Event Processing

Jason Detwiler

MaGe Workshop 2010

18-20 January, Munich

MGTEvent

- Object to give access to all event related information
  - Simplifies user analysis code
  - Don’t make users hunt down event data
- Current data members:
  - double fETotal
  - TClonesArray* fWaveforms
  - TClonesArray* fDigitizerData

MGTEvent

- Possible future additions:
  - Run info
  - Hit pattern
  - PSA parameters
  - Single / multi-site tag
  - MC event / step information

Modular Processing

- Divide analysis into stages:
  Begin()
  BeginRun()
  ProcessEvent()
  [ProcessWaveform()]
  EndRun()
  End()
- Package into modules that can be run in parallel

TAM

- Used by Antares, PHOBOS, some in CMS
  - Several years old: mature, stable
  - Lightweight: only 8 classes
  - See associated slides, and

http://www.cmsaf.mit.edu/twiki/bin/view/Software/TAM

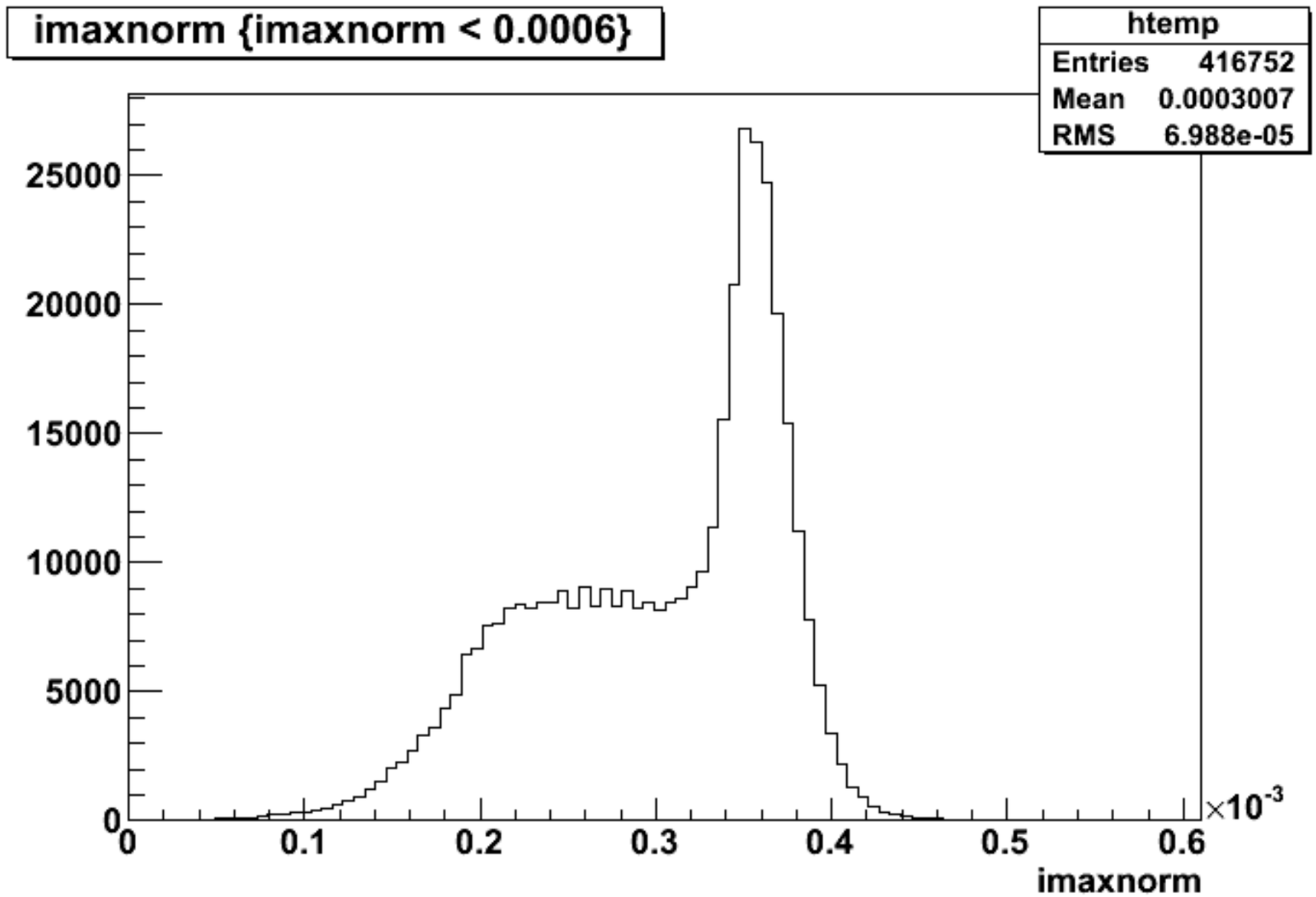
Example: I_{max}/E PSD

```
class GATGerdaPSDProc : public GATEventProcBase
{
  public:
    GATGerdaPSDProc(const char *name="gatgerdapsdproc",
                    const char *title="Processor to calculate the GERDA-style PSD parameter: charge-normalized current max");
    virtual ~GATGerdaPSDProc() {}

  protected:
    virtual void OnSlaveBegin();
    virtual void ProcessWaveform(MGTWaveform& waveform, MJTGretina4DigitizerData& digitizerData, MGTEvent&);
    virtual void SlaveTerminate();

  protected:
    TTree* fTree;
    double fNormCurrentMax;
    MGTWaveform fWaveformOut;
    MGWFRCDifferentiation fWFDifferentiator;
    MGWFExtremumFinder fWFExtremumFinder;

  ClassDef(GATGerdaPSDProc,1); //ROOT standalone GATGerdaPSDProc
};
```

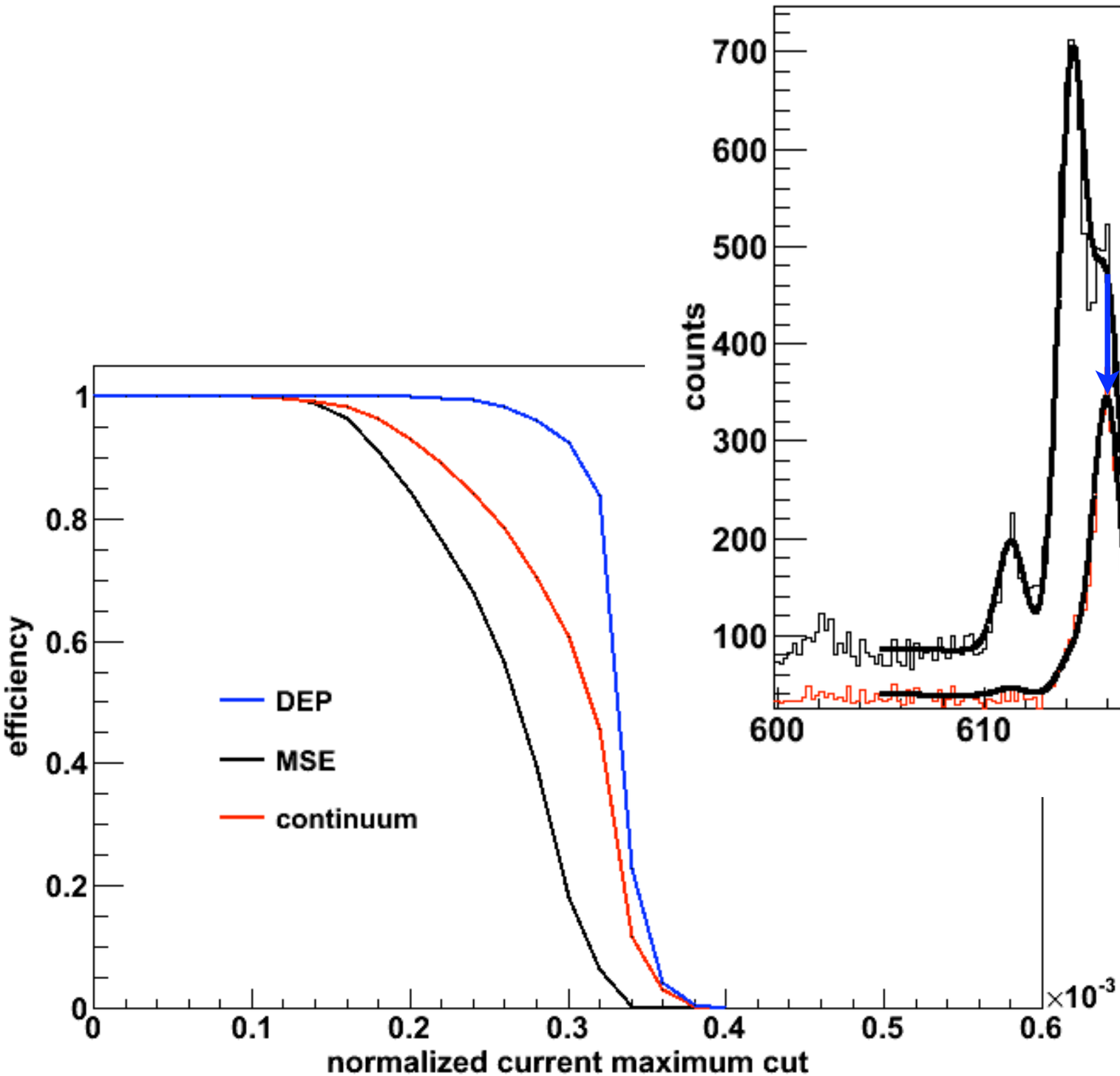
Example: I_{max}/E PSD

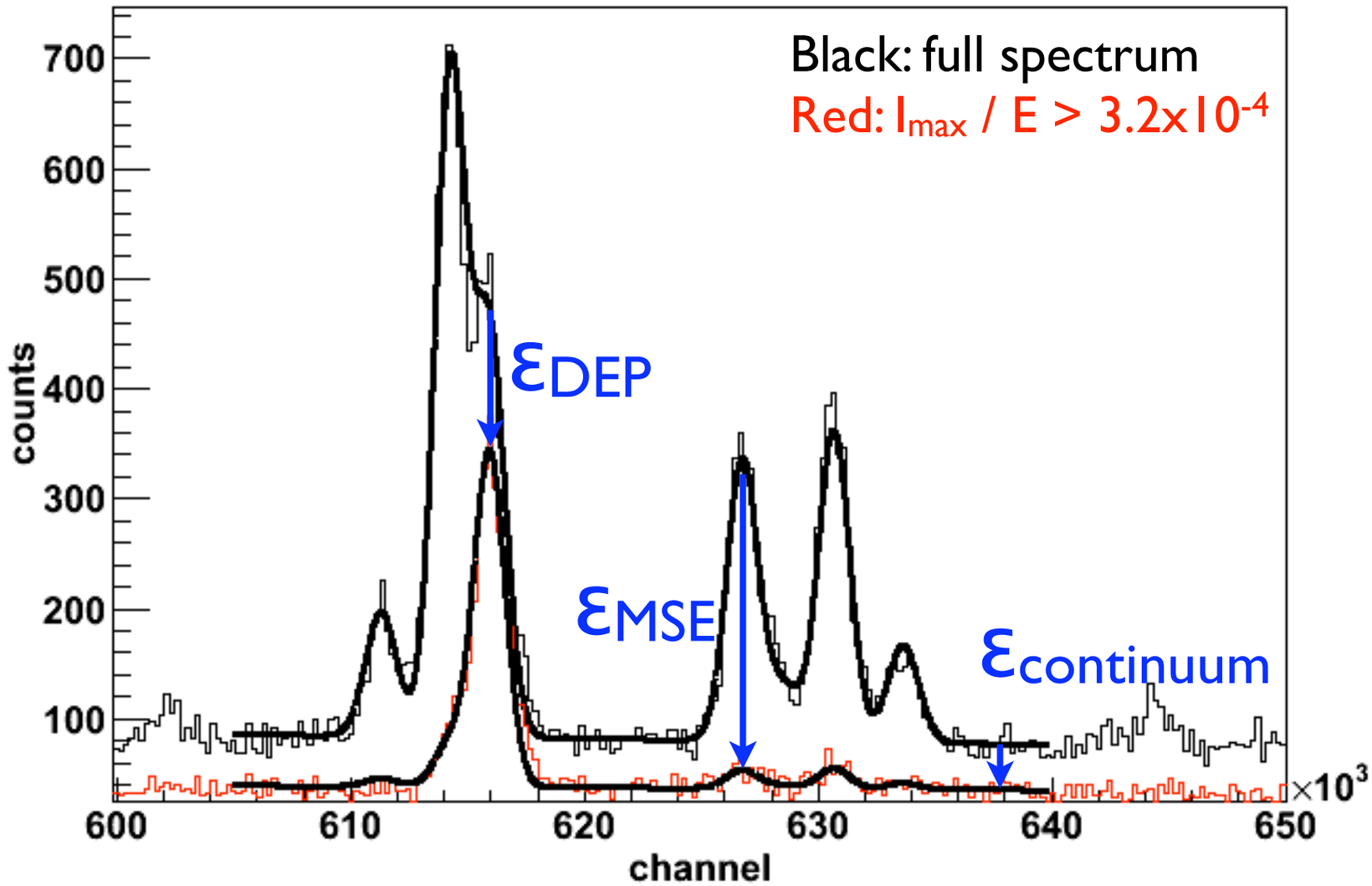
```
GATGerdaPSDProc::GATGerdaPSDProc(const char *name, const char *title) :
GATEventProcBase(name, title), fTree(NULL), fNormCurrentMax(0)
{
  fWFDifferentiator.SetTimeConstant(10.*ns);
  fWFExtremumFinder.SetFindMaximum(true);
}

void GATGerdaPSDProc::OnSlaveBegin()
{
  fTree = new TTree("NormCurrentMaxTree", "NormCurrentMaxTree");
  fTree->Branch("fNormCurrentMax", &fNormCurrentMax, "imaxnorm/D");
}

void GATGerdaPSDProc::ProcessWaveform(MGTWaveform& waveform, MJTGretina4DigitizerData& digitizerData, MGTEvent&)
{
  fWFDifferentiator.Transform(&waveform, &fWaveformOut);
  fWFExtremumFinder.Transform(&fWaveformOut);
  double currentMax = fWFExtremumFinder.GetTheExtremumValue();
  fNormCurrentMax = currentMax / digitizerData.GetEnergy();
  fTree->Fill();
}

void GATGerdaPSDProc::SlaveTerminate()
{
  AddOutput(fTree);
}
```

Example: I_{max}/E PSD



Example: I_{max}/E PSD





Storing Events

- My original idea: store a MGTEvent in each row of a TTree
- Want to store different event data into different files to maximize flexibility and efficiency in data storage, transfer, loading, looping: use TTree friending to associate data in different files
- Can friend two TTrees containing MGTEvents, but accessing both MGTEvents is non-trivial (e.g. can't access friend via TTree::Draw())

Storing Events

- Make TTrees whose branches are the data members of MGTEvents
- Use TAM to load the branch objects into the MGTEvents for users
- Use a database to keep track of what data is where: make a convention for how to divide the event data into different files

Splitting Event Data

- MC Tree
  - MC event info
  - MC step info
- Run/Event Tree
  - Run info
  - Event info
- Waveform Tree
  - Waveforms
  - Digitizer Data
- Analysis Tree
  - Hit Pattern
  - PSA output
  - Event tags