideModule** (double c0, double c1) const =0 *True if coordinate (c0,c1) is within any module.*
virtual bool	**isInsidePad** (double c0, double c1) const =0 *True if coordinate (c0,c1) is within any pad, on any module.*
virtual **GlobalPadIndex**	**getNearestPad** (double c0, double c1) const =0 *Returns globalPadindex Object for nearest pad to given coordinates.*
virtual const std::vector< double > &	**getPlaneExtent** () const =0 *Extent of the sensitive plane - [xmin,xmax,ymin,ymax] CARTESIAN or [rmin,rmax,phimin,phimax] POLAR.*
virtual int	**getCoordinateType** () const =0 *Returns coordinate type as an int (see PadRowLayout2D::CARTESIAN, PadRowLayout2D::POLAR).*
virtual const std::vector < **TPCModule** * > &	**getModules** () const =0 *Returns vector of all modules in this TPC (endplate).*

universitätbonn

New GEAR Classes (2)

TPCModule

Inherits GearParameters ⇒ User can add his/her own parameters

Inherits PadRowLayout2D ⇒ Has all the functionality of a pad layout, but returns global TPC coordinates

Additional functions:

virtual bool	**isInsideModule** (double c0, double c1) const =0 *True if global coordinate (c0,c1) is within this modules area of amplification/interest.*
virtual double	**getReadoutFrequency** () const =0 *The readout frequency in Hz.*
virtual const **Vector2D** &	**getOffset** () const =0 *Returns the offest of Modules origin from the global origin A vector from Global (0,0) to module (0,0).*
virtual double	**getAngle** () const =0 *Returns the rotation of the module, in Rads, with respect to the modules internal origin.*
virtual double	**getDistanceToPad** (double c0, double c1, int index) const =0 *Returns the distance from a global coodinate (c0,c1), to a given pad's nearest boundery; (c0,c1,index).*
virtual double	**getDistanceToModule** (double c0, double c1) const =0 *Returns distastance from a global coodinate (c0,c1), to the module's nearest boundery; (c0,c1).*
virtual int	**getModuleID** () const =0 *Returns module ID.*
virtual bool	**isOverlapping** (**TPCModule** *testThisModule) const =0 *Returns True if this and The given module * overlap pad regions Note: overlaping sensitive regions is ok, Just no two pads with shared physical space.*
virtual double	**getBorderWidth** () const =0 *Returns the amount by which the pad plane has been extended to produce the module plane.*

universitätbonn

TPCModule (2)

Changes wrt. PadRowLayout2D:

virtual const std::vector<double>& gear::TPCModule::getPlaneExtent ( ) const throw (NotImplementedExceptior

Inherited from **PadRowLayout2D**, but not a useful measure in global coordinates, use **getModuleExtent()** instead.

Always throws **NotImplementedException** since a **TPCModule** is a **PadRowLayout2D**, and the **PadRowLayout2D::getPlaneExtent()** gives the extend of the comlete sensitive plane (without dead space). As this is not possible in global coordinates we decided to invalidate this function and introduce **getModuleExtent()**, because we did not want to reinterpert the functionality.

virtual const std::vector<double>& gear::TPCModule::getModuleExtent ( ) const [pure virtual]

Maximal extent of the sensitive plane, defined relative to global origin - [xmin,xmax,ymin,ymax] CARTESIAN or [rmin,rmax,phimin,phimax] POLAR, may contain dead space due to conversion from local to global coordinate system.

The sensitive plane / amplification plane is the pad plane extended by a border (see **getBorderWidth()**). This allows the gas amplification structure to be larger than the pad plane, which in the case of a resistive coating will also produce signal on the pad plane for electrons arriving within this border.

Note: Amplification planes may overlap, pad planes may not.

universitätbonn

XML parser syntax

```
<detector name="TPC" geartype="ModularTPCParameters">
  <numberOfModules value="8" />
  <moduleIDStartCount value="20">

  <maxDriftLength  value="80.0" />
  <coordinateType  value="cartesian" />

  <module[*]>
    <readoutFrequency value="2.0e+07" />

    <PadRowLayout2D type="FixedPadSizeDiskLayout"  rMin="386." rMax="1626."
    padHeight="6." padWidth="2." maxRow="206" padGap="0.">
      <phiMin value="4.12318505419417 9844e-4"/>
      <phiMax value="0.784985844892028861"/>
    </PadRowLayout2D>
    <enlargeActiveAreaBy value="1."/>
  </module[*]>

  <!-- module[0] is not rotated or shifted -->

  <!-- rotate each of the following modules by 1/8 of a circle -->
  <module[1]>
    <angle value="0.7853981633974482 79" />
  </module[1]>

...

</detector>
```

Status

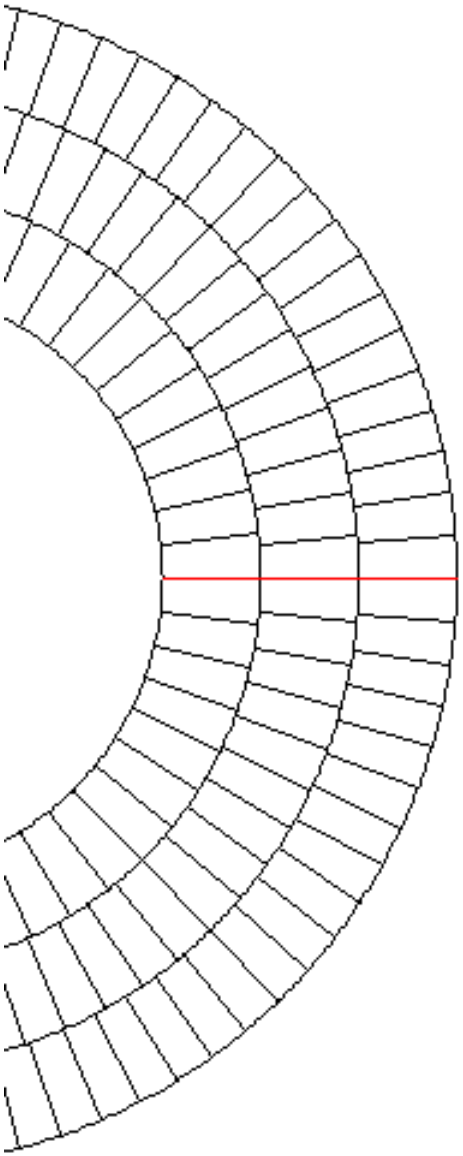
Status:

- Class interfaces are defined.
- Class implementations exist and compile, but no functionality yet (coordinate conversion throws NotImplementedException)
- XML parser syntax is defined
- XML parser is implemented, compiles but completely untested

ToDo:

- Finish class implementations
- Extend functionality of existing pad layouts

FixedPadSizeDiscLayout



- All pads in all rows have the same width (in the middle of the pad row)
- At phi = 0 all pads are aligned symmetrically
- The effective angular pitch is not exactly $2\pi \frac{padWidth}{circumference}$ but $2\pi/nPads$, where $nPads = \lfloor \frac{padWidth}{circumference} \rfloor$
- Pad plane is always a full circle

This is not sufficient!
Please let us know what you want to describe so we can consider it when designing the extension!