STAR MC Filter

V. Perevoztchikov
Brookhaven National Laboratory,USA

What is the problem?

On S&C meeting on Wed, 2008-05 was discussed the problem of filtering:

http://drupal.star.bnl.gov/STAR/system/files/fastMCjan.pdf

So, it is needed 1000M Pythia events, with rejection 1/750. Rejection after GEANT is not realistic. Filtration before Geant is needed.

- Pythia has a filter for Pt of partons. Too simple;
- STARSIM has a filter. It is based on ranges of Pt, Eta ,Z etc… Too simple as well

To solve this problem in STARSIM were implemented 3 filters:

- Inside of Pythia, parton level, before decays;
- Inside of Geant, before tracking;
- Inside of Geant, after tracking, before the output.

StMCFilter base class

StMCFilter base class is :

- A user interface. User must overload functions:
  - RejectEG (StGenParticles &) – called inside Pythia. Interaction point in (0,0,0); EG == EventGenerator
  - RejectGT (StGenParticles &) – called inside Geant before tracking. Interaction point generated by Geant; GT==GeantTracking
  - RejectGE (StGenParticles &) – called inside Geant after tracking; GE==GeantEnd
- All three methods have exactly the same input arguments;
- User must provide the unique name of the his filter. Selection of filter is based on this name.
- Apart of that in this base class all the machinery of connection to Pythia and Geant is hidden
- At the end Finish() is called. Print statistics. Could be overloaded

Input argument of RejectXX

An argument of RejectXX(StGenParticles &Ptl)

Class StGenParticles is a container class containing StGenParticle objects.

Each object represents one particle(track). User methods of StGenParticles class:

- Ptl.Size() – number of particles;
- Ptl(index) – pointer to particle (StGenParticle*)
- Ptl.Print() – print all particles;
- Ptl.Print(char* tit) – print container

Input argument of RejectXX(2)

Class StGenParticle ideologically is based on HEPEVT standard

http://cepa.fnal.gov/psm/simulation/mcgen/lund/pythia_manual/pythia6.3/pythia6301/node39.html

Methods:

- int GetStatusCode; // 1 for final particle
- int GetPdgCode; //PDG particle code
- int GetGeaCode(); //Geant particle code
- StGenParticle *GetMother(int i); //mother particle(0=beam,1=target)
- StGenParticle *GetDaughter(int i); //daughter ith particle
- double GetCalcMass ();//calculated mass
- double GetMass(); //mass
- int GetNDaughters();

Input argument of RejectXX(3)

Methods continue:

- void Momentum(double p4[4]); //four momentum
- void Vertex(double v[3]) //vertex in cm;
- double Time(); //time in cm
- int IsPrimary(); //Is this particle a primary one?
- int IsFinal () //Is this particle a final one?

Input argument of RejectXX(4)

Additional methods:

- double R (); //Rxy of vertex
- double Rho (); //Rxyz of vertex
- double P (); // Full momentum
- double Pt (); //Transverse momentum
- double Energy();
- double Eta (); //Pseudo rapidity
- double Phi ();
- double Theta ();

Filter kumac commands

```
gexec $STAR_LIB/geometry.so
gexec $STAR_LIB/libpythia_6410t.so
 gexec $STAR_LIB/bpythia.so
 gexec $STAR_LIB/StMCFilter.so
GFILTER filterName
```

All three methods are called in a proper places. Method which is not overloaded, always returns zero (no rejection)

Example of user rejection

```
int StExampleFilter::RejectEG(const StGenParticles &ptl) const
{
//  ptl.Print("*************** In RejectEG ************** ");
// Condition: number of tracks etaGate[0]<eta<= etaGate[1] must be > etaGate[2]
const static double etaGate[3]={0.8,1.2, 3};
int n = ptl.Size(),  ntk=0;
  for (int i=0;i<n;i++) {
 const StGenParticle * tk = ptl(i); if (!tk)       continue;
    if (tk->GetStatusCode()!=1) continue;
    if (tk->Eta() < etaGate[0]) continue;
    if (tk->Eta() > etaGate[1]) continue;
    ntk++;
  }
  if (ntk<etaGate[2]) return 1;
  return 0;
}
```

Implementation

All the filter software now in StRoot/StMCFilter/

An example of filt.kumac also is there

An example of user filter class is StExampleFilter.h and .cxx

All the user filters MUST be in this directory as well.

If user need to test his filter and compiling in his local
StRoot/StMCFilter/ directory then line in kumac:

gexec $STAR_LIB/StMCFilter.so

Must be changed to:

Gexec .$STAR_HOST_SYS/lib/StMCFilter.so

Summary

The main features:

- All three methods are in the same named class
- When some production is made, then this class must not be modified. If needed, another class with the different name must be created.
- It is C++ class. When we will use VMC, the same class could be used for filtering;
- This class is ready and waiting for the user.