# Java Physics Generator, Detector Simulation and Analysis Modules

Mike Ronan

Lawrence Berkeley National Laboratory

UTA 2003 American Linear Collider Workshop

# Outline

- Java Analysis Framework
  - Generator interface
  - Generator and analysis modules
  - Analysis classes and tools (e.g. JAS)
- Linear Collider Monte Carlo Event Generators
  - Pandora, Pythia, Whizard and KK2f
  - Low energy Monte Carlo packages such as EvtGen, GamGam and KoralB
- Pandora, Pythia & Whizard Higgsstrahlung Simulation
  - Higgsstrahlung event reconstruction
  - Background simulations
- Detector Simulation Models
  - LCD Fast Monte Carlo(FMC)
  - TESLA SimDet
  - JLC QuickSim
- FMC, SimDet & QuickSim Comparisons
  - Whizard Higgsstrahlung simulation
- SLAC Monte Carlo Event Generation
  - Plans to generate several $ab^{-1}$ of simulated data.
  - Status
- Summary

# Java Framework

## Generator Interfaces

- Methods:
  - setup() - Set up generator configuration.
  - initialize() - Initialize for selected process.
  - generateEvent() - Generate an event.
  - terminate() - Summarize generated sample.

- also
  - getName() - Return generator name and version.

- and generator specific methods such as
  - list(int) - List particles in the event.

- Useage:

```
pythia = new Pythia();

for (int n=1; n<=NEvents; n++) {

  pythia.generateEvent();

  pythia.list(1);
}
```

# Analysis Modules

- Event Generator modules
  - Package hep.lcd.generator

    `Generator`
    `GeneratedEvent`
    `HEPEvent`

  Implementations: e.g. Pythia

```
generator = new PythiaGenerator(processName);

event = generator.generateEvent();
```

- Event Analysis modules
  - Package hep.lcd.util.driver

    `Driver`
    `Processor`

  Example:
  - public class AnlPythiaZHJets implements Driver

```
add(detectorSimulation);

add(analysis);
```

  - class PythiaZHJetanalysis implements Processor

```
particles = event.get("MCParticles");

...
```

# Analysis Classes & Tools

- Physics classes, e.g.
  - Particle

```
ParticleType type = particle.getType();
  int PDGID = type.getPDGID();
double[] PV = particle.getMomentum();
  double cosTh = PV[2]/Math.sqrt(PV[0]*PV[0]+...);
```

- Analysis classes, e.g.
  - Histogram

```
histogram("Mass").fill(particle.getMass());
```

- Analysis tools
  - GNUMakefile & convenient Java packages

```
gmake
  compiles and creates a shared object library
java -mx64M GenWhizard ...
  writes a xxx.javahist unbinned histogram file
```

  - Java Analysis Studio

```
jas
  open files: xxx.javahist yyy.javahist ...
  overlay and format plots
  save as PostScript files
```

# Event Generators

- Pandora-Pythia V2.2 Monte Carlo
  using the PandoraPythia interface package

  - NLC500 machine simulation
  - Using a eetoZHiggs process with Mh = 115 GeV

- Pythia V6.2 Monte Carlo
  w/ Circe beamstrahlung simulation

  - TESLA beamstrahlung simulation
  - Using a eetoZH process with Mh = 115 GeV

- Whizard V1.22 Monte Carlo
  w/ ISR & Circe turned on

  - TESLA beamstrahlung simulation
  - Reading the configuration file whizard.in
    * process_id = "zh_o"
    * MH = 115
    * ISR_on & CIRCE_on = t

# Also

- KoralB V2.5 Monte Carlo
  w/ Tauola Tau decay simulation
  - Using a simple Fortran input data file
  - and e.g. a eetoa1a12nu process with
    tauola.setDecayMode(1,5); ...
  - or a eetoTau13 process generating only Tau13 events

- KK2f V4.16 Monte Carlo
  w/ Tauola Tau decay simulation
  - Using a .KK2f_defaults file and process files

# Event Generation

## Pandora (C++) event generator

- Signal $e^+e^- \rightarrow Z\ Higgs$ process
  - allows simple implementation of Matrix elements including full polarization
  - defines beam, process and utility classes, e.g.

```
b1 = new Beam(Ebeam,Pol_b1,ELECTRON,ELECTRON);
b2 = new Beam(Ebeam,Pol_b2,POSITRON,POSITRON);

Pandora pandora = new Pandora(b1,b2,process);
```

  - use Pandora Java-native interface (JNI)

## Pandora's Box (C++) event source

- Adds background processes
  - $e^+e^- \rightarrow ZZ$
  - $e^+e^- \rightarrow WW$
  - $e^+e^- \rightarrow t\bar{t}$
  - $e^+e^- \rightarrow q\bar{q}$
  - $\gamma\gamma \rightarrow WW$
- Missing background processes
  - $e^+e^- \rightarrow Z\gamma$ with $\gamma \rightarrow q\bar{q}$

# Pythia (F77) event generator

- User defined processes
  - allows access to many different generators and processes

- use Pythia Java-native interface (JNI)
  - allows user defined processes as above

```
if (processName.equals("User-defined")) {

  pythia = new Pythia();

  pythia.give(argument-list);

  pythia.init("CMS","e+","e-",2.*Ebeam);
}
```

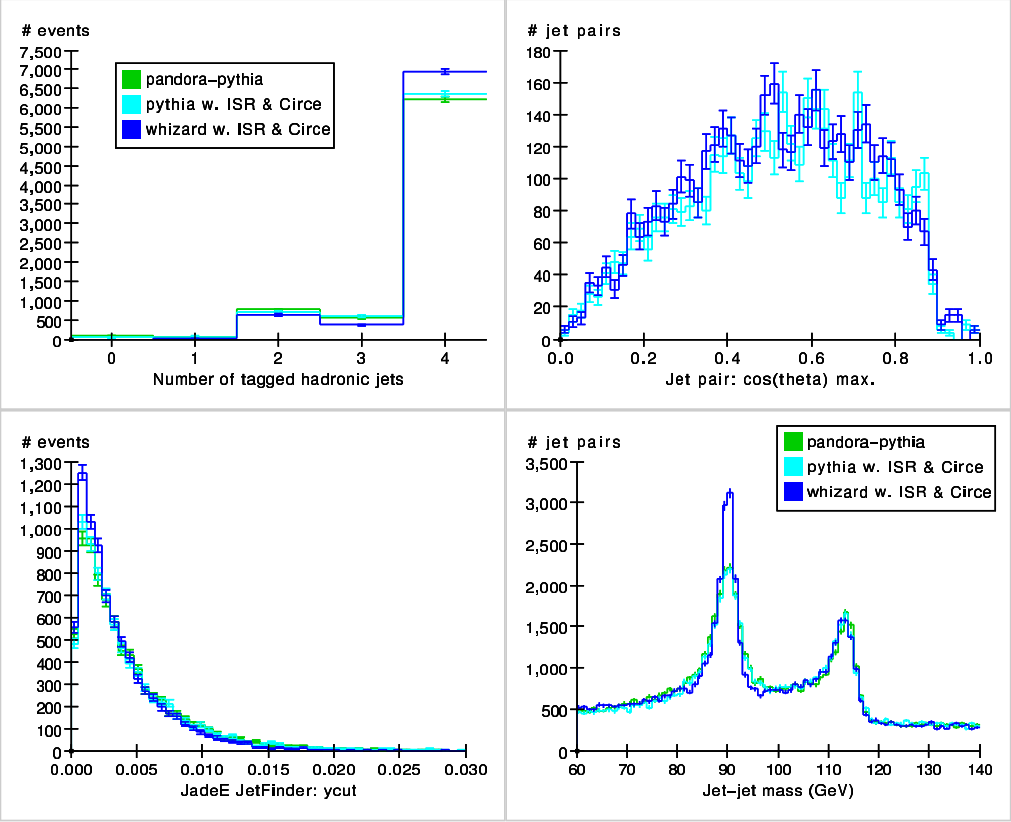
  - or, predefined processes similar to Pandora processes

```
else {
  process = selectProcess(processName);  // Select process

  pythia = new Pythia(process);

  pythia.init("CMS","e+","e-",2.*Ebeam);
}
```

# Pandora, Pythia & Whizard Comparisons




FMC Particle jet distributions:

a.) Number of "correctly" reconstructed jets

Excess at 2 arises from Z decays to neutrinos.

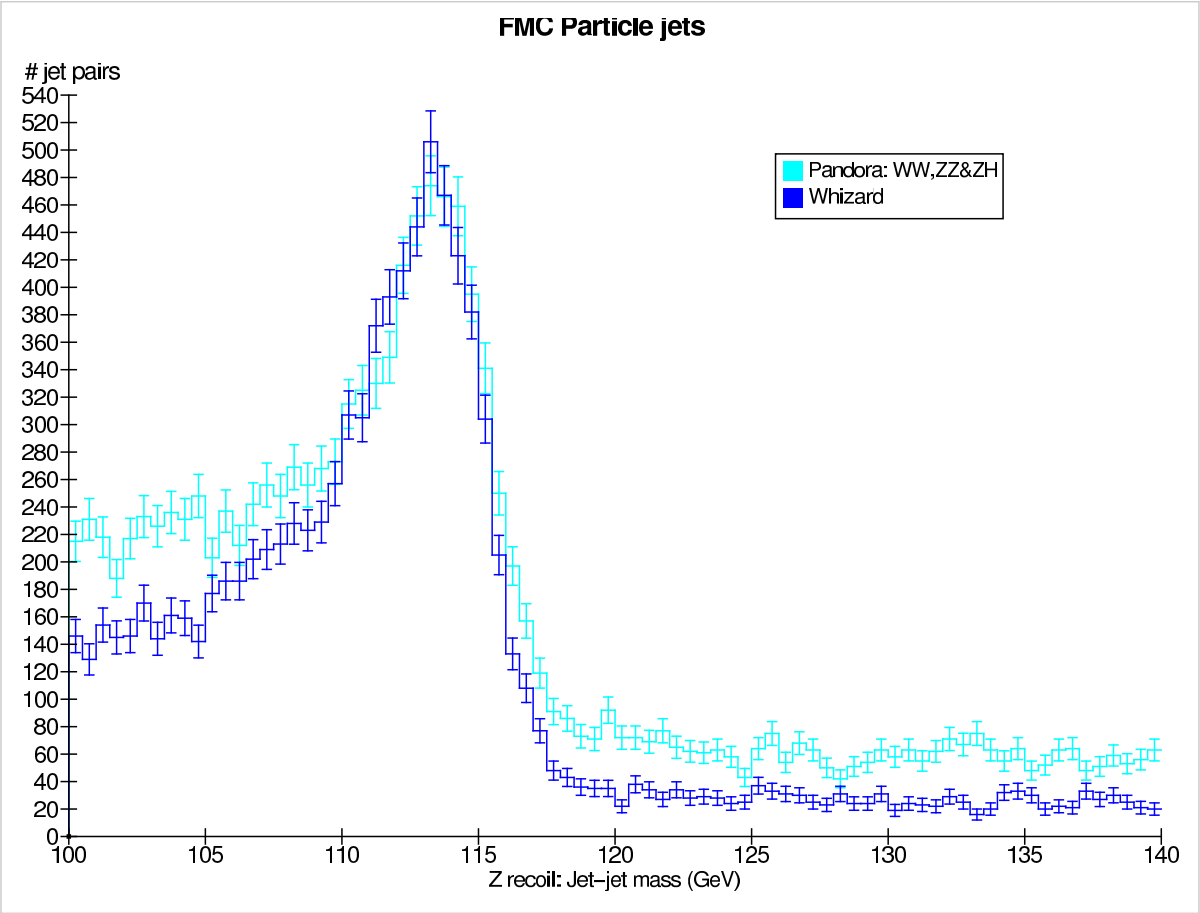
b.) Angular distribution ($\cos(\theta)_{Max}$) of jets

c.) Jet finder final "ycut" value

d.) Direct reconstruction of Z and Higgs through hadronic decays

Good agreement between Pandora, Pythia and Whizard Higgsstrahlung event simulation.

# Pandora / Whizard background comparison




Comparison of the Higgsstrahlung signal, and WW and ZZ multi-jet background simulations from Pandora-Pythia V2.2 and Whizard V1.22 is given for the FMC Particle jet-jet mass distribution recoiling against a reconstructed Z hadronic decay.

The low mass tail is in part due to ZZ backgrounds, while the flat distribution at high mass is due to $e^+e^- \rightarrow WW$ backgrounds. The disagreement between Pandora and Whizard may be partly due to the different machine assumptions of NLC500 and TESLA, respectively.

# Detector Models

- LCD Fast Monte Carlo V1.4 (hep.lcd.mc.zfast) includes
  - charged particle momentum smearing based on detailed error estimates,
  - gaussian energy smearing for photons and neutral hadrons,
  - acceptance and energy threshold requirements and
  - perfect energy flow.

- TESLA SimDet V4.0 includes
  - parameterized charged and neutral energy smearing based on full (Brahms) Monte Carlo simulations,
  - acceptance requirements and
  - a new energy flow algorithm.

- JLC QuickSim V02.1 includes
  - charged particle momentum and position smearing based on detailed error estimates,
  - simulation of individual calorimeter cell hits and cluster finding,
  - track-cluster association to separate charged and neutral clusters and
  - ...

# Detector Simulation Modules

- SimDet hep.tesla.simdet

```
To create an interface to SimDet
  simdet = new SimDet();
To get the results of SimDet's simulation
  NEntries = simdet.getNEFLOW();
  for (int n=0; n<NEntries; n++) {
    entry = simdet.getEFLOW(n);
    ...
```

Application: hep.lcd.mc.simdet

```
In a LCD framework driver
  simulation = new SimDetModule();
In finding SimDet jets
  cl = (ClusterList) event.get("SimDet E-flow objects");
```

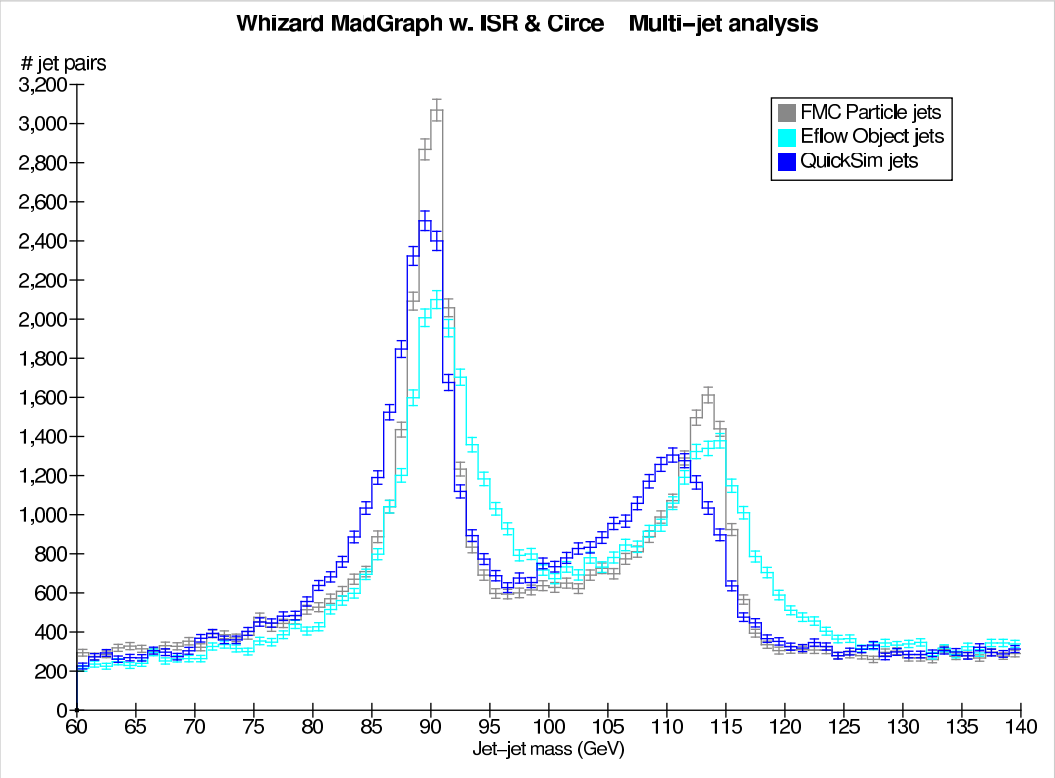
- QuickSim hep.jlc.quicksim

```
To create an interface to JLC's QuickSim
  quicksim = new QuickSim();
To get the results of QuickSim's simulation
  NTracks = quicksim.getNCmbTrks();
  for (int n=0; n<NTracks; n++) {
    track = quicksim.getCombinedTrack(n);
    ...
```

Application: hep.lcd.mc.quicksim

```
In a LCD framework driver
  simulation = new QuickSimModule();
In finding QuickSim jets
  cl = (ClusterList) event.get("QuickSim Combined tracks");
```

# U.S. FMC, SimDet & QuickSim Detector Simulations




Direct reconstruction of Z and Higgs through hadronic decays is shown for Higgsstrahlung signal events only. Jet-jet mass distributions for U.S. LCD Fast Monte Carlo (FMC), TESLA SimDet and JLC QuickSim detector simulations are reconstructed for Whizard-MadGraph Monte Carlo events including ISR and Circe beamstrahlung effects.

The LCD FMC jet-jet mass resolution is significantly better since it assumes "perfect" energy flow. TESLA SimDet and JLC QuickSim detector simulations give comparable jet energy resolutions but different mean reconstructed jet-jet masses.

# Framework Modules

- JLC Software Framework (JSF) hep.jlc.jsf

```
To create an interface to JSF
  framework = new JLCSoftwareFramework();

To create a new JSF package
  framework.newJSF();

To set up the JLC LCFull interface
  framework.setupLCFull();

To add a JSF Physics generator
  framework.addGenerator(type);
```

Application: hep.lcd.framework.jsf

```
To create a JLC Software/Study Framework interface
  jsf = new JLCSoftwareFramework();

To create a JSF event generator
  generator  = new JSFPythiaEventGenerator(processName, jsf);

To add the JSF QuickSim detector simulation
  simulation = new JSFQuickSimModule(jsf);
  add(simulation);

In finding QuickSim jets
  cl = (ClusterList) event.get("QuickSim Combined tracks");
```

# Summary

- MC generators available for LC studies: Pandora, Pythia and Whizard and KK2f.

- Good overall agreement:
  - Pandora, Pythia and Whizard signal generation are in good agreement. Need to understand differences in Pandora and Whizard $e^+e^- \rightarrow WW$ background simulations.
- Need to develop LCD FMC detector simulation model:
  - improve calorimeter clustering model,
  - simulate effects due to "realistic" energy flow algorithm, and
  - add secondary vertex finding (ZVTop) and b-tagging.

- All tools uses in this study will be available from http://www.lbl.gov/∼ronan/docs/lcstudies