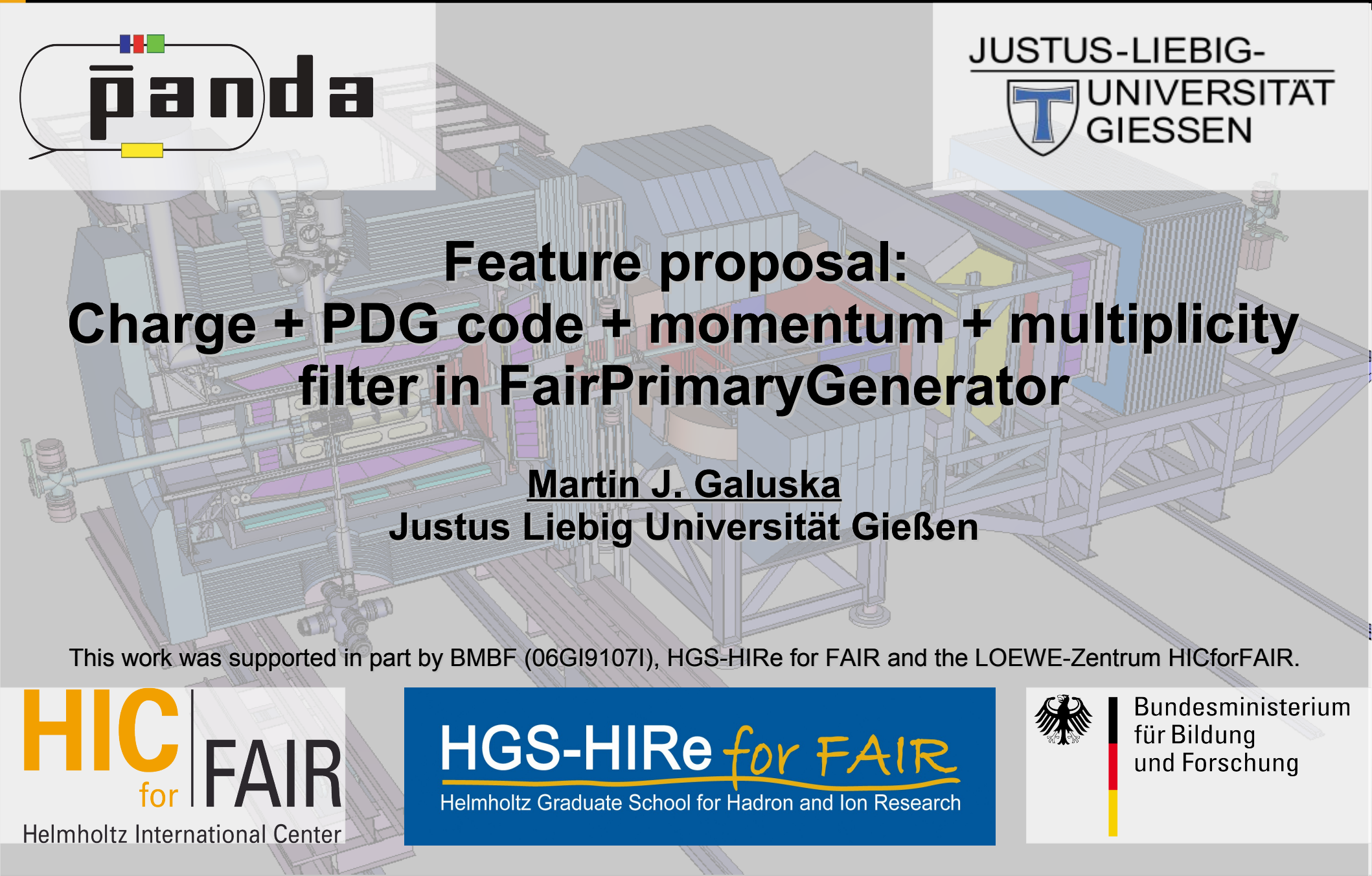

Feature proposal: Charge + PDG code + momentum + multiplicity filter in FairPrimaryGenerator

Martin J. Galuska
Justus Liebig Universität Gießen

This work was supported in part by BMBF (06GI9107I), HGS-HIRe for FAIR and the LOEWE-Zentrum HICforFAIR.

Motivation / History

- Our group wanted to run a **full PandaRoot simulation** of **specific events** which should be produced with PndDpmDirect.
- I started writing a pretty flexible **filter for PndDpmDirect**.
- Stefano suggested that such a filter could be **useful also for other event generators** and that we should think about **implementing it inside FairPrimaryGenerator**.
- Discussion on PandaRoot Forum:
  http://forum.gsi.de/index.php?t=msg&goto=15374&rid=1493&S=ad3eb0b32fc82b24f8fc5ea7cd91adae#msg_15374
  (you can skip the first two messages)

Example: Imagine that...

- ... we want to generate events using DpmDirect
- … we want to run the full PandaRoot simulation for very specific events
- … time is short
- … we are only interested in events that have:
  - at least 2 e- OR pi-
    - with a p_T within [1.0, 3.0] GeV/c
    - and with a p within [1.5, 3.0] GeV/c
  - at most 4 e+ OR pi+
    - with a p_z within [0.5, 4.0] GeV/c
  - at least 4 and at most 6 pos. Charged particles
    - with theta within $[20.0, 90.0]^0$
    - and phi within $[10.0, 80.0]^0$
  - at least 3 and at most 10 gamma
    - With E>2 GeV/c

Example (Assuming We Already Had Such a Filter)

- In sim macro:

```
FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
primGen->SetEventMeanTime(10);
fRun->SetGenerator(primGen);

// here you put the generator that you want to use, let's say DpmDirect
PndDpmDirect *Dpm= new PndDpmDirect(mom,1);
primGen->AddGenerator(Dpm);

// The standard behaviour is the same as before (i.e. no event filtering)

// now add some filters on multiplicities of charged/neutral particles or pdg codes within certain momentum regions
primGen->AddFilterMin( 2, 11, -211 ); // request at least 2 e- OR pi-
primGen->ForPt(1.0, 3.0); // with a p_T within [1.0, 3.0] GeV/c
primGen->ForP(1.5, 3.0); // and with a p within [1.5, 3.0] GeV/c

primGen->AddFilterMax( 4, -11, 211 ); // request at most 4 e+ OR pi+
primGen->ForPz(0.5, 4.0); // with a p_z within [0.5, 4.0] GeV/c

primGen->AddFilterPlusMinMax(4,6); // request at least 4 and at most 6 pos. Charged particles
primGen->ForTheta(20.0, 90.0); // with theta within [20.0, 90.0]^0
primGen->ForPhi(10.0,80.0); // and phi within [10.0, 80.0]^0

primGen->AddFilterMinMax( 3, 10, 22 ); // request at least 3 and at most 10 gamma
primGen->ForP(2.0, 9999.0); // and with a p within [2.0, 9999.0] GeV/c
```

Setting Parameters and Getting Info

- You can tell PndDpmDirect how often it should try to find a suitable event before giving up:
```
primGen->SetFilterMaxTries(99999);
```
- and the filter can tell you at the end of the simulation macro how many events the generator simulated (in total) to get the number of filtered events that you wanted...
```
  cout << primGen->GetNumberOfSimulatedEvents() << " events were simulated in dpm\n";
```
- … as well as how many events reached the limit of tries without success:
```
  // should be 0 if filter is applicable and SetFilterMaxTries is not too low
  cout << primGen->GetNumberOfFilterFailedEvents() << " unsuccessful attempts to find an event that suits your filters\n\n";
```

Desirable Features of a Filter in FairPrimaryGenerator

- Filter events produced by **any event generator**
  - Implementation in FairPrimaryGenerator
- Define **arbitrary groups of particles** (based on PDG code) that satisfy momentum or geometrical requirements
- Filter based on **charge** (+ momentum or geometrical requirements)
- Accept only events from generator that feature the desired multiplicities (min. and max.)
- Count how many events were produced by the generator to reach the number of acceptable events (give user back such information)
- If no acceptable event can be found (in a user-defined amount of tries), accept a random event and warn the user
  - No infinite loop

First Implementation in PndDpmDirect

A first test version of the filter is uploaded at:
https://subversion.gsi.de/trac/fairroot/browser/pandaroot/development/mgaluska/PndDpmDirectWithFilter

It does not include all features
and has the limitation that you can define a constraint for each PDG code ONLY ONCE!

The final implementation will not have this limitation!

Current Implementation in PndDpmDirect – Adding Filter

- Example usage (in sim macro):

```
FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
primGen->SetEventMeanTime(10);
fRun->SetGenerator(primGen);

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

// The standard behaviour is the same as before (i.e. no event filtering)

// now add some filters on multiplicities of pdg codes
Dpm->AddFilterMinMax( 1, 5, 11, -211 ); // request at least 1 and at most 5 e- OR pi-
Dpm->AddFilterMinMax( 1, 5, -11, 211 ); // request at least 1 and at most 5 e+ OR pi+
Dpm->AddFilterMinMax( 3, 9999, 22 ); // request at least 3 and at most 9999 gamma
```

Current Implementation in PndDpmDirect – Restriction of First Implementation

- One restriction is that you can add a filter for a specific pdg code ONLY ONCE:
  Dpm->AddFilterMinMax(1, 3, 11, -211); // request at least 1 and at most 3 e- OR pi-
  Dpm->AddFilterMinMax(2, 5, 11, 211); // request at least 1 and at most 5 e- OR pi+
  The second filter would be ignored in the current implementation.

Current Implementation in PndDpmDirect – MaxTries

- You can tell PndDpmDirect how often it should try to find a suitable event before giving up:
  Dpm->SetFilterMaxTries(99999);
- and it can tell you at the end of the simulation macro how many events dpm simulated in total to get the number of filtered events that you wanted as well as how many events reached the limit of tries without success:

```
  cout << Dpm->GetNumberOfSimulatedEvents() << " events were simulated in dpm\n";
  cout << Dpm->GetNumberOfFilterFailedEvents() << " unsuccessful attempts to find an event that suits your filters\n\n";
```

What else do we need to filter events on?

- Filter events based on
  - Particle multiplicities [>= and =<]
- Criteria
  - PDG code(s) / Charge (+ / – / neutral / charged)
  - Momentum (total / transversal / z)
  - Geometry (theta, phi)

- Any other criteria?

Thank You!