Subject: Re: FairPrimaryGenerator
Posted by MartinJGaluska on Fri, 02 May 2014 09:53:38 GMT
View Forum Message <> Reply to Message

As an addition to my last post (which I wrote on my cell phone and therefore wanted to keep short):

For the user-defined event filters which are foreseen for more complex filtering than on single particle multiplicities I had the idea to convert all TParticle which are generated from the event generators and convert them into RhoCandidate which is a PANDA specific solution and will not be adopted by other experiments. The idea is that PANDA users will be able to use all the tools from analysis also for writing user-defined specific event filters.

As this is PANDA specific we have developed the general event filter part first which can be used by all FairRoot based frameworks (I hope). We might add the missing functionality later, but time is an issue, of course. All we actually need is a FillList method that takes the TParticles from the generators and puts them into a TClonesArray of RhoCandidate with some added functionality to take wrong PID assignments into account. We might be able to reuse code which already exists for the "fast simulation", however. I assume the rest (like Combine and so on) will work without modifications for the user-defined event filters as they do for the analysis.

For now, all you can do without writing that FillList method and then your own event filter yourself is to use FairEvtFilterOnCounts. However, in your case you might not gain much if you only can require at least one neg. and one pos. charged particle in your events. If you can only filter on that criteria, you canonly filter out all events which do not have a positively or a negatively charged particle. I assume that there won't be many of those. With more detailed criteria you might be able to reduce the number of events better, however.

If you decide to do that you can put the following code into your sim macro:

```
FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
fRun->SetGenerator(primGen);

PndDpmDirect *Dpm= new PndDpmDirect(mom,1);
primGen->AddGenerator(Dpm);

// now add the event filters (in our case only non-veto filters)

// only accept events with at least one pos. and one neg. charged particle
FairEvtFilterOnCounts* min1pos1neg= new FairEvtFilterOnCounts("min1pos1neg");
//min1pos1neg->SetVerbose();
min1pos1neg->AndMinCharge(1,FairEvtFilter::kPlus); // events with min. 1 pos. charged particles will match and will be selected
min1pos1neg->AndMinCharge(1,FairEvtFilter::kMinus); // events with min. 1 neg. charged particles will match and will be selected
primGen->AddVetoFilter(min1pos1neg);// regular non-veto filter with higher priority than regular event filter
```

The same behavior can also be achieved with a veto filter:

```
FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
fRun->SetGenerator(primGen);

PndDpmDirect *Dpm= new PndDpmDirect(mom,1);
primGen->AddGenerator(Dpm);

// now add the event filters (in our case only a veto filter)

//veto events without pos. charged particles
FairEvtFilterOnCounts* noPlusVeto= new FairEvtFilterOnCounts("noPlusVeto");
//noPlusVeto->SetVerbose();
noPlusVeto->AndMaxCharge(0,FairEvtFilter::kPlus); // Actually it does not matter for the first filter whether it is set with And... or with Or...
noPlusVeto->OrMaxCharge(0,FairEvtFilter::kMinus); // events with max. 0 pos. OR max. 0 neg. charged particles will match and will be vetoed
primGen->AddVetoFilter(noPlusVeto);// veto filter with higher priority than regular event filter
```

Kind regards,
Martin

PS: I have simplified the example code for the veto filters.
