Geometry Access in DANA

David Lawrence JLab

May 23, 2008

Issue: How does one access geometry information in the reconstruction code without hardwiring it?

DGeometry class

- Provides an interface to the underlying *JGeometry* object
- A single *DGeometry* object is used for all threads/ factories needing to access the geometry it represents
- Multiple *DGeometry* objects can exist in a single process if it is analyzing events corresponding to different geometries.
- Provides both a ***low-level*** and a ***high-level*** API for accessing the geometry info.

Telling *JANA* where the geometry information is stored

- Set the *JANA_GEOMETRY_URL* environment variable to the location of the geometry:

```
setenv JANA_GEOMETRY_URL xmlfile:///${HALLD_HOME}/src/programs/Simulation/hdds/main_HDDS.xml
```

  - URL specifies top-level XML file. Other XML files that are referenced in it are automatically read in by the underlying parser (Xerces)
  - Using a URL in this way allows the possibility to change the mechanism through which the geometry is obtained in the future (e.g. http, database, …)

Getting a pointer to *DGeometry*

- A pointer is obtained through the *DApplication* object:

```
47  jerror_t DX_factory::brun(JEventLoop *loop, int runnumber)
48  {
49     DApplication *dapp = dynamic_cast<DApplication*>(loop->GetJApplication());
50     const DGeometry *dgeom = dapp->GetDGeometry(runnumber);
51
52        // ... use dgeom ...
```

Low-level API: using *xpath**

- *xpath* is a commonly used syntax for specifying a specific tag or attribute in an XML string.

```
<A>
    <B type="3">
        <C id="1"></C>
    </B>
    <B type="4”>
        <C id="2"></C>
    </B>
</A>
```

The following xpath can be used to specify the “type” attribute of the tag=“B” node that has a subtag “C” with an “id” attribute equal to “2” in the XML snippet on the left.

```
/A/B/@type/C[@id="2"]
```

*Full *xpath 1.0* specification not supported. Only an *xpath*-like syntax for now

A more practical example

xpath for extracting radiation length of Oxygen gas from the XML below

```
//materials/element[@name='Oxygen']/real[@name='radlen']/@value
```

```
<materials version="3.0"
           date="2006-11-22"
           author="R.T. Jones"
           specification="v1.0" >

  <element name="Oxygen" symbol="O" z="8" a="15.9995">
        <real name="density"    value="0.00133" unit="g/cm^3" />
        <real name="radlen"     value="34.24"   unit="g/cm^2"   />
        <real name="collen"     value="63.2"    unit="g/cm^2"   />
        <real name="abslen"     value="91.0"    unit="g/cm^2"   />
        <real name="dedx"       value="1.801"   unit="MeV/g/cm^2"
  </element>

</materials>
```

Constructing an appropriate *xpath*

- The *JANA* tool *jgeomread* can be used to both 1.)test *xpath*s and 2.)provide guidance as to what *xpath*s are available in the current XML string.
- To get a list of ***all*** *xpath*s, use the "-L" argument.
- To include ***all*** attributes in the listing, use the "-a" argument

```
fwing-dhcp13:~> jgeomread -L -a | grep "Oxygen" | grep "radlen"
/HDDS[@specification='v1.0' and @xmlns='http://www.gluex.org/hdds' and @xmlns:xsi='http://www.w3.org/
2001/XMLSchema-instance' and @xsi:schemaLocation='http://www.gluex.org/hdds HDDS-1_0.xsd']/
Material_s/materials[@author='R.T. Jones' and @date='2006-11-22' and @specification='v1.0' and
@version='3.0']/element[@a='15.9995' and @name='Oxygen' and @symbol='O' and @z='8']/real
[@name='radlen' and @unit='g/cm^2' and @value='34.24']
```

```
//materials/element[@name='Oxygen']/real[@name='radlen']/@value
```

Low-level methods in *DGeometry*

- Access is similar to other *JANA* access methods, using a *Get(…)* method
- First argument is an *xpath* specifying the desired attribute(s).
- Second argument is reference to container to hold result

```
56 bool Get(string xpath, string &sval) const;
57 bool Get(string xpath, map<string, string> &svals) const;
58 template<class T> bool Get(string xpath, T &val) const;
59 template<class T> bool Get(string xpath, vector<T> &vals, string delimiter=" ") const;
60 template<class T> bool Get(string xpath, map<string,T> &vals) const;
```

Example using low-level API

To get the atomic mass of Antimony, one could do the following:

```
63  double a;
64  dgeom->Get("'//hdds:element[@name='Antimony']/@a", a);
65  // a now equals 121.75
```

To get all of the attributes of Antimony, one could do the following:

```
67  map<string,string> attr;
68  dgeom->Get("'//hdds:element[@name='Antimony']", attr);
69  // attr now contains the following entries:
70  // a       121.75
71  // name    Antimony
72  // symbol  Sb
73  // z       51
```

High-level API using *DGeometry*

DGeometry contains several “convenience” methods. Most users will want to use these.

```
68  bool GetFDCZ(vector<double> &z_wires) const; ///< z-locations for each of the FDC wire planes in cm
69  bool GetFDCStereo(vector<double> &stereo_angles) const; ///< stereo angles of each of the FDC wire layers
70  bool GetFDCRmin(vector<double> &rmin_packages) const; ///< beam hole size for each FDC package in cm
71  bool GetFDCRmax(double &rmax_active_fdc) const; ///< outer radius of FDC active area in cm
72
73  bool GetCDCOption(string &cdc_option) const; ///< get the centralDC_option-X string
74  bool GetCDCCenterZ(double &cdc_center_z) const; ///< z-location of center of CDC wires in cm
75  bool GetCDCAxialLength(double &cdc_axial_length) const; ///< length of CDC axial wires in cm
76  bool GetCDCStereo(vector<double> &cdc_stereo) const; ///< stereo angle for each CDC layer in degrees
77  bool GetCDCRmid(vector<double> &cdc_rmid) const; ///< Distance of the center of CDC wire from beamline for
78  bool GetCDCNwires(vector<int> &cdc_nwires) const; ///< Number of wires for each CDC layer
79  bool GetCDCEndplate(double &z,double &dz,double &rmin,double &rmax) const;
80
81  bool GetBCALRmin(double &bcal_rmin) const; ///< minimum distance of BCAL module from beam line
82  bool GetBCALNmodules(unsigned int &bcal_nmodules) const; ///< Number of BCAL modules
83  bool GetBCALCenterZ(double &bcal_center_z) const; ///< z-location of center of BCAL module in cm
84  bool GetBCALLength(double &bcal_length) const; ///< length of BCAL module in cm
85  bool GetBCALDepth(double &bcal_depth) const; ///< depth (or height) of BCAL module in cm
```

Documentation on the wiki

A HOWTO describing all of this in more detail can be found on the web at:

http://www.jlab.org/Hall-D/software/wiki/index.php/HOWTO_Access_Geometry_Information_from_JANA/DANA

i.e.
GlueX wiki: Offline Software->HOWTO Access Geometry Information from JANA/DANA