A digitization simulation package for the ILC and CALICE test beam

Guilherme Lima
for the CALICE group at NIU

nicadd

NORTHERN ILLINOIS UNIVERSITY

CALICE Meeting at DESY
December 08, 2004

Talk Outline

- DigiSim: Purpose and requirements
- Package description
- Usage and configuration
- Developing new functionality
- Preliminary results
- Current status and summary

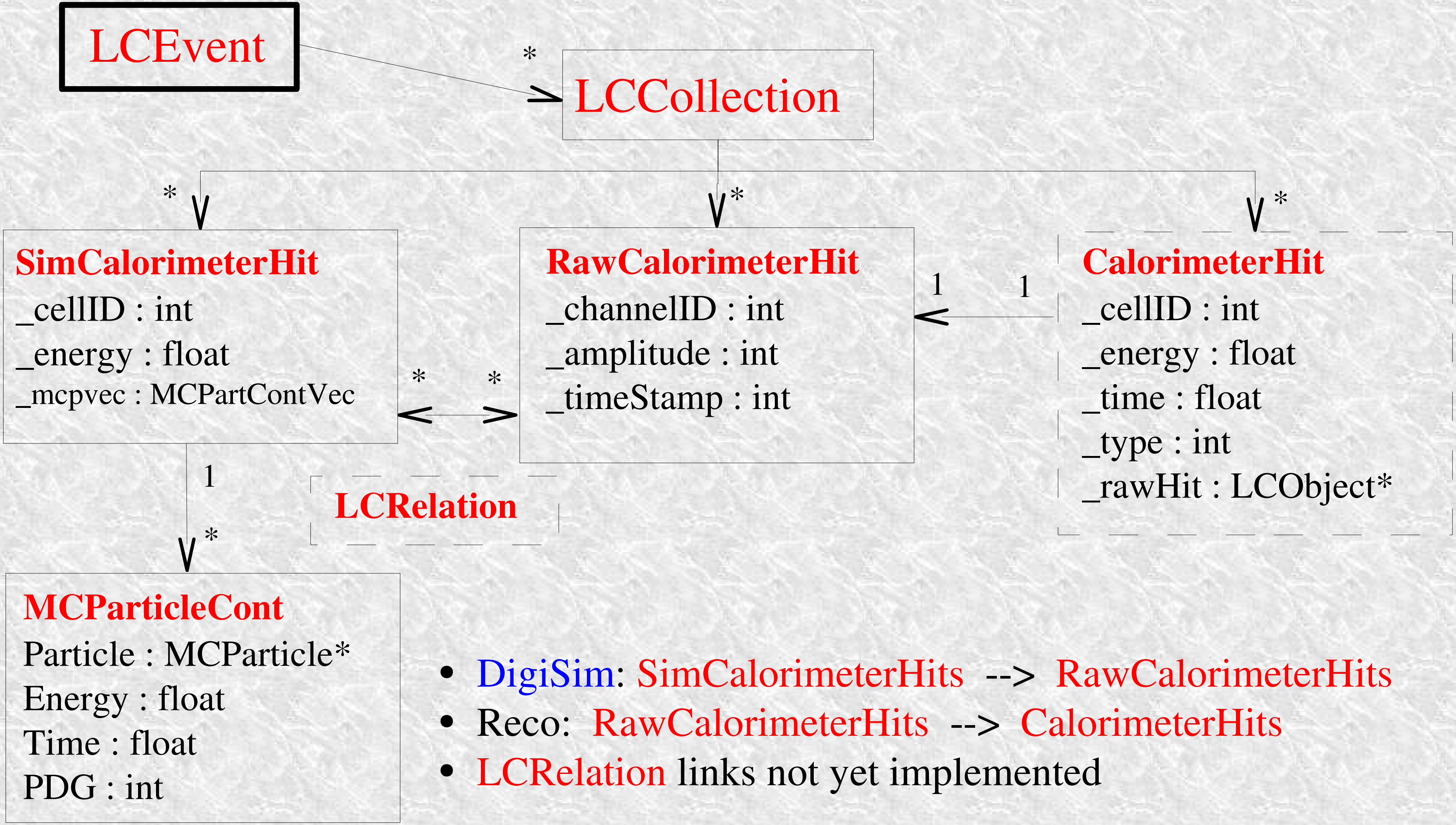
DigiSim

- Goal: a program to simulate the digitization process for the ILC detector simulation
- DigiSim role is to convert the simulated data (energy depositions and hit timings) into the same format AND *as close as possible* to real data from readout channels. Same reco / analysis software can be used for both
- *As close as possible* means that all known effects from digitization process should be taken into account, if possible (cell ganging, inefficiencies, noise, crosstalks, hot and dead channels, non-uniformities, etc.)

Requirements and choices

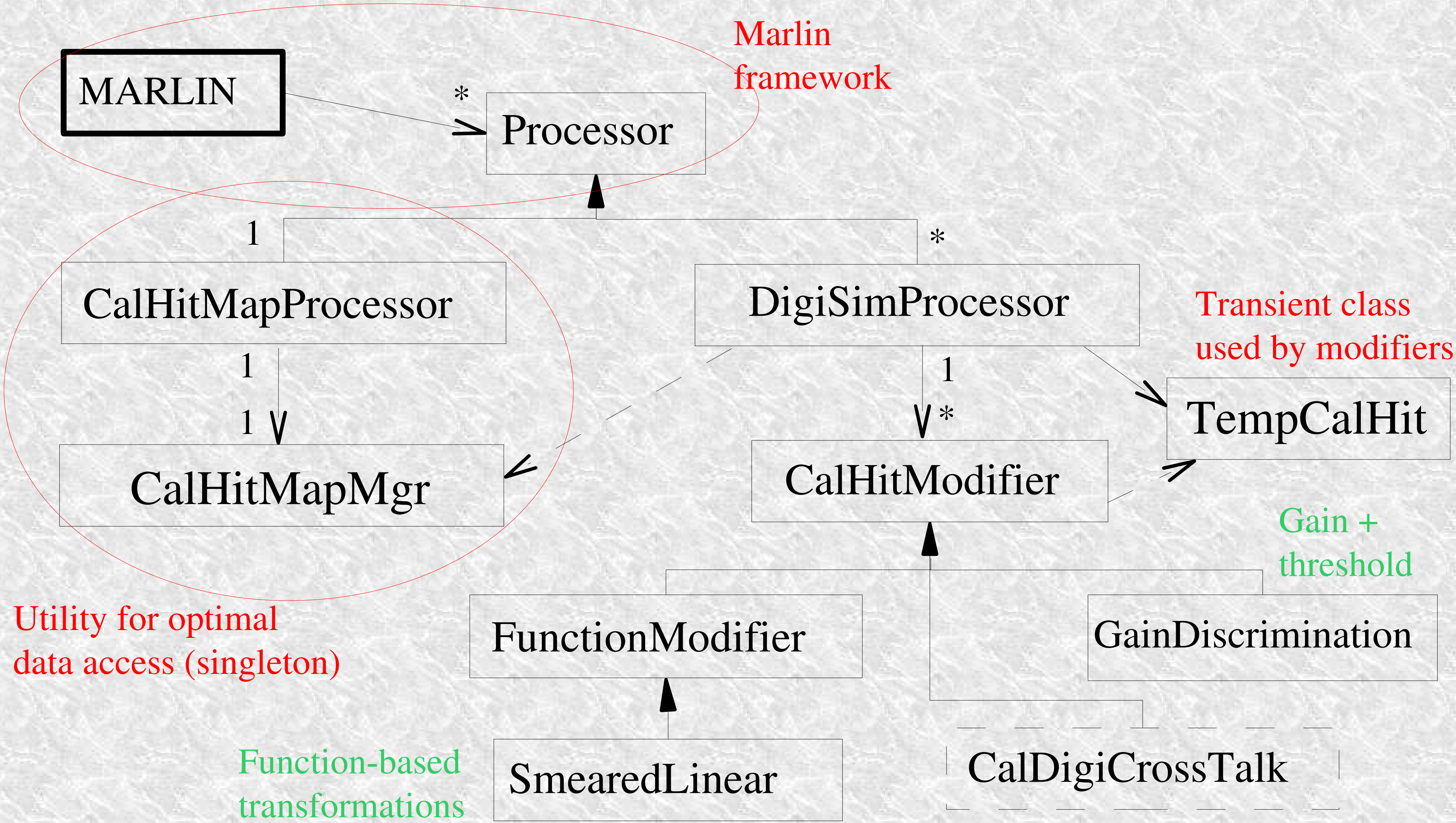
- Basic requirements:
  - Object oriented design to simplify maintenance and implementation of new functionality
  - Should be used within the CALICE test beam project, as a testbed for the reconstruction framework
  - All test beam code based on C++ and LCIO
- Gaede's Marlin chosen as the C++ framework
  (still using version available by ~Nov.20, 2004)

LCIO event model

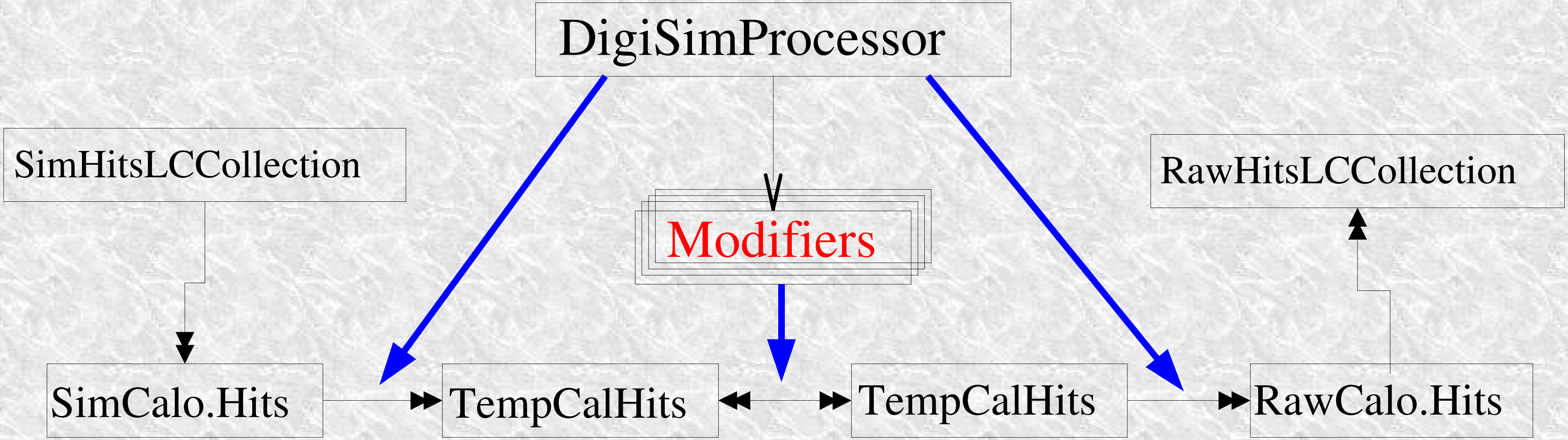


- DigiSim: SimCalorimeterHits --> RawCalorimeterHits
- Reco: RawCalorimeterHits --> CalorimeterHits
- LCRelation links not yet implemented

DigiSim class diagrams



DigiSim event loop



- Calorimeter hits are shown here, but the same structure could be used for trackers as well
- TempCalHits are both input and output to each modifier
- All processing is controlled by a DigiSimProcessor (one per subdetector)
- Modifiers are configured at run time, via the Marlin steering file
- New modifiers can be easily created for new functionality (more info later)

Steering file example

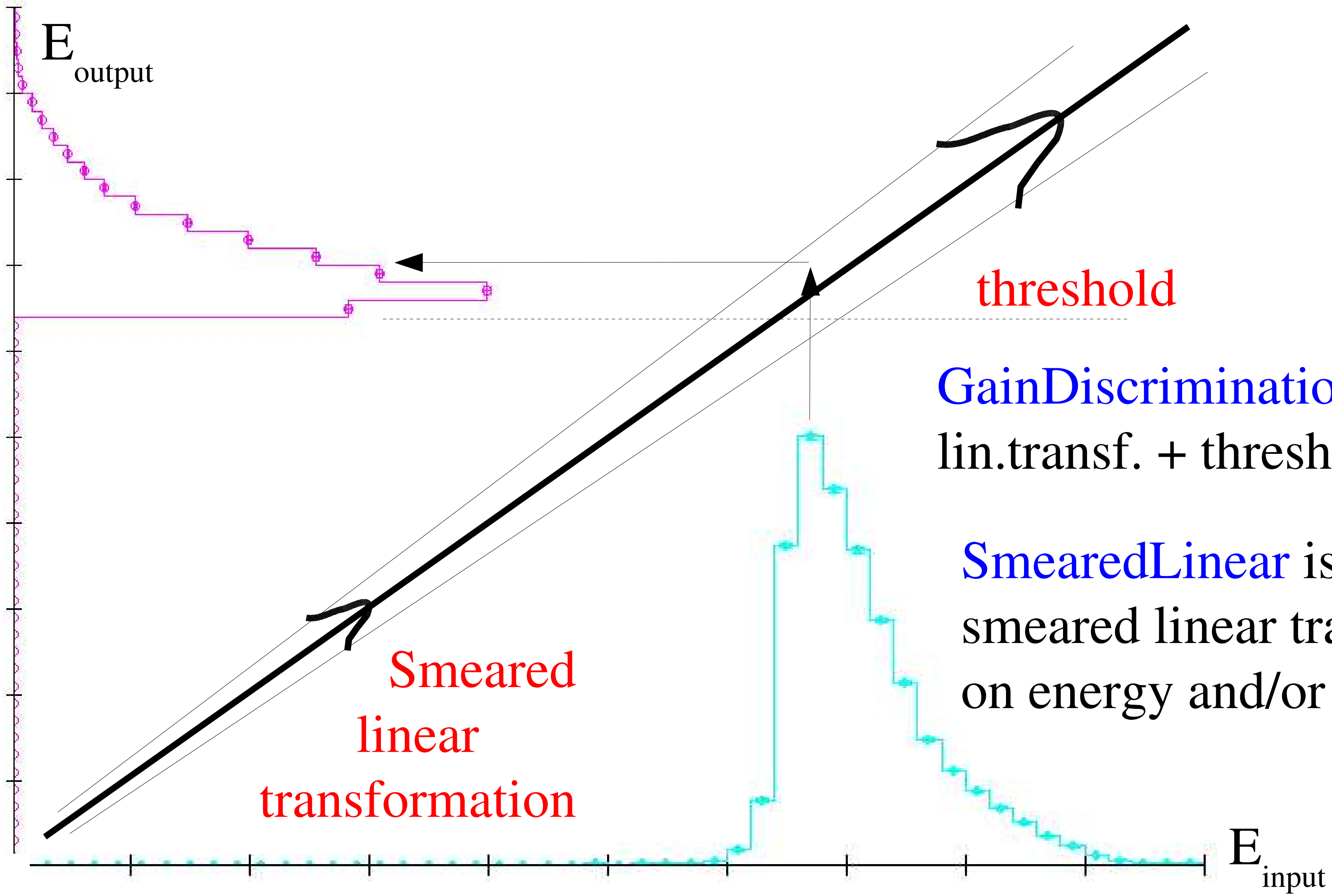
```
#############################################
#  Example steering file for DigiSim
#############################################
.begin Global  -------------------------------------------
# specify one or more input files (in one or more lines)
LCIOInputFiles inputfile

# the active processors that are called in the given order
ActiveProcessors CalHitMapProcessor
ActiveProcessors EMDigiSimProcessor
ActiveProcessors HADDigiSimProcessor
ActiveProcessors TCDigiSimProcessor
ActiveProcessors OutputProcessor

# limit the number of processed records (run+evt):
MaxRecordNumber 500
.end Global  ---------------------------------------------
```

Configuring processors and modifiers

```
#####################################################################
# A DigiSim processor.  Instantiates one or more calorimeter hit
#  "modifiers", which together represent the full digitization process
.begin EMDigiSimProcessor -----------------------------------------
ProcessorType    DigiSimProcessor
InputCollection  EMcalCollection
OutputCollection EMrawCollection
CalorimeterType  EM

ModifierNames    EMFixedGain EMThreshOnly EMDigiIdent

# Parameters:  type                gainNom gainSig  thresh   thrSig
EMFixedGain    GainDiscrimination  1000000       0       0        0
EMThreshOnly   GainDiscrimination        1       0      30        0
EMGainThresh   GainDiscrimination  1000000   50000      30      1.5

# A function-based modifier        ElinNom ElinSig TlinNom  TlinSig
EMDigiIdent    SmearedLinear             1       0       1        0
.end ----------------------------------------------------------------
```

(As many as needed)

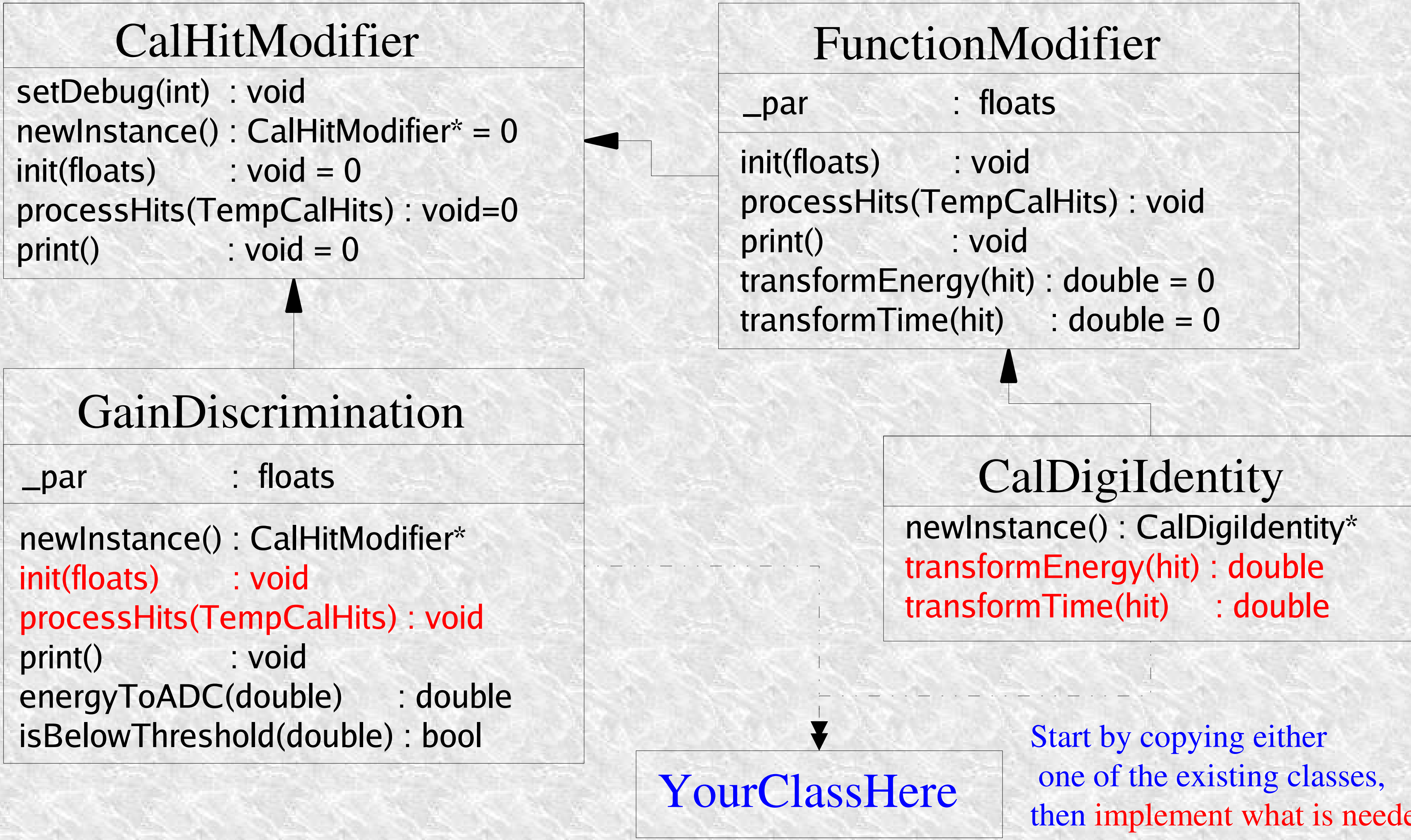
GainDiscrimination and SmearedLinear



GainDiscrimination is a smeared lin.transf. + threshold on energy

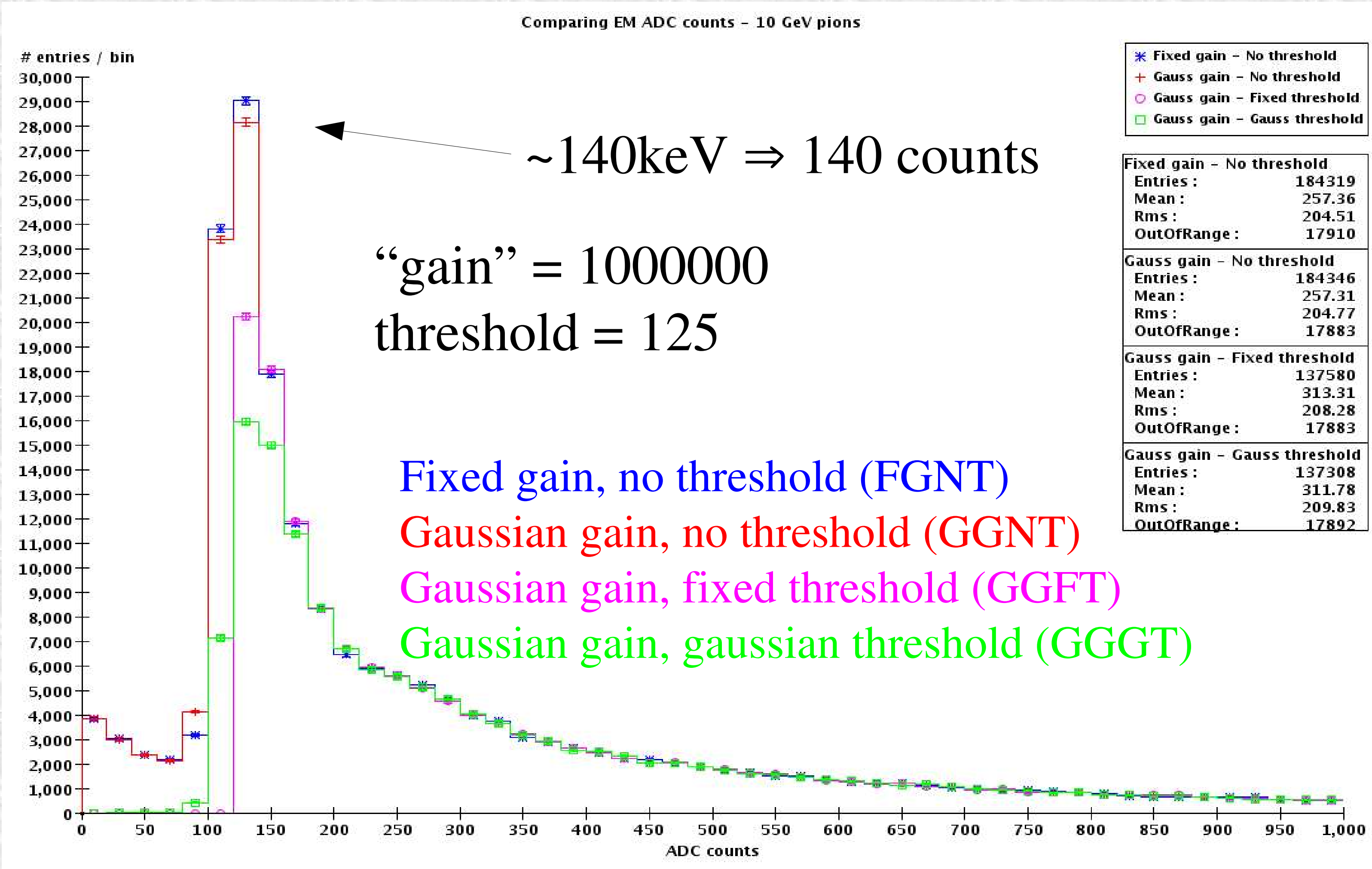
SmearedLinear is a func-based smeared linear transformation on energy and/or timing

Creating new modifiers

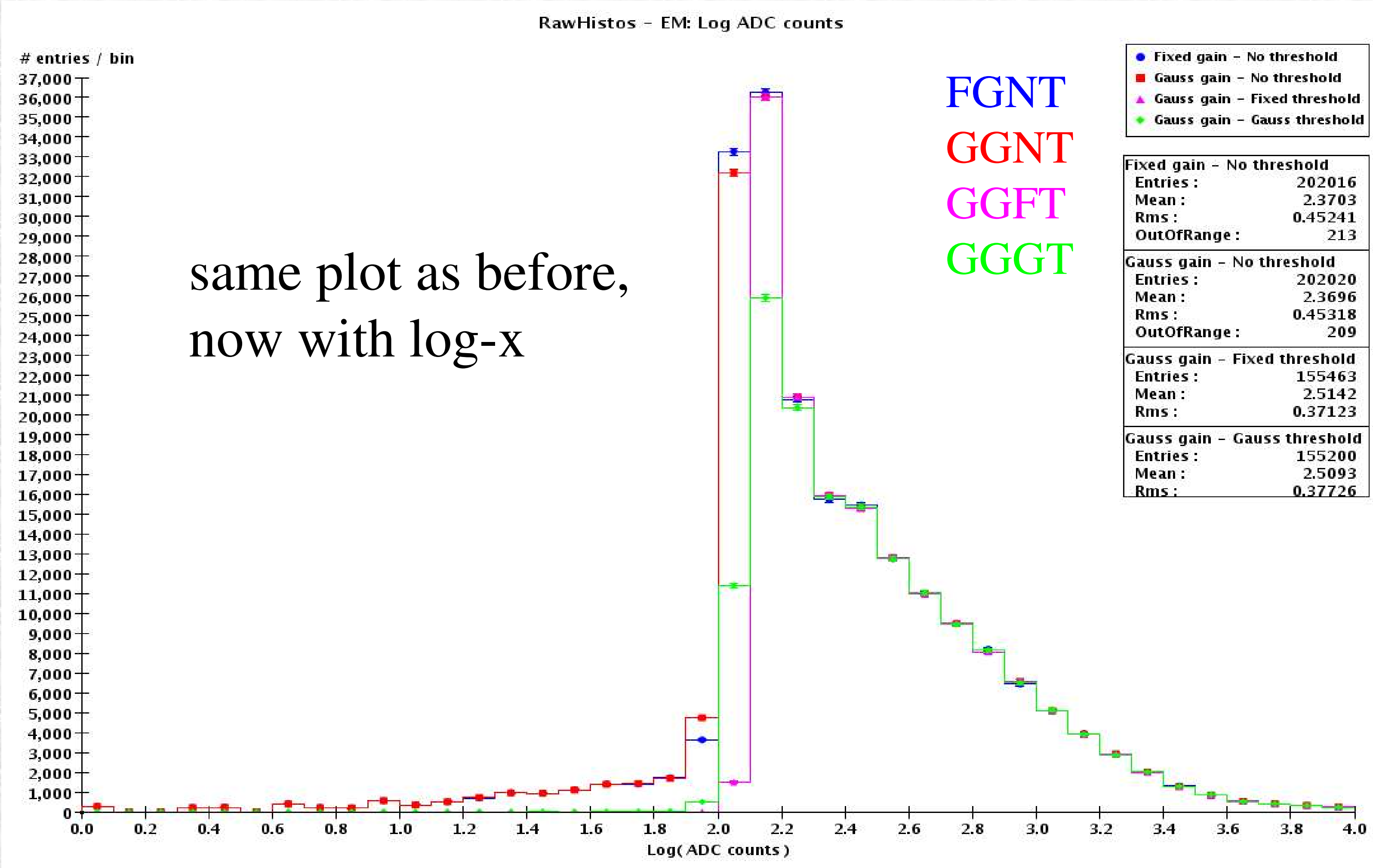


Start by copying either
one of the existing classes,
then implement what is needed

10 GeV $^{\pm}$ on EM: gain + threshold

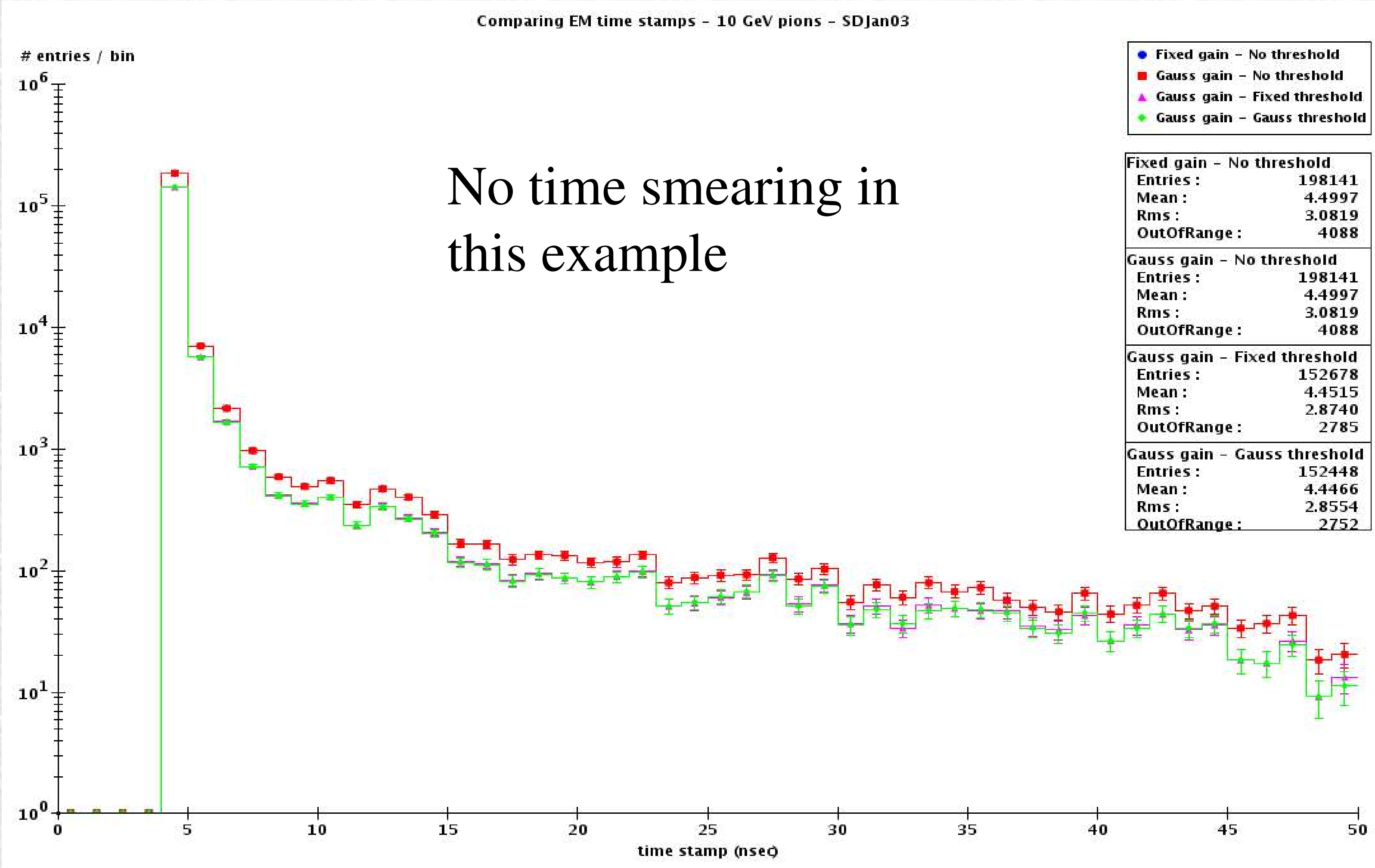


10 GeV $^{\pm}$ on EM: gain + threshold



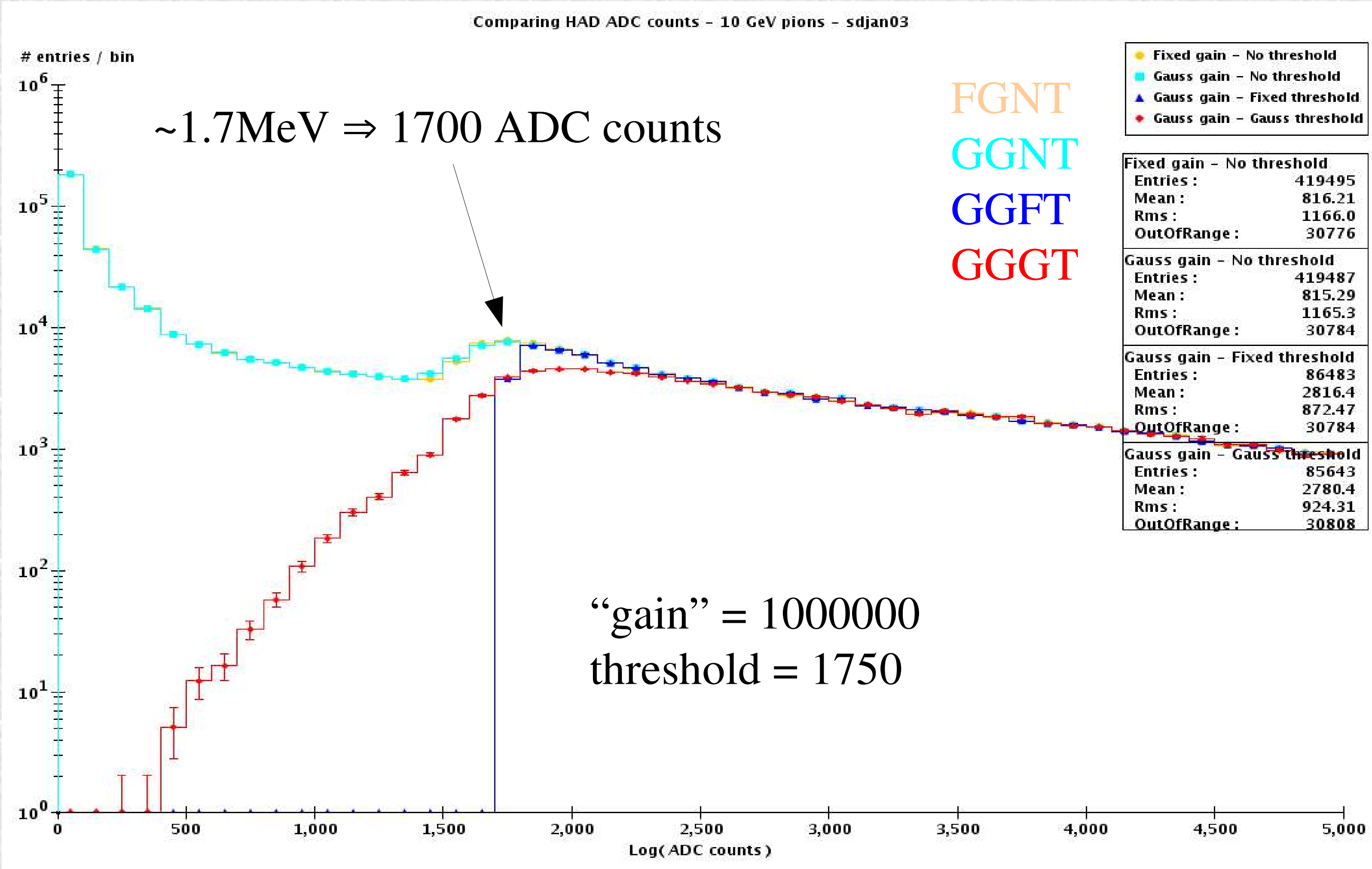
same plot as before,
now with log-x

10 GeV $^{\pm}$ on EM: time stamps

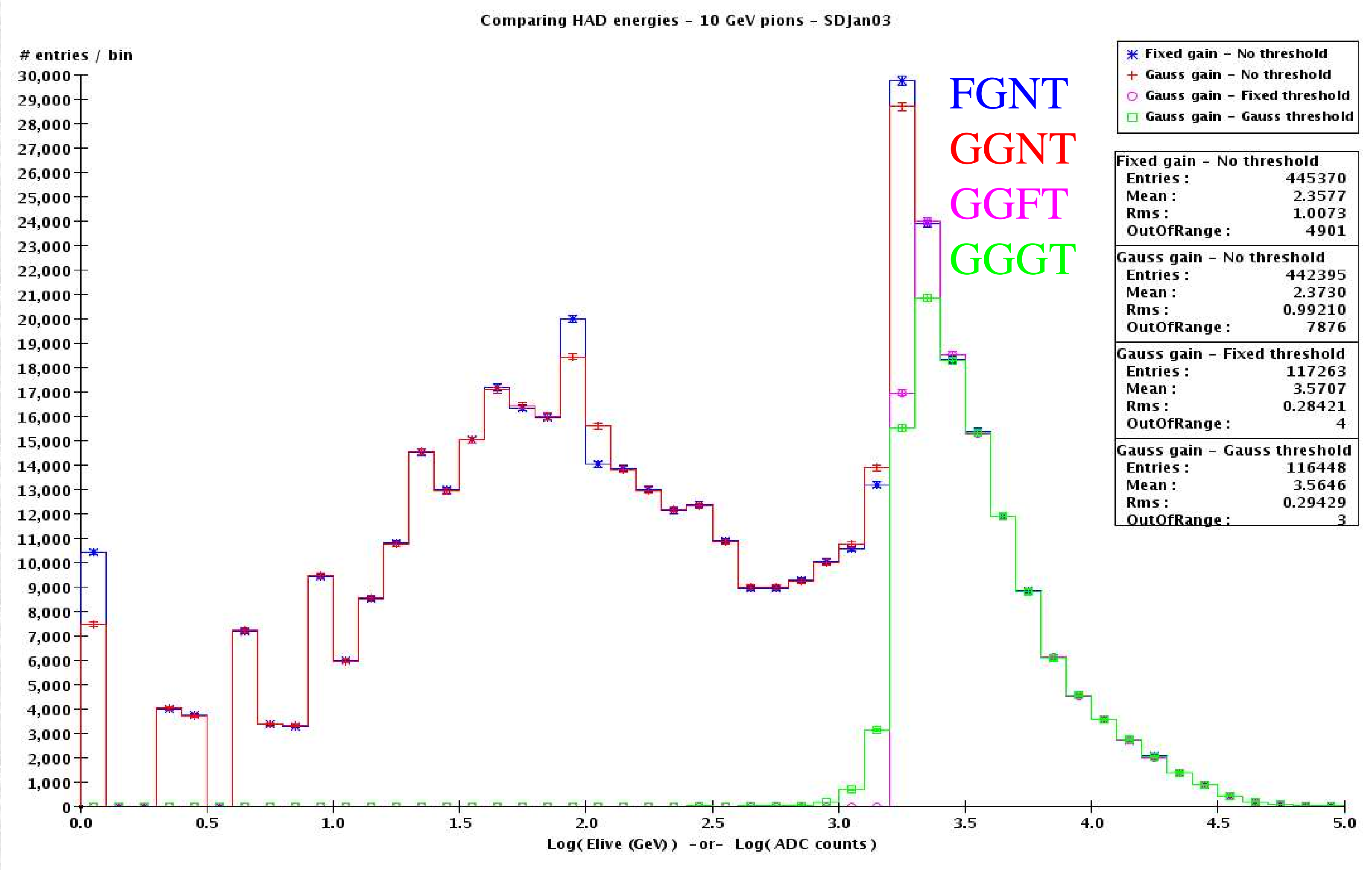


No time smearing in this example

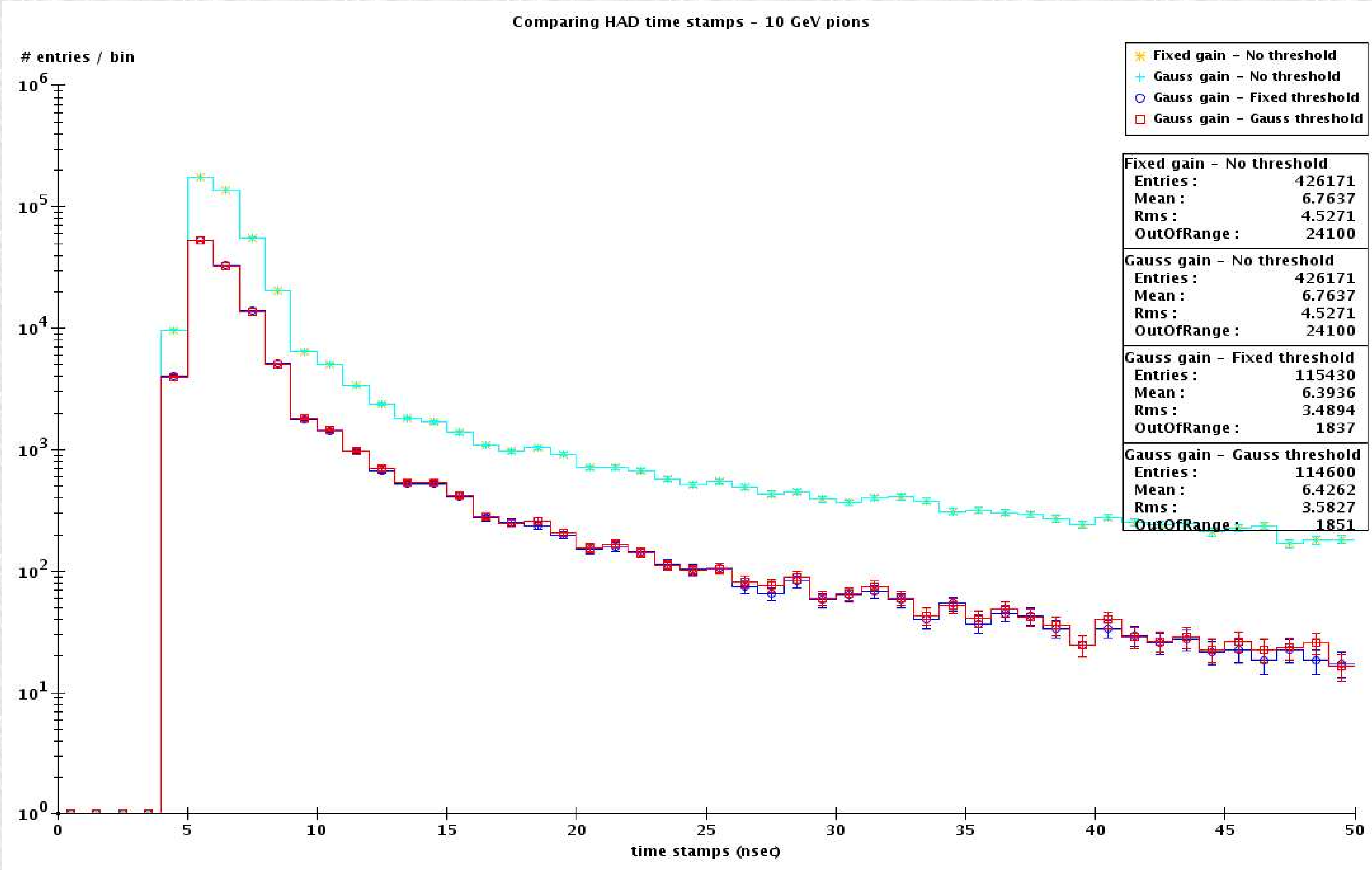
10 GeV $^{\pm}$ on HAD: gain + threshold



10 GeV $\pi^{\pm}$ on HAD: gain + threshold



10 GeV ± on HAD: time stamps



Status

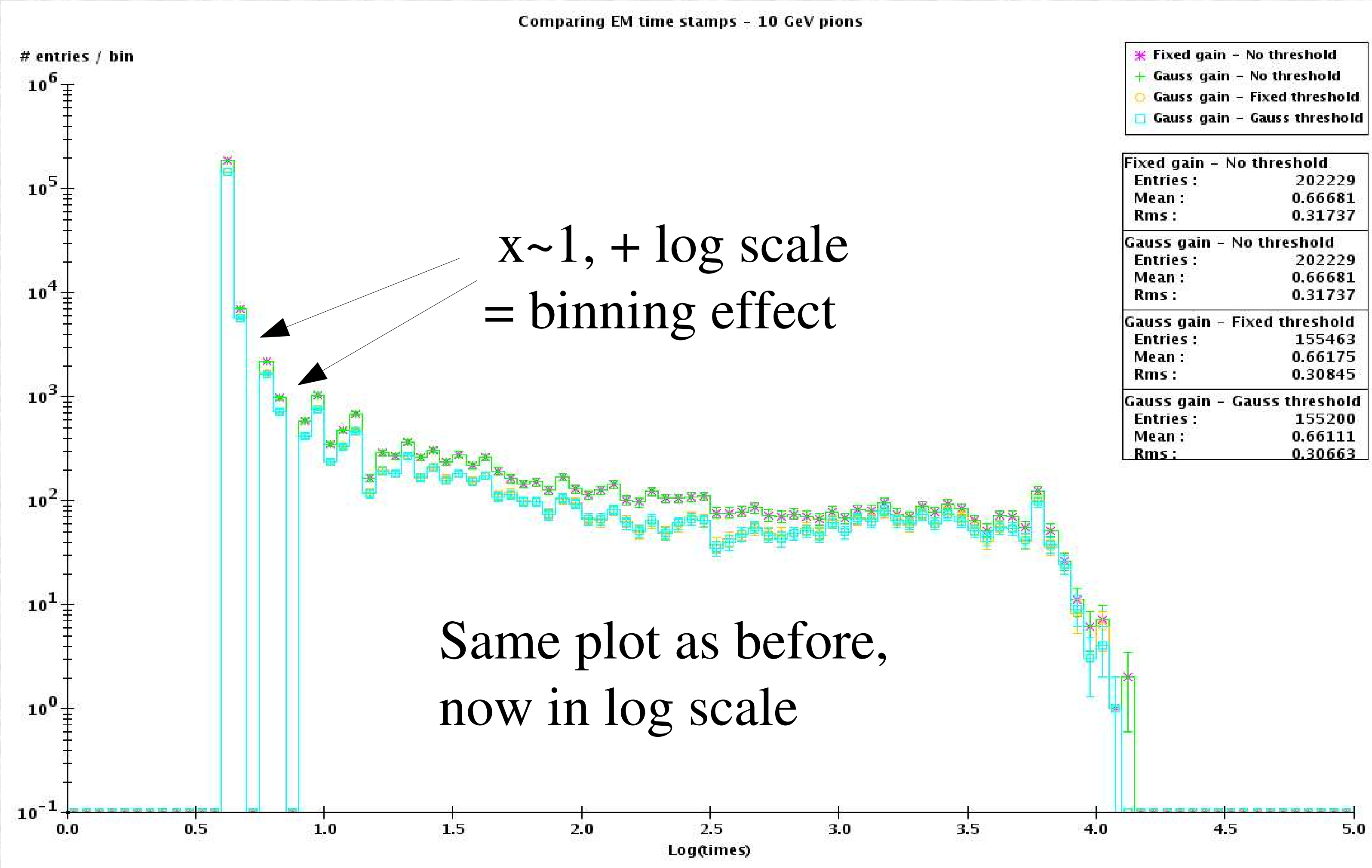
- A first version of DigiSim (proof of concept) is implemented
  - Two real modifiers implemented: GainDiscrimination and function-based SmearedLinear
  - LCRelation: to be implemented (soon?), example code at hand (thanks to F.Gaede)
- Output LCIO files contain EM and HAD RawCalorimeterHit collections, while keeping simulation collections untouched.
- Analysis code for plotting raw hits, plots confirm expected behavior.
- Creation of new (simple) modifiers is quite easy, by just copying one of the existing modifier classes and implementing the desired transformation.
- Some complicated ones: crosstalk and cell ganging require neighborhood information, from geometry-aware classes. NonProj code exists (java and C++), but projective geometry only available in java.

Summary

- A first version of a digitization simulation program, DigiSim, has been developed at NIU, aiming at full digitization of ILC detector simulations
- DigiSim is object-oriented, powerful, extensible, yet very simple implementation using C++
- Digitization of tracking detectors can probably be implemented as easily as the calorimeter hits
- CALICE test beam can use DigiSim as a testbed, for evaluation and improvement suggestions. Please use it for your digitization studies.
- Send any questions or comments to me: lima at nicadd.niu.edu
- Documentation available from http://nicadd.niu.edu/~lima/digisim, including download and building instructions

Danke!

10 GeV ± on EM: time stamps



10 GeV $\pi^{\pm}$ on HAD: time stamps