simpleSpectrometer and some Geant4 insights :s

Bino
BPM spectrometer meeting
16 november 2006
UCL

spectrometerDipoleFieldMap

Loads a fieldmap file, format :

```
# Dubna Fieldmap - N. Morozon, S. Kostromin
# Simulation of 10D37 magnet using TOSCA (N. Morozov)
# All positions are in mm, (z,x) = (0,0) is center of magnet
# Magnetic field values are in Gauss
# z(mm)     B(x=0mm) B(x=5mm) B(x=10 ) B(x=15 ) B(x=20 ) B(x=25 ) B(x=30 ) B(x=35m) B(x=40 )
*           0.0      5.0      10.0     15.0     20.0     25.0     30.0     35.0     40.0
   0.0      1125.36  1125.36  1125.39  1125.42  1125.47  1125.54  1125.61  1125.69  1125.78
  10.0      1125.35  1125.36  1125.38  1125.42  1125.47  1125.53  1125.60  1125.69  1125.77
...
```

- Dynamic (using std::vector<double>) so no need to specify size in the file, just need to know the x-positions for each z position... hence the “*”- line, characterizing the x positions..
- Fieldmap specified only for a quadrant of the magnet... -z and -x are mirrored...
- Does this need to be more general ??

- Interpolation
  - Define closest 3x3 square, do first 3 parabolic interpolations (3 points = 1 parabola) in z direction (more points...) and then 1 parabolic interpolation in x

- Implements the spectrometerDipoleFieldMap::GetFieldValue( ) as a **concrete** implementation of the **pure virtual function** in the **abstract G4MagneticField class** from which it is derived
- `GetFieldValue( const G4double Point[4], G4double *Bfield )`
  - Point[x, y, z, time]
  - Returns Bfield[x, y, z]

- Currently little hack to **get to local coordinates** in the dipole volume as the GetFieldValue() routine gets queried with the global coordinates
  - Know how to do it properly (see BDSIM), just have to do it :)

- Sometimes the stepper queries the fieldmap with extreme positions, set field to 0 outside of the boundaries of the field (no extrapolation...)

SpectrometerMaterials

- Problem of defining vacuum :
  - No real vacuum exists, vacuum is just low density air :)
  - Get **G4_AIR** from NIST database (implemented in Geant) : G4NistManager
  - Set temperature (300 K) and pressure e.g. (1.e-9 mbar) and calculate density using ideal gas law & molar mass of air (~29 g/mol)
  - construct vacuum :

```
spec_Vacuum = new G4Material( "spec_Vacuum", dens, 1, kStateGas, temp, press );
spec_Vacuum->AddMaterial( spec_Air, 1. );  // ... a vacuum of air :)
```

- All materials will be in spectrometerMaterials
- Global `extern spectrometerMaterials* gMat;` object, instantiated at top of spectrometerMaterials.cc

SpectrometerOutput & the hits collection

At end of event, a hits collection is available in Geant, so need to derive a concrete class , **spectrometerUserEventAction**, from **G4UserEventAction** that implements

```
spectrometerUserEventAction::BeginOfEventAction( const G4Event * );
spectrometerUserEventAction::EndOfEventAction( const G4Event * );
```

Implement EndOfEventAction to do something with each hits collection, like e.g. Dump to output

```
// get the hits collections of this event
G4HCofThisEvent* hce = evt->GetHCofThisEvent();

// get the digit collections of this event
//G4DCofThisEvent* dc = evt->GetDCofThisEvent();

if ( ! hce ) {
  G4cerr << "*** unable to retrieve hits collections in event " << evt->GetEventID()
         << std::endl;
  return;
}
```

Each sensitive detector (BPM in this case) has a hits collection !!

```
// for each hit collection ( so each sensitive detector ), we dump the hits to the
// output
spectrometerBpmHitCollection *bpmHC;
spectrometerBpmHit           *bpmHit;

for ( G4int i = 0; i < hce->GetNumberOfCollections(); i++ ) {
  bpmHC = ( spectrometerBpmHitCollection* ) hce->GetHC( i );

  for ( G4int j = 0; j < bpmHC->entries(); j++ ) {
    G4cout << " - Hit collection " << i << " contains " <<  bpmHC->entries()
           << " entr" << (bpmHC->entries() > 1 ? "ies" : "y" ) << " : " << std::endl;

    bpmHit = (*bpmHC)[j];
```

$\longrightarrow$ Can do something here with the hit

```
  } /* end of loop over the hits in one collection */
```

$\longrightarrow$ So we can plug here also a routine that takes the events hce and spits to whatever

The filling of the hits collections happens for each SD in the ProcessHits routine, and the EndOfEvent routine (see spectrometerBpmSD::ProcessHits(...)

```
/**
   Let's store the hit, so construct a spectrometerBpmHit...
*/
G4String BpmName = hitCollectionName + "Hit";
spectrometerBpmHit *bpmHit = new spectrometerBpmHit( BpmName, x, y, z,
                                                     xPrime, yPrime, pdgType, id );

// insert the hit into the collection
bpmCollection->insert( bpmHit );
```

and spectrometerBpmSD::EndOfEvent(...) :

```
G4SDManager *SDman = G4SDManager::GetSDMpointer();
G4int HCID = SDman->GetCollectionID( hitCollectionName );
HCE->AddHitsCollection( HCID, bpmCollection );
```

So just would need then a **global gOuput pointer** which **gets pointed to the current output object** (corresponding with a file) for each run that gets created and destroyed at beginning and end of a run (spectrometerUserRunAction) and that has a routine

```
    gOuput->WriteHits( G4HCofThisEvent *hce )
```

that gets called at each spectrometerUserEventAction::EndOfEvent(...)

SpectrometerConfiguration

Based on tinyXML loader, example config file : ***proposal !!!*** :

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Energy spectrometer simulation configuration file -->
<spectrometer>
  <simulation>
    <author>Bino</author>
    <comment>This is just an example...</comment>
    <runs>10</runs>
    <events>100000</events>
  </simulation>
  <beam>
    <energy>28.5</energy>
    <particle>e-</particle>
  </beam>
  <optics> ... </optics>
  <components>
    <bpm name="BPM1" xpos="0." ypos="0." zpos="10.">
       <adcbits>14</adcbits>
    </bpm>
    <dipole name="M1" xpos="0." ypos="0." zpos="10.">
      <length>3.</length>
      <sign>1</sign>
      <fieldmap>10D37.MAP</fieldmap>
    </dipole>
    <bpm name="BPM1" xpos="0." ypos="0." zpos="5.">
      <adcbits>12</adcbits>
    </bpm>
  </components>
</spectrometer>
<!--   ....ooooOOOOOOOO "This is the end" OOOOOOOOoooo....  -->
```

category

variable

Specify components with mandatory attributes --> can generate error upon missing attribute

all other options can have default values...

Example code :

```
spectrometerConfig* cnf = new spectrometerConfig();
cnf->Load( "spectrometer.xml" );

double ebeam, betx;
int runs;
string particle;

cnf->Get( "beam", "energy",      &ebeam );
cnf->Get( "beam", "particle",    &particle );
cnf->Get( "simulation", "runs", &runs );

cnf->Get( "optics", "betx", &(betx) );

cout << "energy = " << ebeam << endl;
cout << "runs   = " << runs << endl;
cout << "particle = |" << particle << "|" << endl;
```

Never store values in real member variables of the config object : most flexible.

You define a new xml tag, and it is immediately available in program...

```
Get( category, name, <type> *variable )
```

Overloading a “Get” function in the config class that each time queries the XML DOM which is read in from the xml file upon creation of the object :

```
_dom = new TiXmlDocument( xmlfile );
_dom->LoadFile();
```

Private member of spectrometerConfiguration class

Need to instantiate a global spectrometerConfiguration object upon starting the program...

Need to think about the components... maybe store them in the configuration object as

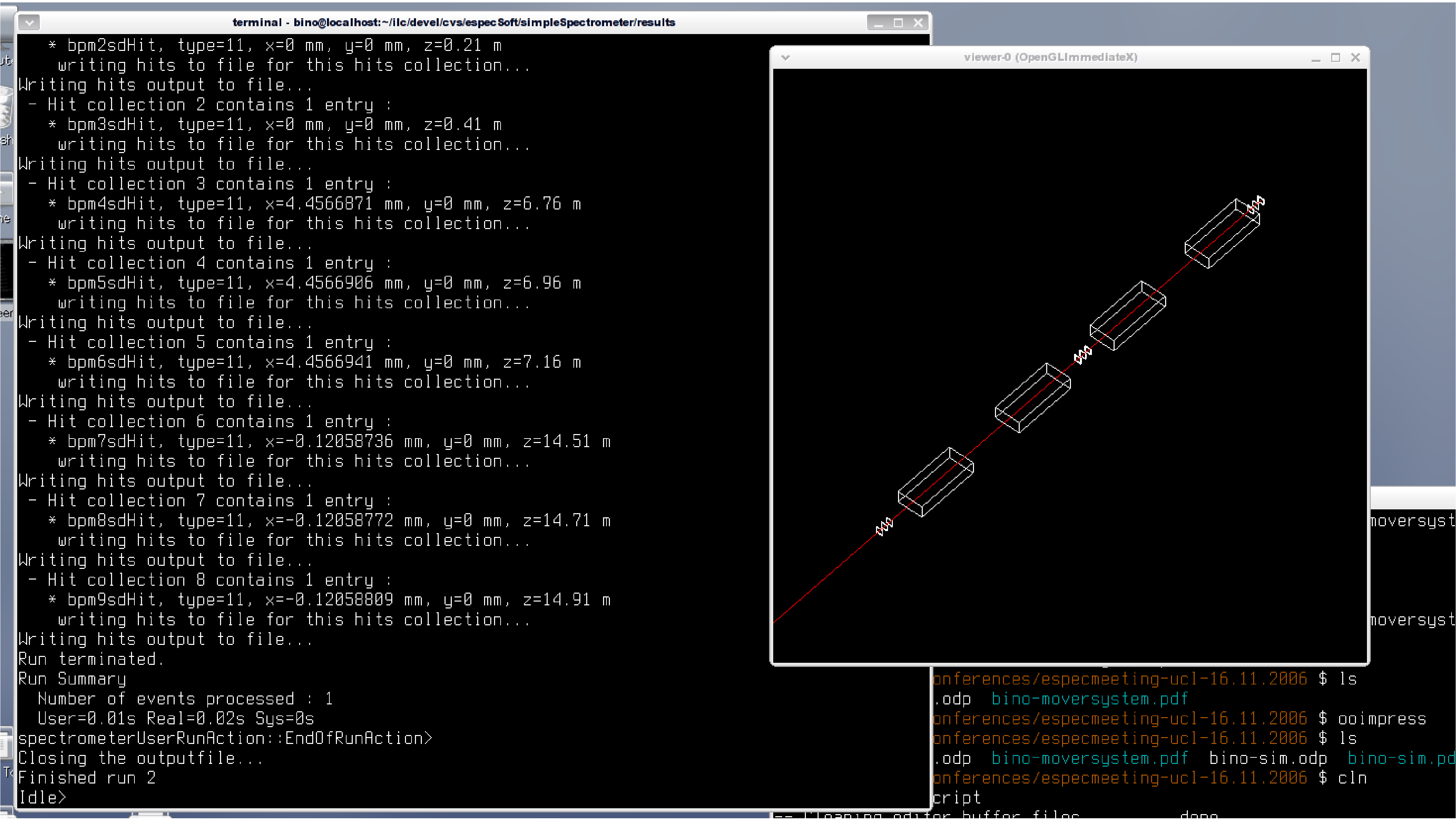
```
    std::vector<spectrometerBpmConfig>
    std::vector<spectrometerDipoleConfig>
```

where spectrometerBpmConfig and spectrometerDipoleConfig are then classes that represent the configuration of 1 BPM or 1 dipole magnet...

What else...

Implement somehow beampipe

think of stuff tomorrow... brain doesn't cooperate anymore....


terminal - bino@localhost:~/ilc/devel/cvs/especSoft/simpleSpectrometer/results
viewer-0 (OpenGLImmediateX)

```
   * bpm2sdHit, type=11, x=0 mm, y=0 mm, z=0.21 m
    writing hits to file for this hits collection...
Writing hits output to file...
 - Hit collection 2 contains 1 entry :
   * bpm3sdHit, type=11, x=0 mm, y=0 mm, z=0.41 m
    writing hits to file for this hits collection...
Writing hits output to file...
 - Hit collection 3 contains 1 entry :
   * bpm4sdHit, type=11, x=4.4566871 mm, y=0 mm, z=6.76 m
    writing hits to file for this hits collection...
Writing hits output to file...
 - Hit collection 4 contains 1 entry :
   * bpm5sdHit, type=11, x=4.4566906 mm, y=0 mm, z=6.96 m
    writing hits to file for this hits collection...
Writing hits output to file...
 - Hit collection 5 contains 1 entry :
   * bpm6sdHit, type=11, x=4.4566941 mm, y=0 mm, z=7.16 m
    writing hits to file for this hits collection...
Writing hits output to file...
 - Hit collection 6 contains 1 entry :
   * bpm7sdHit, type=11, x=-0.12058736 mm, y=0 mm, z=14.51 m
    writing hits to file for this hits collection...
Writing hits output to file...
 - Hit collection 7 contains 1 entry :
   * bpm8sdHit, type=11, x=-0.12058772 mm, y=0 mm, z=14.71 m
    writing hits to file for this hits collection...
Writing hits output to file...
 - Hit collection 8 contains 1 entry :
   * bpm9sdHit, type=11, x=-0.12058809 mm, y=0 mm, z=14.91 m
    writing hits to file for this hits collection...
Writing hits output to file...
Run terminated.
Run Summary
  Number of events processed : 1
  User=0.01s Real=0.02s Sys=0s
spectrometerUserRunAction::EndOfRunAction>
Closing the outputfile...
Finished run 2
Idle>
```

```
moversyst
moversyst
onferences/especmeeting-ucl-16.11.2006 $ ls
.odp  bino-moversystem.pdf
onferences/especmeeting-ucl-16.11.2006 $ ooimpress
onferences/especmeeting-ucl-16.11.2006 $ ls
.odp  bino-moversystem.pdf  bino-sim.odp  bino-sim.pd
onferences/especmeeting-ucl-16.11.2006 $ cln
cript
```

In summary... we can bend the beam using the fieldmap and obtain more or less the same deflection as Sergei's simulation...