DigiSim
A digitization simulation package

Guilherme Lima
for the CALICE group at NIU

NORTHERN ILLINOIS
UNIVERSITY

CALICE meeting @ NIU
March 14, 2005

Talk Outline

- DigiSim: Purpose and requirements
- Package description
- Usage and configuration
- Existing modifiers
- Preliminary results
- Current status and summary

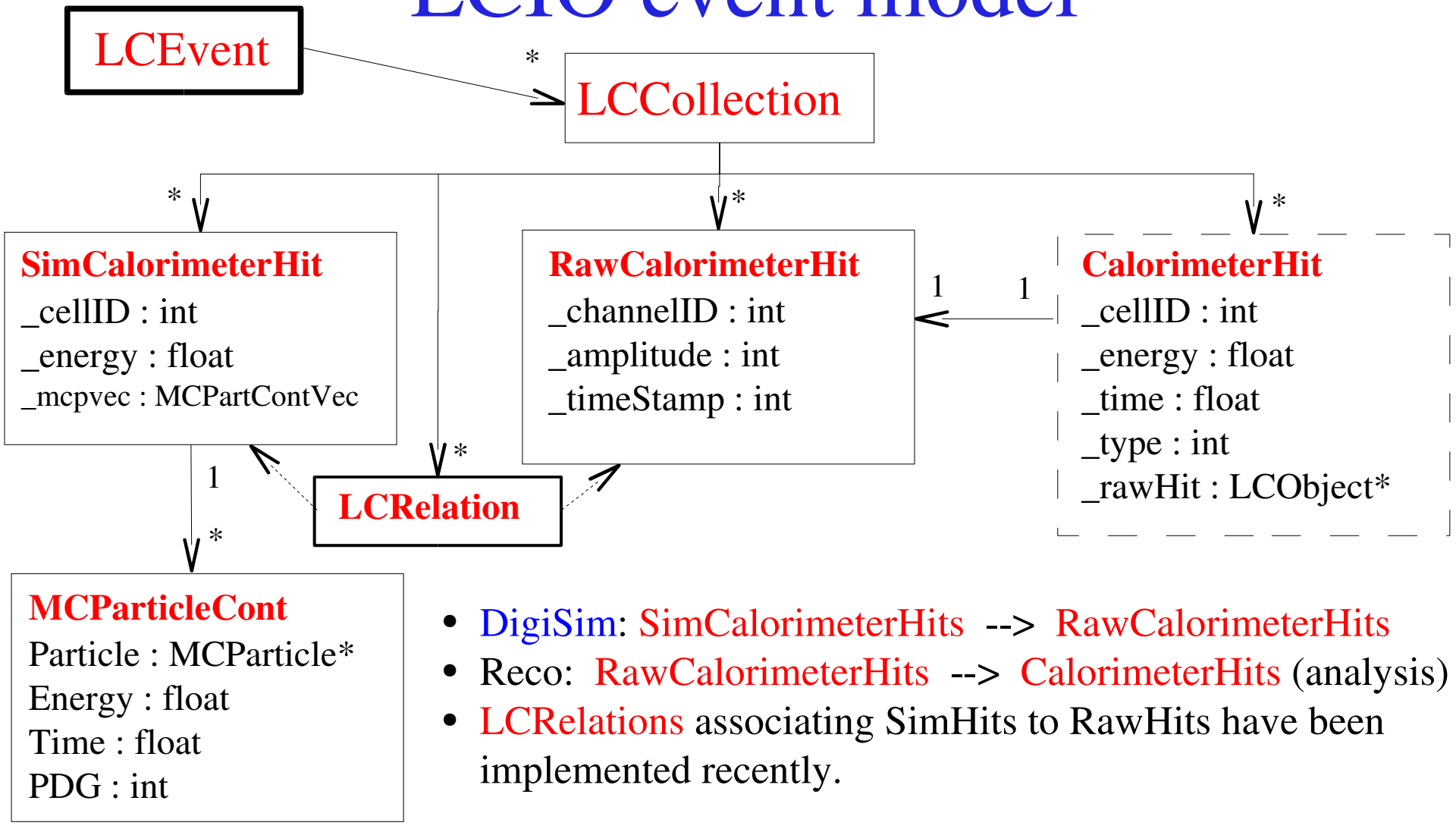
DigiSim

- Goal: a program to simulate the signal propagation and digitization processes 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 MC and real data.
- *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-linearities, attenuation, etc.

Requirements and choices

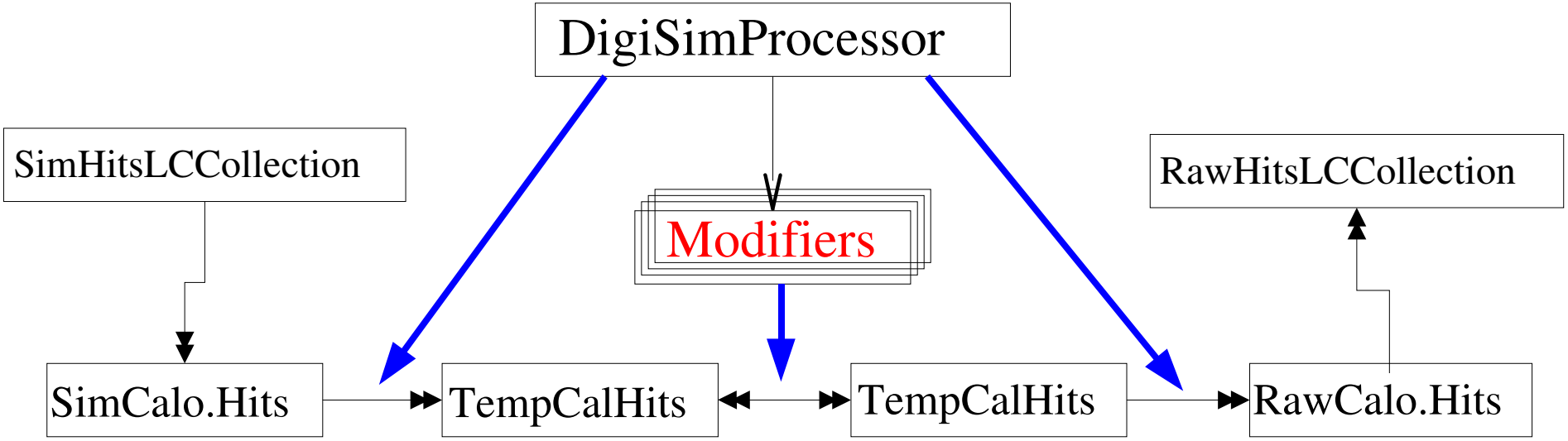
- Basic requirements:
  - Object oriented design to simplify maintenance and implementation of new functionality
  - Intended not only for the full ILC detector, but also for the test beam (TB), as a testbed for full detector digitizaton
  - To be run either standalone, or as an on-the-fly preprocessor to reconstruction and/or analysis program
  - Like all test beam software, to be written on C++ and based on the LCIO event model
- Marlin chosen as the C++ framework

LCIO event model



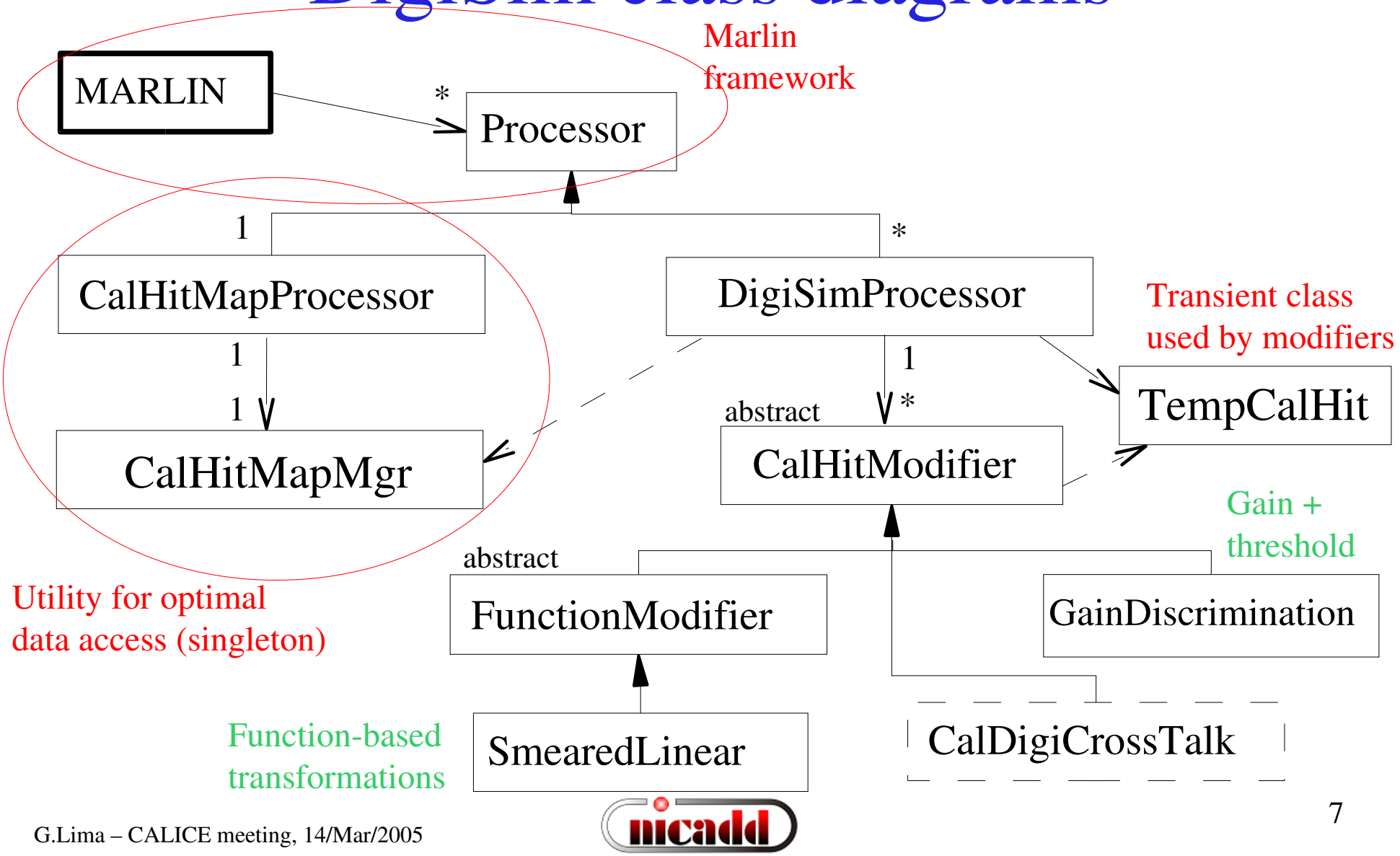
- DigiSim: SimCalorimeterHits --> RawCalorimeterHits
- Reco: RawCalorimeterHits --> CalorimeterHits (analysis)
- LCRelations associating SimHits to RawHits have been implemented recently.

DigiSim event loop

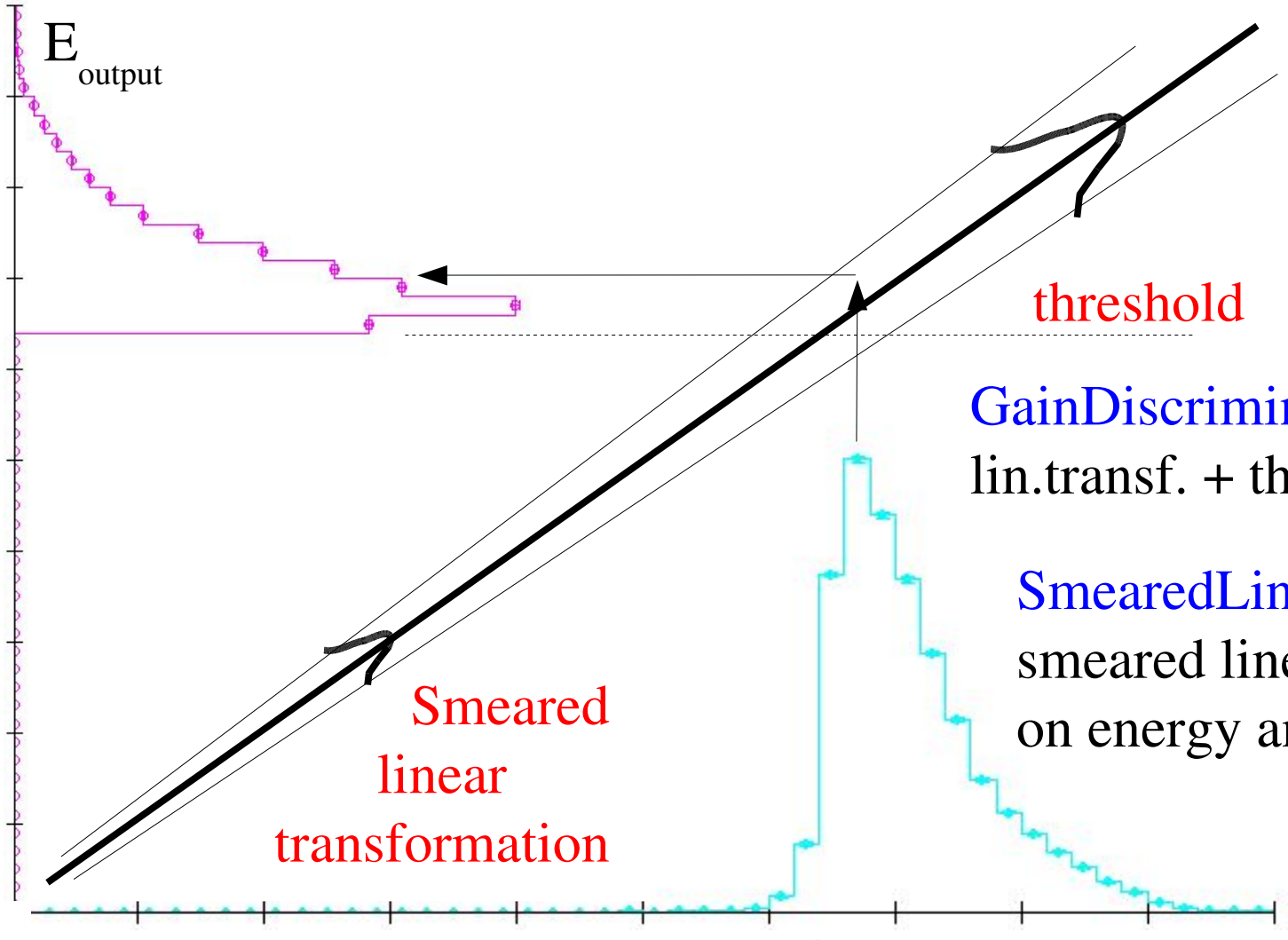


- Calorimeter hits are shown here, but the same structure could also be used for tracker hits
- TempCalHits are both input and output to each modifier (allows for chaining)
- All processing is controlled by a DigiSimProcessor (one per subdetector)
- Modifiers are configured at run time, via the Marlin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

DigiSim class diagrams



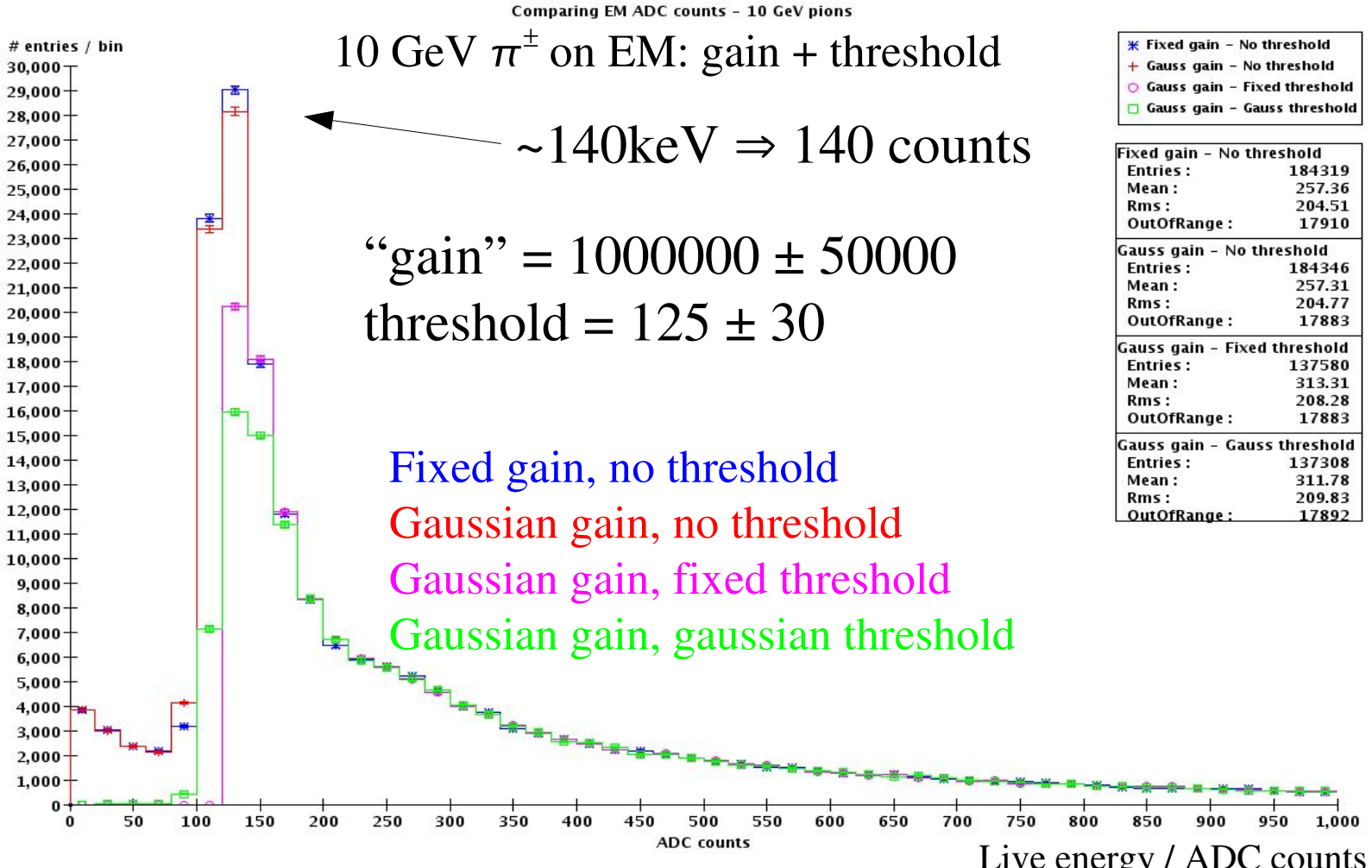
Existing modifiers



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

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

Demo: GainDiscrimination modifier



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

~140keV ⇒ 140 counts

"gain" = 1000000 ± 50000

threshold = 125 ± 30

Fixed gain, no threshold

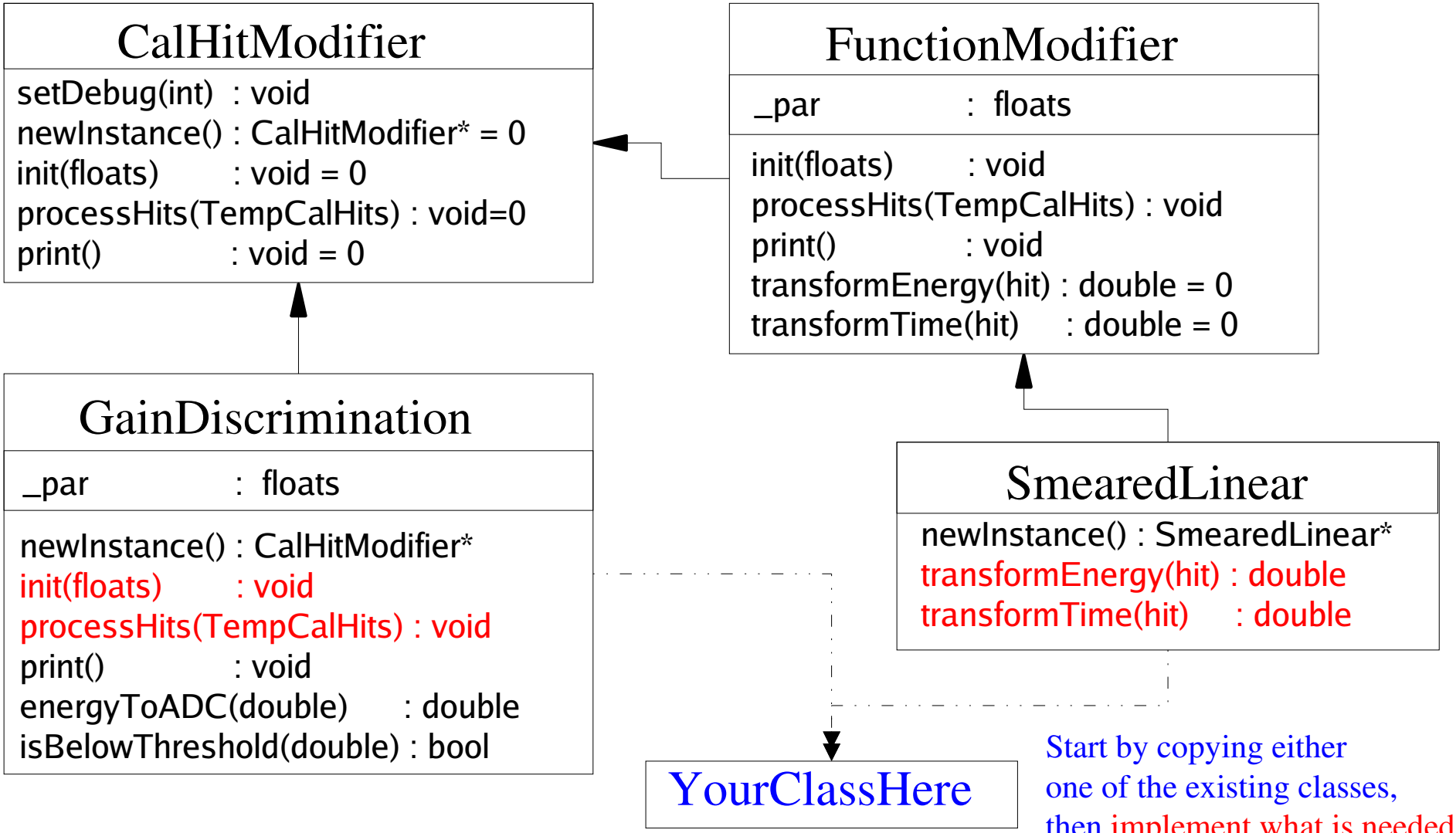
Gaussian gain, no threshold

Gaussian gain, fixed threshold

Gaussian gain, gaussian threshold

Live energy / ADC counts

Use existing modifiers or create new ones?



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

Simple example: numbers from the tile HCal

- Scintillation: 100 eV/photon $\rightarrow$ 10^{+4} photons/MeV

  flat factor of 10^{+7} photons/GeV $\rightarrow$ use GainDiscrimination

  Ex: For a MIP (Geant4): 0.9MeV (normal, 5mm thick) $\rightarrow$ 9000 photons

- SiPM detection efficiency:

  QE ~15%. Again, use GainDiscriminator

- Photon collection efficiency:

  9000 scint.photons/MIP $\rightarrow$ 15PE/MIP detected $\rightarrow$ 15/0.15 = 100 incident photons

  => Eff_{coll} = 100 inc. / 9000 tot.scint. = 0.0111

  Variance (Poisson): $\sigma^2 = \langle N \rangle \rightarrow$ for $\langle N_{PE} \rangle = 15$, $\sigma_N / N \sim 26\%$

  Therefore: $Eff_{coll} = 0.0111 \pm 0.0029$ => use GainDiscrimination with smearing

- Saturation: a function-based class (to be derived from FuncModifier)

Steering file example

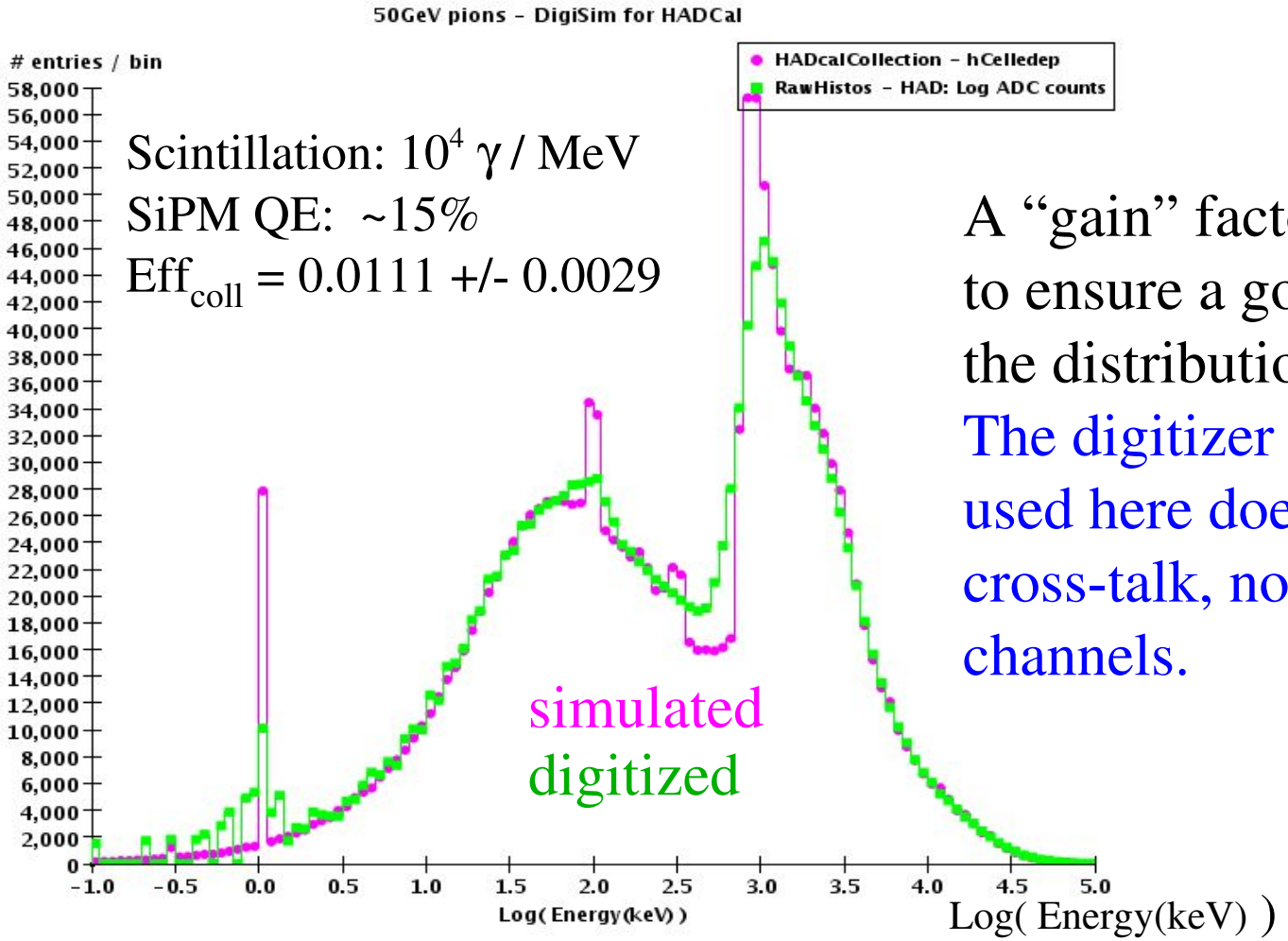
```
##########################################
#  Example steering file for DigiSim
##########################################
.begin Global  ---------------------------------------
# specify one or more input files (in one or more lines)
LCIOInputFiles inputfile

# 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.  It instantiates one or more calorimeter hit
#  "modifiers", which together represent the full digitization process
.begin HADDigiSimProcessor
ProcessorType      DigiSimProcessor
InputCollection    HADcalCollection
OutputCollection   HADrawCollection
Raw2SimLinksCollection  HADraw2simCollection

ModifierNames    HADlightYield HADlightCollEff SiPMQEandAmplif
# Parameters:
#modifierName   type                  gainNom gainSig  thresh   thrSig
HADlightYield   GainDiscrimination 10000000       0       0        0
HADlightCollEff GainDiscrimination   0.0111  0.0029       0        0
# 15% Qeff * 601 amplification
SiPMQEandAmplif GainDiscrimination    90.15       0       0        0

HADGainThresh   GainDiscrimination  1000000   50000     120      2.4

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

(As many as needed)

HCal digitization (1st attempt)



A "gain" factor was used to ensure a good overlap of the distributions shown.

The digitizer configuration used here does not include cross-talk, noise or hot/dead channels.

DigiSim Status

- A digitization simulation package, DigiSim, has been developed at NICADD/NIU
  - Generic modifiers available: GainDiscrimination and function-based SmearedLinear
  - LCRelations implemented to associate raw hits to corresponding simulated hits
- EMraw, HADraw (raw hits) and LCRelations collections are stored into output LCIO files, in addition to the (untouched) simulated calorimeter hits collections.
- A first version of a real (though incomplete) tile HCal digitizer exists.
  It can be used as an example to implement ECal or TCMT.
- When reuse of existing generic modifiers is not appropriate, the creation of new modifiers is quite easy, by just copying one of the existing modifier classes and implementing the desired transformation.
- Some modifiers, like crosstalk and cell ganging, require neighborhood information from geometry-aware classes, which are currently under development.

Summary

- A digitization simulation program, DigiSim, has been developed at NICADD/NIU, aiming at full digitization of both ILC detector and test beam simulations. Please use it and provide feedback (DESY TB data?)
- Digisim can be run in either in a stand-alone mode to produce persistent lcio output, or as an on-the-fly preprocessor to reconstruction/analysis
- In the stand-alone mode, the output should contain all the information present at input, plus quantities calculated by DigiSim.
- Digitization of tracking detectors hits can probably be implemented as easily as for calorimeter hits, as shown in this talk.
- Documentation available at http://nicadd.niu.edu/digisim/DigiSim.html, including download and building instructions
- Send any questions or comments to me: lima at nicadd.niu.edu