GeneratorFilters: A package for generator event filtering in Athena Version in release 11.0.0 and later

Ian Hinchliffe (I_Hinchliffe@lbl.gov)

May 25, 2005

This package contains a set of algorithms that run from within Athena. The algorithms extract the generated Monte Carlo event from the transient event store (Storegate) and then test for various criteria. The base class. **GenFilter.cxx** extracts the event from Storegate and makes it available; all the other filter classes inherit from this.

The events are in HepMC format which is independent of the actual Monte Carlo generator. The same filter can therefore be used with Isajet, Herwig, Pythia *etc.* Uses are directed to the ElectronFilter.cxx which illustrates the basic logic and can be used as a template for preparing other classes. (Note that SampleFilter.cxx is easier to read as it does not use the base class, **users should not use SampleFilter as a template as their code will not then be protected against changes in the way the events are held in Storegate**.

First some properties settable from the jobOptions are declared and defaults set. **Note that the parameters are in ATLAS units, the energies and momenta are in MeV**

```
ElectronFilter::ElectronFilter(const std::string& name,
      ISvcLocator* pSvcLocator): GenFilter(name,pSvcLocator) {
  //----------------------------
  declareProperty("Ptcut",m_Ptmin = 10000.0); \\ 10 GeV
  declareProperty("Etacut",m_EtaRange = 3.0);
```

The initialization and loading of the event is done by the base class. The core of the algorithm is in

```
StatusCode ElectronFilter::filterEvent() {}
```

The event is accessed and each particle examined

```
// Loop over all events in McEventCollection
  McEventCollection::iterator itr;
  for (itr = m_cCollptr->begin(); itr!=m_cCollptr->end(); ++itr) {
    // Loop over all particles in the event
    HepMC::GenEvent* genEvt = (*itr)->pGenEvt();
    for(HepMC::GenEvent::particle_iterator pitr=genEvt->particles_begin();
        pitr!=genEvt->particles_end(); ++pitr ){
```

pitr is an iterator for the current particle. We examine its PDG Id (see the Review of Particle Properties for these codes) and its transverse momentum and rapidity. Note that since HepMC's GenParticle inherits from the the CLHEP particle class, the transverse momentum and rapidity are available.

```
  for(HepMC::GenEvent::particle_iterator pitr=genEvt->particles_begin();
      pitr!=genEvt->particles_end(); ++pitr ){
    if( ((*pitr)->pdg_id() == 11) || ((*pitr)->pdg_id() == -11) ){
      if( ((*pitr)->momentum().perp() >=m_Ptmin) &&
          fabs((*pitr)->momentum().pseudoRapidity()) <=m_EtaRange){
        return StatusCode::SUCCESS;
      }
```

If the particle is an electron (or positron) and the transverse momentum and rapidity are in the accepted range, we have found an electron that passes the requirement. Hence we return. The default is setFilterPassed(true) so we have passed the filter. If the loop runs with out finding a candidate then we fall to the end of the loop

```
      }
    }
  }
}
setFilterPassed(false);
return StatusCode::SUCCESS;
```

The code is set so that the filter failed.
The existing filters are as follows.

- ElectronFilter: looks for electron or positron above some pt and inside some rapidity range. Set properties via

  ```
  ElectronFilter.Etacut = 2.7
  ElectronFilter.Ptcut = 4000. \\4 GeV
  ```

- ZtoLeptonFilter: looks for a Z that decays to e^+e^- or $\mu^+\mu^-$. This was used in DC0

- MultiLeptonFilter: looks for e or μ of either sign. Set properties via

  ```
  MultiLeptonFilter.NLeptons = 4
  MultiLeptonFilter.Etacut = 2.7
  MultiLeptonFilter.Ptcut = 4000. \\4 GeV
  ```

- JetFilter: Looks for either cone or grid jets.

  ```
  JetFilter = Algorithm( "JetFilter" )
  JetFilter.JetNumber = 1
  JetFilter.EtaRange = 2.7
  JetFilter.JetThreshold = 17000.;  # Note this is 17 GeV
  JetFilter.GridSizeEta=2; # sets the number of (approx 0.06 size) eta
  JetFilter.GridSizePhi=2; # sets the number of (approx 0.06 size) phi cells
  #the above pair are not used if cone is selected
  JetFilter.JetType=FALSE; #true is a cone, false is a grid
  JetFilter.ConeSize=0.4; #cone size (not used if grid is selected)
  ```

- PhotonFilter

```
PhotonFilter.NPhotons = 2
PhotonFilter.Etacut = 2.5
PhotonFilter.Ptcut = 10000. \\4 GeV
```

- BSignalFilter. The interested user should read the documentation of the Generators/PythiaB package to understand the functionality of this filter.

The filters are used in an Athena seqence as in the following example

```
theApp.DLLs  += [ "Herwig_i" ]
theApp.DLLs  += [ "TruthExamples" ]
theApp.DLLs += [ "HbookCnv" ]
theApp.DLLs += [ "GeneratorFilters" ]
theApp.DLLs += ["GaudiAlg"]
theApp.HistogramPersistency = "HBOOK"
theApp.TopAlg = ["Sequencer/Generator"]
Generator.Members = ["Herwig", "ElectronFilter", "HistSample"]
HistogramPersistencySvc.OutputFile  = "herwig.hbook";
```

If the ElectronFilter passes then HistSample is executed and the event ends up in the histogram. If it fails then HistSample is not executed. A more complicated example showing how to write out filtered events can be found in Generators/GeneratorFilters/share/JobOptions.pythia.higgs.filter.py

The base class keeps account of the number of passed and failed events and reports this at output. You can stop the processing when a userdefined number of events has passed. This is done via the commmand by setting the variable TotalPassed. For example,

```
ElectronFilter.TotalPassed=1200
```

will stop cause the athena to stop once 1200 event have passed the filter. To use this the number of requested events must be big enough so that at least TotalPassed will occur.