Subject: FairPrimaryGenerator
Posted by Klaus Götzen on Wed, 30 Apr 2014 11:28:22 GMT
View Forum Message <> Reply to Message

Hi,

when using an EvtGen decay file with a PDG code not in TDatabasePDG, FairPrimaryGenerators prints for every single decay some warning like

-W FairPrimaryGenerator: PDG code 88880 not found in database. This warning can be savely ignored

Is there a (verbose-)switch which allows to suppress this warning? If not, would it be possible to implement something to get rid of this warning?

Best and thanks,
Klaus

Subject: Re: FairPrimaryGenerator
Posted by Malte Albrecht on Wed, 30 Apr 2014 15:56:38 GMT
View Forum Message <> Reply to Message

Hi,

I have noticed that too, but ignored it up to now. If one tries to generate a lot of events (e.g. for background studies) the log files are getting huge, so it is urgent to get rid of that warning...
I have seen, that it is produced by a simple cout in FairPrimaryGenerator.cxx, isn't there a general Error-Logger in FairRoot that should be used for all warnings/errors!?

Maybe one could keep the warning, but just limit the output to lets say ten times (the first ten times it occurs)? That way one still could see that there is something wrong, but the log files dont get that large.

By the way: Is there a simple way to restrict the mass range e.g. of a pair of charged particles at the level of the generator? Then I dont have to produce tons of useless events for my background study...

Best regards,
Malte

Subject: Re: FairPrimaryGenerator
Posted by Klaus Götzen on Wed, 30 Apr 2014 16:08:47 GMT
View Forum Message <> Reply to Message

Hi Malte,

a temporary workaround is to add a line like

TDatabasePDG::Instance()->AddParticle("pbarpSystem0","pbarpSystem0",fIni.M(),kFALSE,0.1,0, "",88880);

for each of the unknown PDG codes in the beginning of the fast sim or analysis macro.

Concerning the generator filter, there actually is something in preparation for at least multiplicity cuts, but I don't believe that you will be able to cut on combined invariant masses on that level.

Best,
Klaus

Subject: Re: FairPrimaryGenerator
Posted by Malte Albrecht on Wed, 30 Apr 2014 16:47:15 GMT
View Forum Message <> Reply to Message

Hi Klaus,

ok, this workaround seems to work for now.
Ok... so there is no such filter on the generator level as I would like to use... then I will have to generate simply much more events. Whats the quota on /hera/panda? ;)

Thanks for your quick reply!
Malte

Subject: Re: FairPrimaryGenerator
Posted by MartinJGaluska on Wed, 30 Apr 2014 17:20:15 GMT
View Forum Message <> Reply to Message

Hi Malte,

we have been working on implementing a fairly general event filter.
Please have a look at this thread:
https://forum.gsi.de/index.php?t=msg&th=4135&start=0&
You can filter on multiplicities and on pretty much all properties of single particles using the FairEvtFilterOnCounts out of the box. For filtering on particle combinations you can implement your own specific filter by deriving from FairEvtFilter. You basically only have to implement the EventMatches function and can even logically combine your own filter with other filters.

Unfortunately, we have not yet written the tutorial with usage examples and so on. We are working on that.

Hope that helps,
Martin

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.

Subject: Re: FairPrimaryGenerator
Posted by MartinJGaluska on Sun, 25 May 2014 07:33:25 GMT
View Forum Message <> Reply to Message

A remark to my last post:

A filter on invariant masses is also available now. As the Rho package is used internally, this is a PandaRoot specific feature implemented in PndEvtFilter and derived classes (which is itself derived from the general FairRoot FairEvtFilter).

For more details see my post here:
https://forum.gsi.de/index.php?t=msg&th=4135&goto=16710&#msg _16710
