

org.lcsim

Reconstruction and Analysis framework for ILC Detectors

Tony Johnson

SLAC

July 2006

org.lcsim: Contents

- Overview/Goals
 - Geometry/Conditions/Detector system
 - Reconstruction overview/status
 - Using org.lcsim with JAS3
 - Using org.lcsim with WIRED4
 - Becoming an org.lcsim Developer
 - Where next?
-

org.lcsim Goals

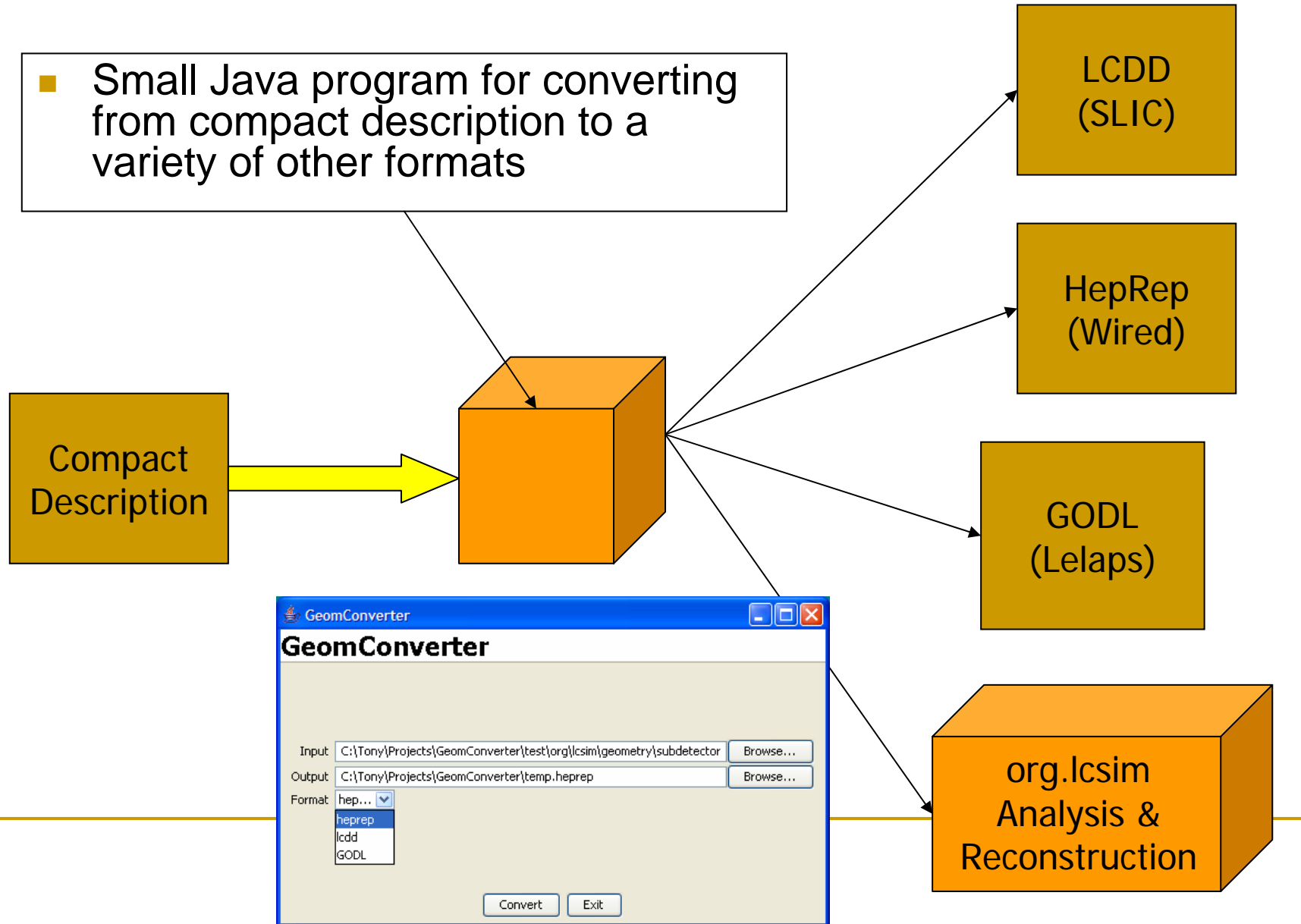
- “Second generation” ILC reconstruction/analysis framework
 - Builds on hep.lcd framework used since 1999
 - Full suite of reconstruction and analysis tools
 - Uses LCIO for IO and as basis for simulation, raw data and reconstruction event formats
 - Isolate users from raw LCIO structures
 - Maintain full interoperability with other LCIO based packages
 - Detector Independence
 - Make package independent of detector, geometry assumptions so can work with any detector
 - Read properties of detectors at runtime
 - Written using Java (1.5)
 - High-performance but simple, easy to learn, OO language
 - Enables us last 10 years of software developments in the “real world”
 - Ability to run standalone (command line or batch) or in JAS3 or IDE such as Netbeans, Eclipse
-

org.lcsim: Compact Geometry Description

- org.lcsim uses “Compact Geometry Description” to define detector
 - Simple XML format for describing ILC detectors
 - Handles typical ILC detector geometries
 - Range of detectors handled is extensible (by writing Java modules)
 - Allows rapid prototyping of new detector geometries
 - Does not require network access or installation of database software to run
 - Automatic generation of full Geant4 LCDD geometry for full compatibility with SLIC
-

org.lcsim: Geometry Converter

- Small Java program for converting from compact description to a variety of other formats

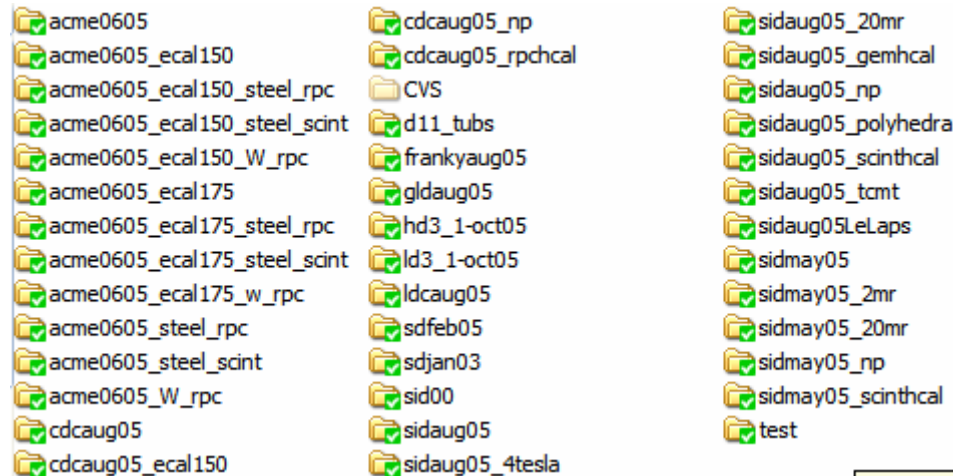


org.lcsim Conditions Data

- Provide access to a extensible set of conditions for each detector including:
 - Detector Geometry
 - Algorithm Specific Constants
 - E.g. FastMC smearing parameters
 - Doesn't make assumptions about format of data
 - Doesn't rely on internet access, or local database installation
 - Detector Constants stored in .zip file
 - Typically contains:
 - Compact geometry file
 - Set of (ascii) constants for standard algorithms
 - Can additionally contain:
 - Arbitrary files (xml, ascii, binary) needed by other algorithms
 - Other geometry formats (HepRep, LCDD)
 - Full fieldmap
 - To define a new detector just create a new .zip file.
-

Available Detector Descriptions

- Although detector descriptions can live anywhere we maintain a CVS repository of detector descriptions
 - Exported to org.lcsim web site for automatic download
- 40 detector variants as of July 2006
- Many SiD variants, but also some gld, ldc



- You are welcome to contribute more

Org.lcsim Reconstruction

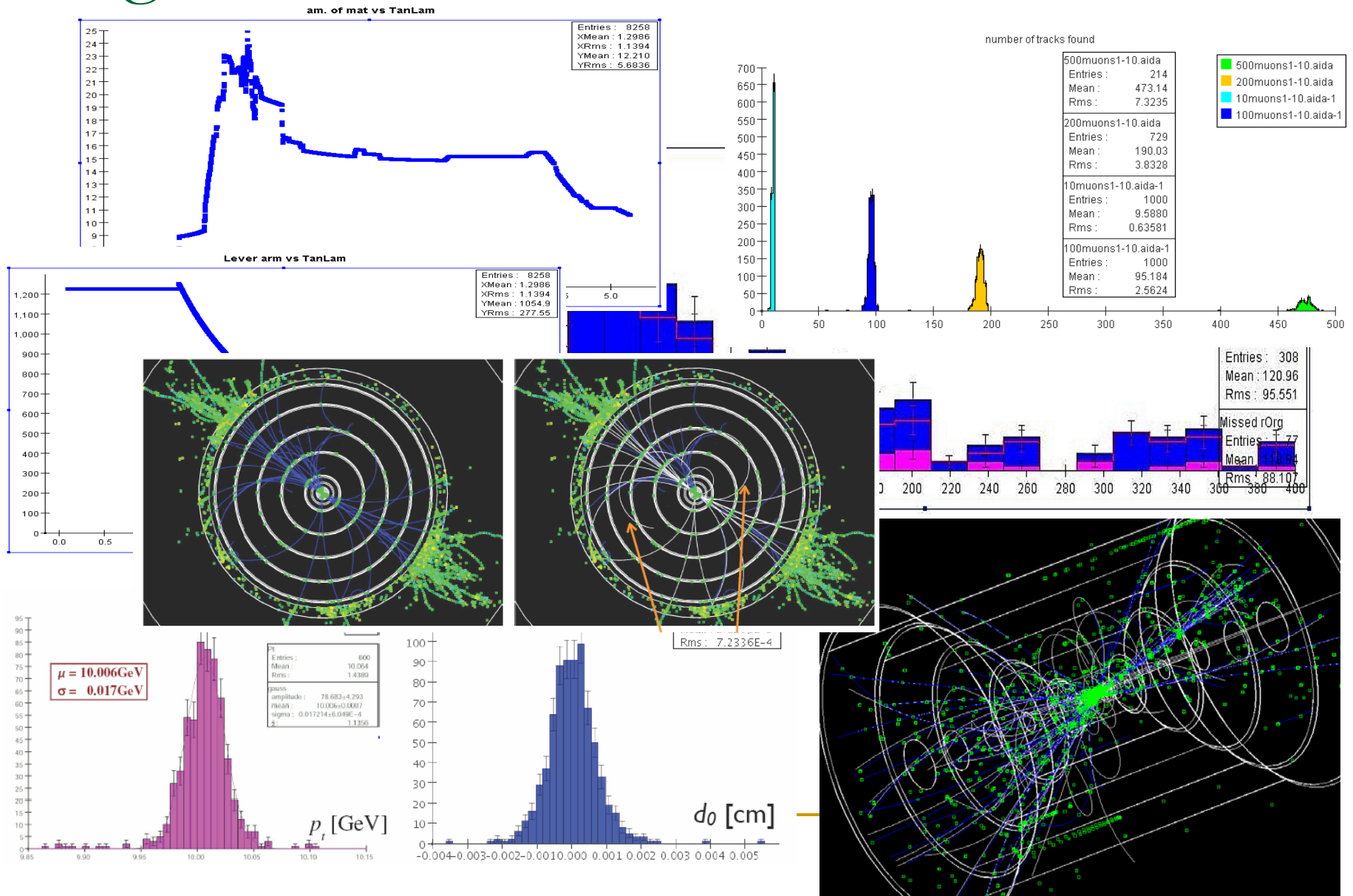
- Reconstruction package includes:
 - Physics utilities:
 - Jet finders, event shape routines
 - Diagnostic event generator, stdhep reader/translator
 - Histogramming/Fitting/Plotting (AIDA based)
 - Event Display
 - Processor/Driver infrastructure
 - Fast MC
 - Track/Cluster smearing
 - Reconstruction
 - Cheaters (perfect reconstruction)
 - Detector Response
 - CCDSim, Digisim
 - Clustering Algorithms
 - Cheater, DirectedTree, NearestNeighbour, Cone
 - Tracking Finding/Fitting Algorithms
 - TRF,
 - Muon Finding, Swimming
 - Vertex Finding (ZvTop)
-

org.lcsim: Contrib Area

- Goal of org.lcsim is not to provide “A single reconstruction package” but rather a framework into which reconstruction algorithms can be plugged.
 - We encourage users to contribute code to the “contrib” area as soon as possible.
 - Important to encourage collaboration, reuse, and as learning tool.
 - Many contributions added in last year:
 - HMatrix cluster analysis
 - VertexFitter
 - PFA algorithms/template
 - SODTracker
 - Garfield Tracker
 - Calorimeter Cell Ganging
 - FastMC improvements
 - Tracking finding/fitting
 - MIP Finder
 - Minimum Spanning Tree Clustering
-

org.lcsim results

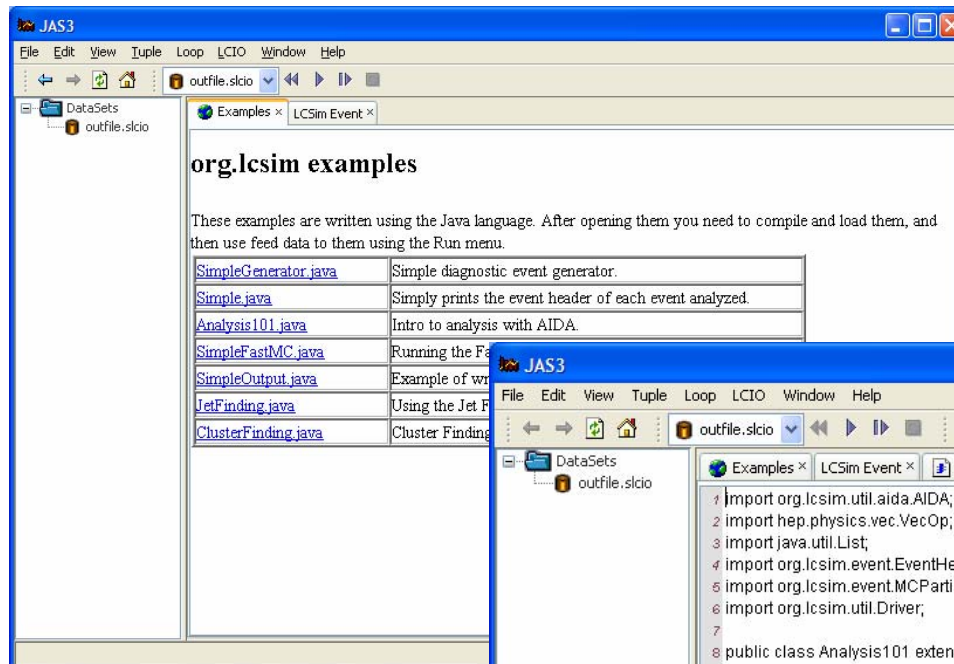
(See many other talks at this workshop)



Using org.lcsim with JAS3

- The org.lcsim can be used standalone, within an IDE, or inside JAS3. Same code can be used in all modes, so easy to move back and forth
 - E.g. develop in IDE and run in JAS3
 - E.g. develop in JAS3 and run in batch
 - JAS3 org.lcsim plugin adds:
 - Example Analysis Code
 - org.lcsim Event browser
 - Easy viewing of analysis plots
 - WIRED event display integration
-

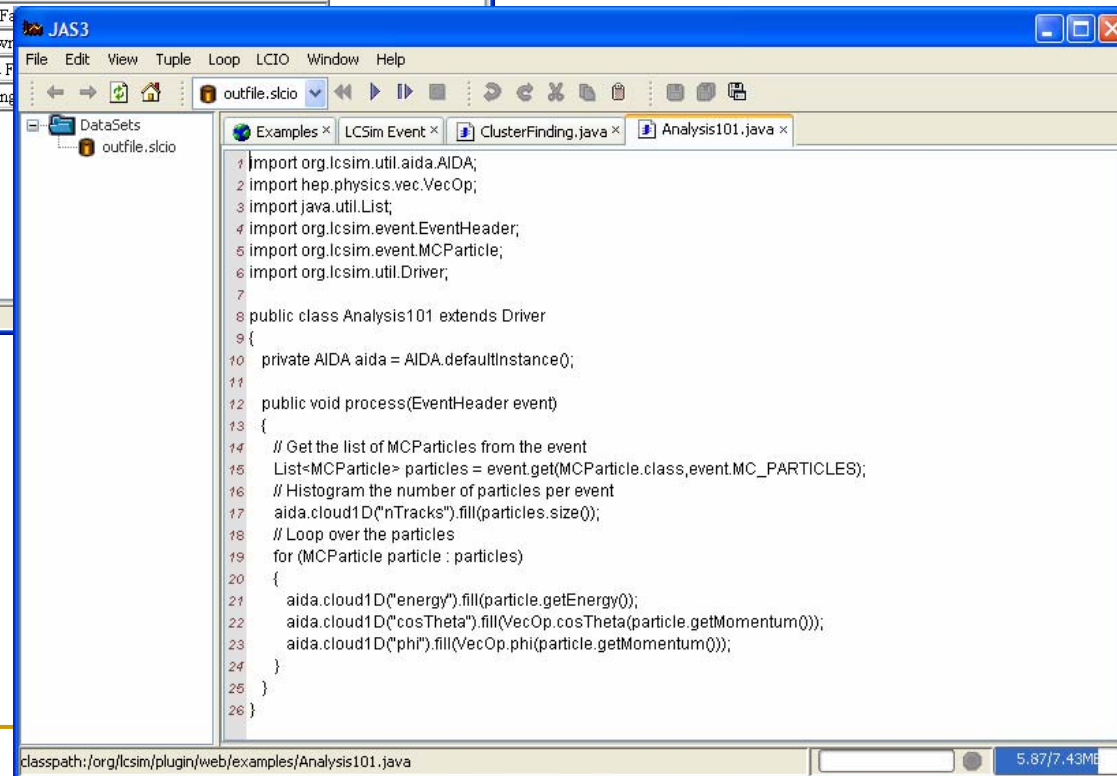
org.lcsim: Examples



org.lcsim examples

These examples are written using the Java language. After opening them you need to compile and load them, and then use feed data to them using the Run menu.

SimpleGenerator.java	Simple diagnostic event generator.
Simple.java	Simply prints the event header of each event analyzed.
Analysis101.java	Intro to analysis with AIDA.
SimpleFastMC.java	Running the FastMC
SimpleOutput.java	Example of writing output
JetFinding.java	Using the Jet FINDER
ClusterFinding.java	Cluster Finding



```
1 import org.lcsim.util.aida.AIDA;
2 import hep.physics.vec.VecOp;
3 import java.util.List;
4 import org.lcsim.event.EventHeader;
5 import org.lcsim.event.MCParticle;
6 import org.lcsim.util.Driver;
7
8 public class Analysis101 extends Driver
9 {
10     private AIDA aida = AIDA.defaultInstance();
11
12     public void process(EventHeader event)
13     {
14         // Get the list of MCParticles from the event
15         List<MCParticle> particles = event.get(MCParticle.class,event.MC_PARTICLES);
16         // Histogram the number of particles per event
17         aida.cloud1D("nTracks").fill(particles.size());
18         // Loop over the particles
19         for (MCParticle particle : particles)
20         {
21             aida.cloud1D("energy").fill(particle.getEnergy());
22             aida.cloud1D("cosTheta").fill(VecOp.cosTheta(particle.getMomentum()));
23             aida.cloud1D("phi").fill(VecOp.phi(particle.getMomentum()));
24         }
25     }
26 }
```

classpath:/org/lcsim/plugin/web/examples/Analysis101.java 5.87/7.43MB

org.lcsim: Examples

The screenshot shows the JAS3 interface with the following components:

- File Explorer:** Shows a tree view with folders 'DataSets', 'outfile.slcio', 'Programs', and 'aida31133aida'.
- Event Header:** A table with the following data:

Run	0
Event	0
Time Stamp	Fri Mar 11 14:25:13 PST 2005
Detector Name	sdjan03
- Blocks:** A table listing event blocks with their names and types:

Name	Type
HcalEndcapHitsNNClusters	org.lcsim.recon.cluster.nn.NearestNeighborCluster
HcalBarrHitsNNClusters	org.lcsim.recon.cluster.nn.NearestNeighborCluster
EcalEndcapHitsNNClusters	org.lcsim.recon.cluster.nn.NearestNeighborCluster
EcalBarrHitsNNClusters	org.lcsim.recon.cluster.nn.NearestNeighborCluster
MuonEndcapHitsNNClusters	org.lcsim.recon.cluster.nn.NearestNeighborCluster
LumEndcapHits	org.lcsim.util.lcid
MCParticle	org.lcsim.event

Analyzed 1 records in 406ms

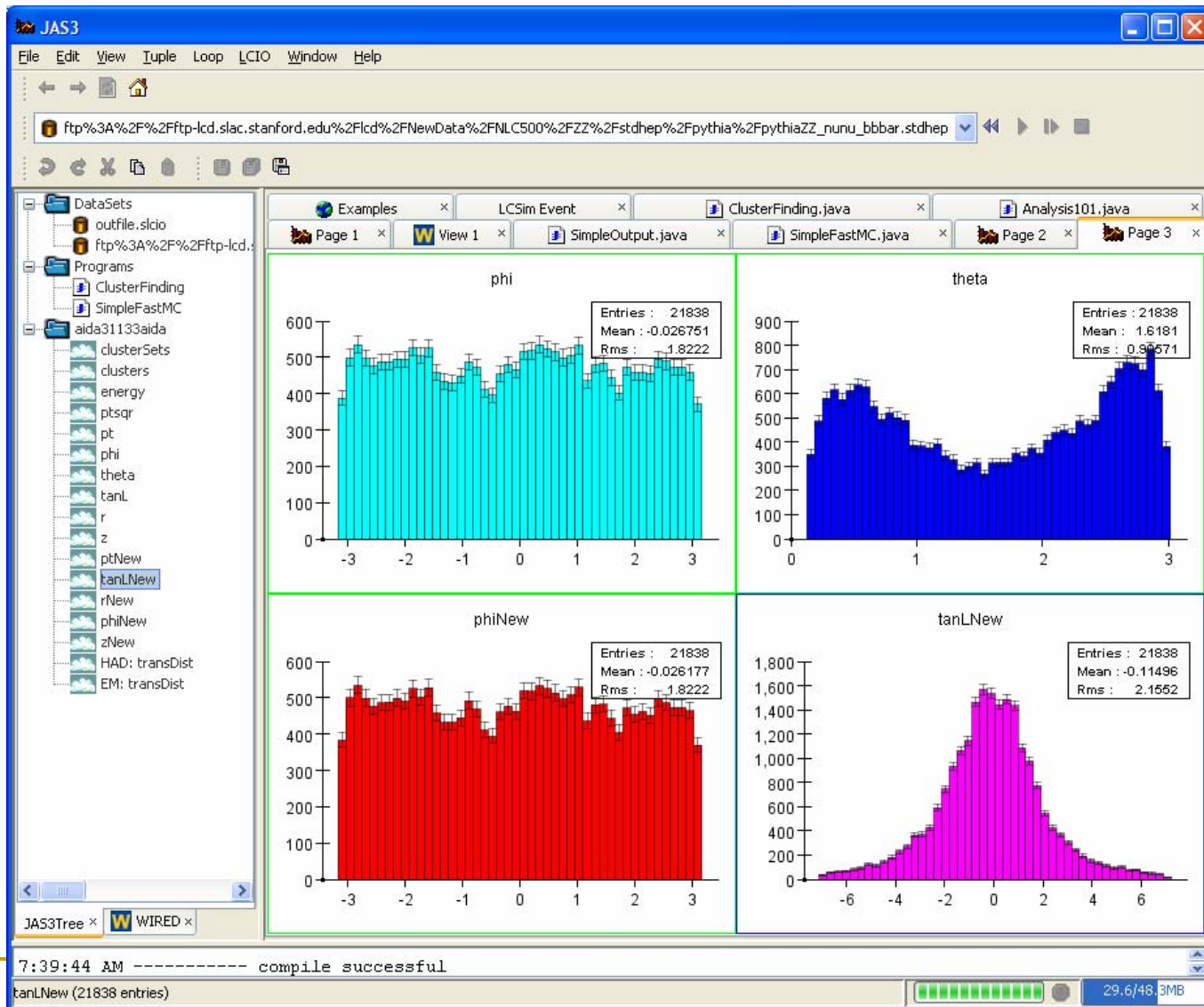
The screenshot shows the JAS3 interface with a table of EcalBarrHits data. The table has the following columns: layer, system, barrel, theta, phi, energy, x, y, z. The data is as follows:

layer	system	barrel	theta	phi	energy	x	y	z
0	2	0	333	1595	4.0386E-4	1210.1	-395.70	426.89
1	2	0	333	1594	1.1317E-4	1213.4	-401.80	428.57
9	2	0	341	1593	6.0089E-5	1249.8	-419.05	398.53
1	2	0	333	1595	.00251117	1214.9	-397.26	428.57
2	2	0	333	1595	3.3759E-4	1219.7	-398.81	430.24
0	2	0	416	881	1.1273E-4	-1257.9	-196.82	16.667
1	2	0	416	880	3.5485E-4	-1263.6	-192.87	16.733
2	2	0	416	880	1.1914E-4	-1268.5	-193.62	16.798
3	2	0	416	880	1.0678E-4	-1273.5	-194.38	16.863
4	2	0	416	880	1.3202E-4	-1278.4	-195.13	16.929
5	2	0	416	880	1.0821E-4	-1283.3	-195.89	16.994
6	2	0	416	880	1.4717E-4	-1288.3	-196.64	17.060
7	2	0	416	880	1.1575E-4	-1293.2	-197.40	17.125
8	2	0	416	880	1.2397E-4	-1298.2	-198.15	17.191
9	2	0	416	880	1.3174E-4	-1303.1	-198.90	17.256
10	2	0	416	879	1.1775E-4	-1308.8	-199.77	17.322
11	2	0	416	879	1.3348E-4	-1313.7	-199.50	17.387
12	2	0	416	879	3.6082E-4	-1318.7	-196.24	17.453
13	2	0	416	879	1.1621E-4	-1323.6	-196.97	17.518
14	2	0	416	879	1.0455E-4	-1328.6	-197.71	17.583
15	2	0	416	879	1.0607E-4	-1333.5	-198.45	17.649
16	2	0	416	879	1.2895E-4	-1338.5	-199.18	17.714
17	2	0	416	879	1.2762E-4	-1343.4	-199.92	17.780
18	2	0	416	878	1.0828E-4	-1348.4	-200.65	17.845

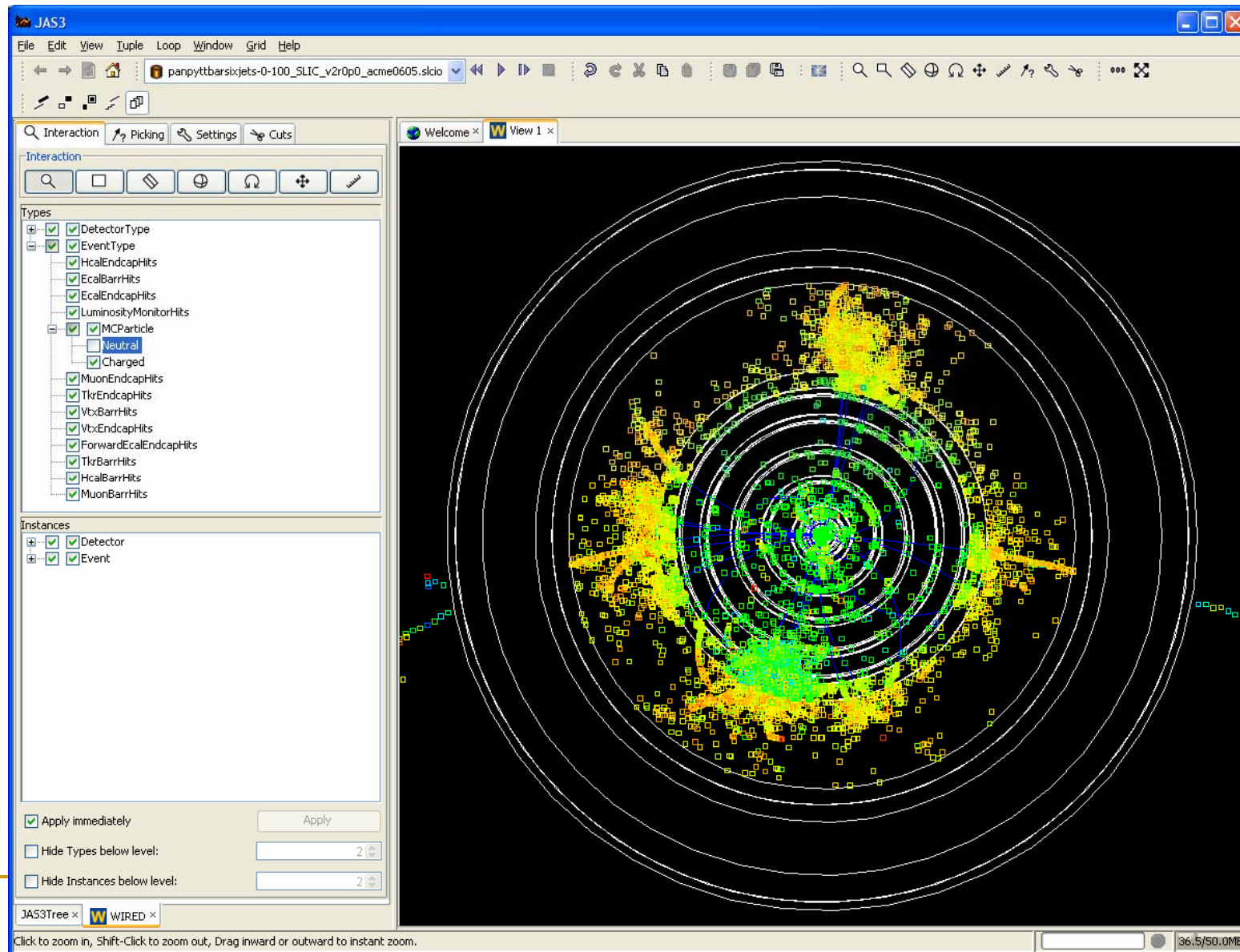
Analyzed 1 records in 406ms

7.22/7.43MB

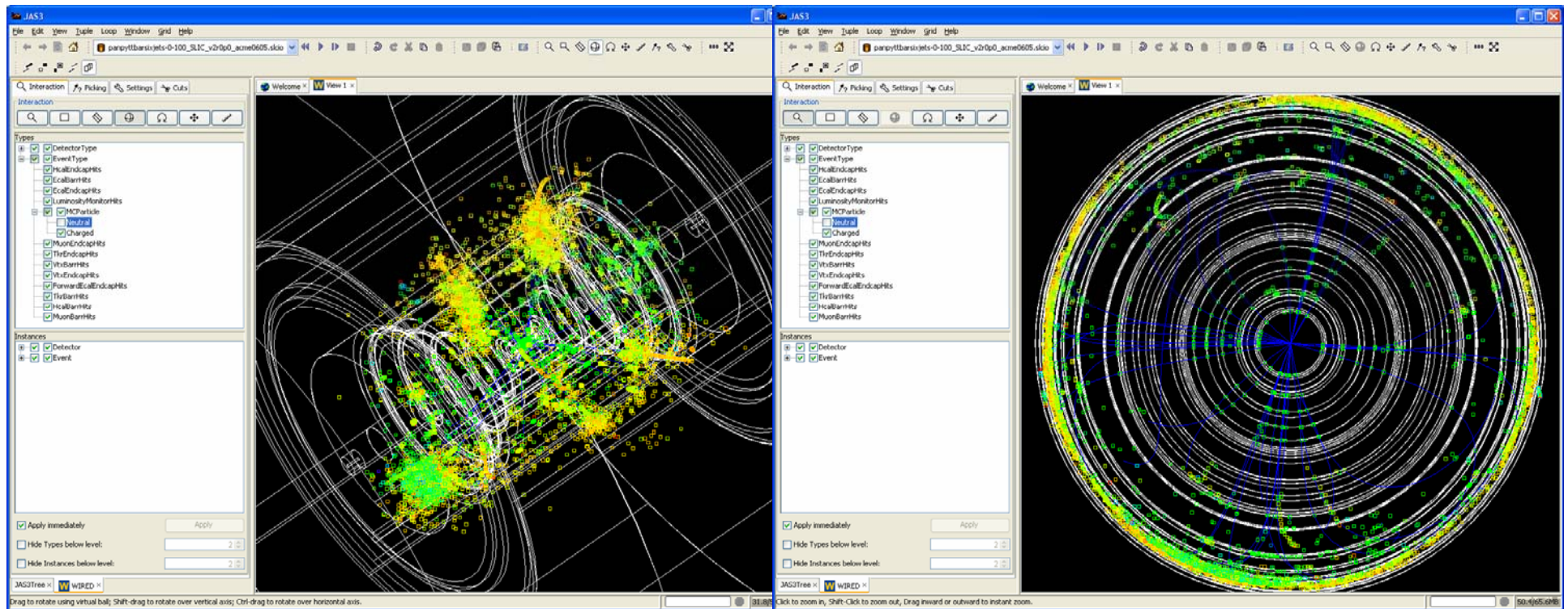
org.lcsim: Plot Viewing



Using org.lcsim with WIRED4



Using org.lcsim with WIRED4



Using org.lcsim with WIRED4

The screenshot displays the JAS3 software interface. The main window shows a complex, circular particle detector simulation visualization with numerous concentric rings and radial lines, representing the detector's geometry and particle paths. The interface includes a menu bar (File, Edit, View, Tuple, Loop, Window, Grid, Help), a toolbar with various icons, and a sidebar with several panels:

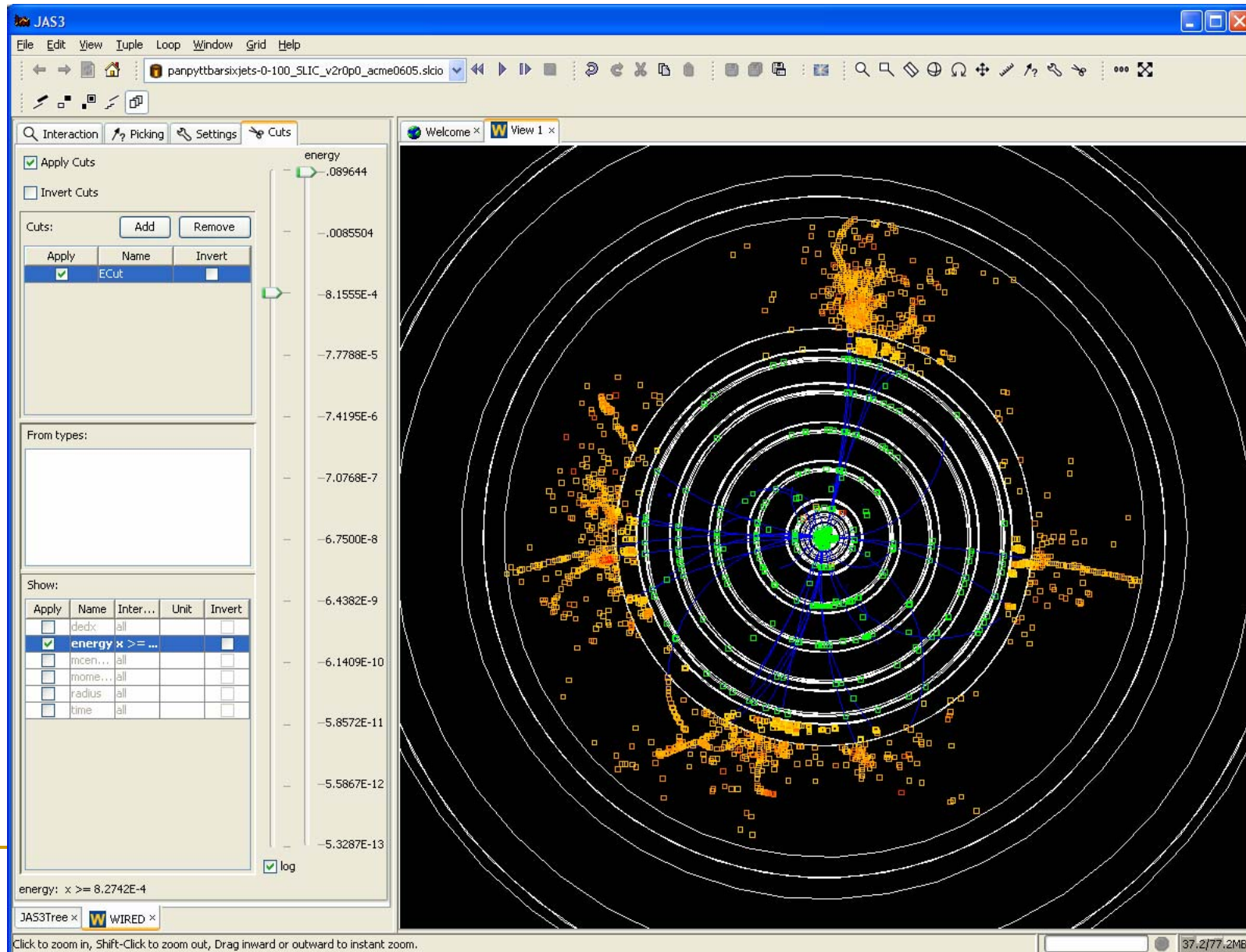
- Interaction**: Contains buttons for Shape, Actions / Settings, and a checkbox for "Pick while Moving/Dragging".
- Picked objects (1):** A table showing the selected object's details.
- Attributes of picked object (9):** A table listing various attributes and their values.

The status bar at the bottom indicates the current memory usage: 61.5/65.6MB.

Type	Points	Children
VtxEndcapHits	1	0

Name	Value	Unit	Node
MarkName	Box		Type
color			Type
dEdx	1.8306E-5		Instance
drawAs	Point		Type
fill	<input checked="" type="checkbox"/>		Type
fillColor			Type
layer	Hits		Type
mcEnergy	.030593		Instance
time	2328.6		Instance

Using org.lcsim with WIRED4



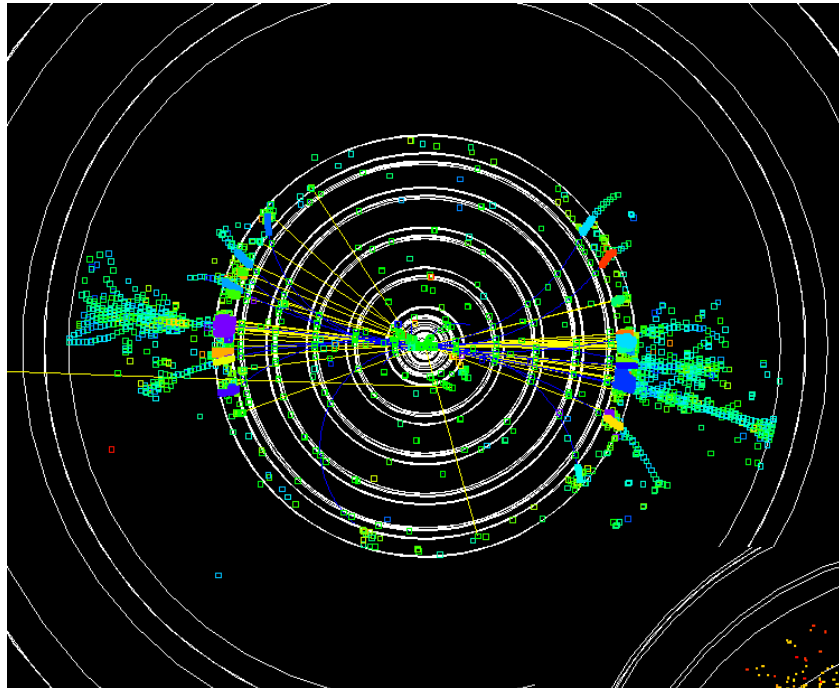
Using org.lcsim with WIRED4

The screenshot displays the JAS3 software interface. The main window shows a visualization of a particle detector simulation, likely a calorimeter, with concentric circular layers and a central vertex. The visualization is overlaid with a network of colored lines and nodes, representing particle tracks and their interactions. The interface includes a menu bar (File, Edit, View, Tuple, Loop, Window, Grid, Help), a toolbar with various icons, and a sidebar with several panels:

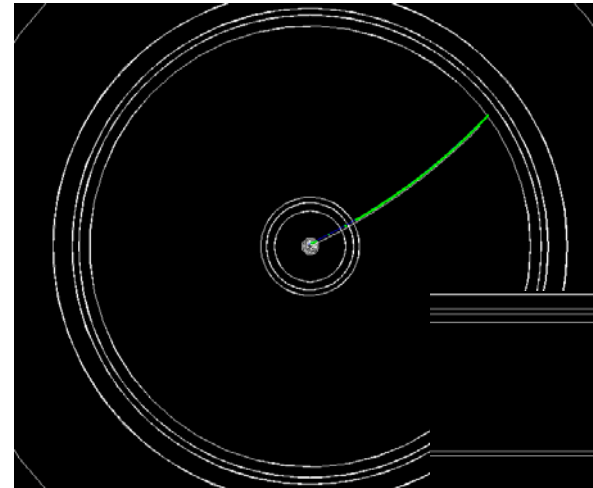
- Interaction:** Contains icons for search, pick, settings, and cuts.
- Types:** A list of detector and event types with checkboxes, including: DetectorType, EventType, TkrEndcapHits, VtxEndcapHits, EcalEndcapHitsNNClusters, VtxBarrHits, EcalBarrHits, LuminosityMonitorHits, MuonEndcapHits, HcalEndcapHits, HcalBarrHitsNNClusters, MuonBarrHitsNNClusters, MuonEndcapHitsNNClusters, HcalEndcapHitsNNClusters, TkrBarrHits, ForwardEcalEndcapHits, and MuonBarrHits.
- Instances:** A list of instances with checkboxes, including: Detector and Event.
- Apply immediately:** A checkbox and an 'Apply' button.
- Hide Types below level:** A checkbox and a dropdown menu set to '2'.
- Hide Instances below level:** A checkbox and a dropdown menu set to '2'.

At the bottom of the window, there is a status bar with the text: "7:43:50 AM ----- compile successful". Below the status bar, there is a message: "Click to zoom in, Shift-Click to zoom out, Drag inward or outward to instant zoom." and a memory usage indicator: "49.6/83.1Mb".

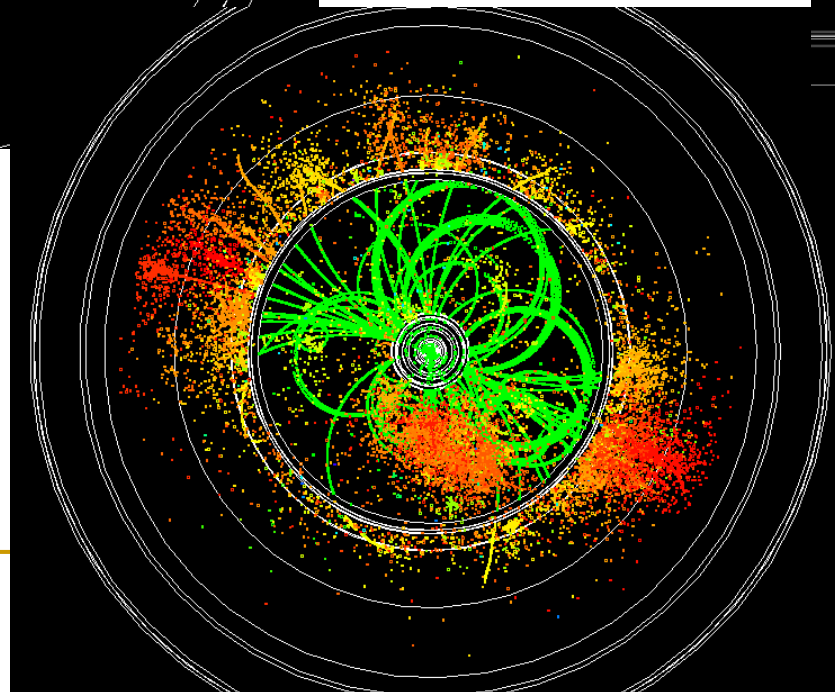
Interoperability



SiD



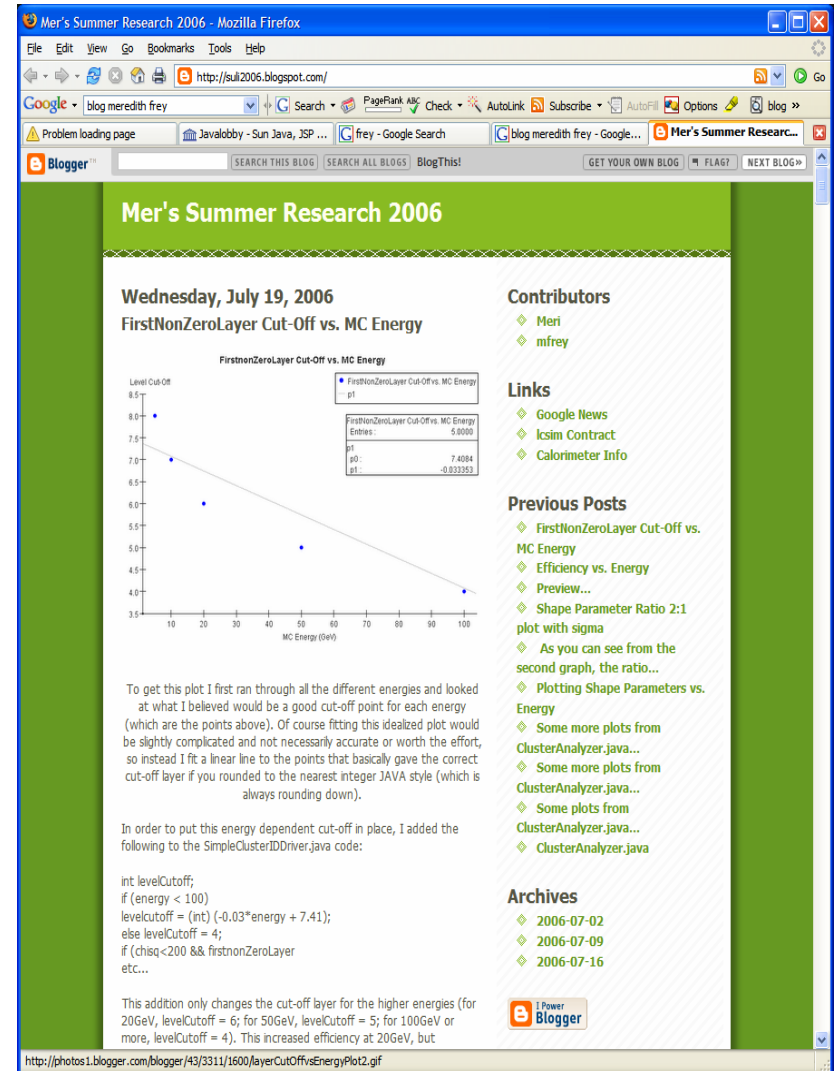
LDC



GLD

How hard is it to get started with org.lcsim?

- Works on Linux, MacOSX, Windows
 - Should take about 15 minutes to install JAS3 and org.lcsim plugin.
- Case Study: SLAC Summer student
 - 2 semesters of Java experience
 - (no C++, Fortran etc)
 - Using tutorial on lcsim.org Wiki; installed software, downloaded data, and got useful results in one day (and fixed a few errors in the documentation along the way).
 - Regular analysis updates have been appearing on her blog ever since!
- Even if you don't have Java experience you can get started almost as fast
 - (the only thing you will miss is the core dumps)
- Start here:
 - <https://confluence.slac.stanford.edu/display/ilc/lcsim+Getting+Started>
 - Problems? Attend Tuesday afternoon "Simulation" phone meeting or use discussion forum at <http://forum.linearcollider.org/>



Becoming an org.lcsim developer

- To get started you just need “Java”, “cvs”, “maven”
 - Maven is a Java based project management tool
 - Single command “maven”
 - downloads dependencies, compiles code, runs tests, deploys code
 - All code in CVS
 - To check-out and build all code:
 - set CVSROOT=“pserver:anonymous@cvs.freehep.org:/cvs/lcsim”
 - cvs co GeomConverter
 - cd GeomConverter
 - maven
 - cd ..
 - cvs co lcsim
 - cd lcsim
 - maven
 - Find more documentation at:
 - <http://lcsim.org/>
 - Read/Contribute to the Wiki at: <https://confluence.slac.stanford.edu/display/ilc/Home>
 - Discuss at: <http://forum.linearcollider.org/>
 - We strongly encourage developers to use IDE
 - Netbeans, Eclipse both free, easy to learn, very powerful
 - Use mevenide to teach IDEs about maven system
-

Using org.lcsim with Netbeans

The screenshot displays the NetBeans IDE 4.1 interface for the org.lcsim project. The left sidebar shows the project structure with various packages like org.lcsim.recon.tracking and org.lcsim.util. The main editor window shows the source code for Helix.java, which includes imports for java.lang.Math and java.lang.Object, and a public class Helix. A tooltip is visible over the equals method of the Hep3Vector class, providing a detailed description of the equals method's behavior and its properties: reflexive, symmetric, transitive, and consistent. The bottom of the IDE shows the VCS Output window, which is currently empty, indicating that the command has finished successfully.

```
import static java.lang.Math.asin;
import static java.lang.Math.PI;

/**
 * This class
 * All quanti
 * except for
 * @author to
 * @version $
 */
public class
{
    /** Create
     * @param
     * @param
     * @param
     * @param
     */
    public Helix(double lambda)
    {
        if (abs
        this.or
        this.ra
        this.ph
        // Calc
        cosLamb
        sinLamb
        xCenter
        yCenter
        phiToCe
        radius0
        origin
    }
}
```

java.lang.Object

public boolean equals(Object obj)

Indicates whether some other object is "equal to" this one.

The equals method implements an equivalence relation on non-null object references:

- It is reflexive: for any non-null reference value x, x.equals(x) should return true.
- It is symmetric: for any non-null reference values x and y, x.equals(y) should return true if and only if y.equals(x) returns true.
- It is transitive: for any non-null reference values x, y, and z, if x.equals(y) returns true and y.equals(z) returns true, then x.equals(z) should return true.
- It is consistent: for any non-null reference values x and y, multiple

Members View

- getDOCA(Hep3Vector point)
- getDistanceToInfiniteCylinder(double r)
- getDistanceToZPlane(double z)
- getMomentumAtLength(double s)
- getPointAtDistance(double alpha)
- getPositionAtLength(double s)

VCS Output

Type	Filename	Path
U	CalorimeterHITTableModel.java	src/org/lcsim/plugin/browser/CalorimeterHITTableModel.java

Where Next?

- Some clean-up of “Track/Track Parameters” and “Geometry” interface
 - Form org.lcsim clean-up “Task Force”
 - Jeremy and Jan Strube are seconded
 - Other volunteers welcome
 - Complete Tracking/Vertexing packages
 - Migrate some contrib code to main code base
 - We are close to complete tracking/PFA/vertexing/flavor tagging chain.
 - Should create fully simulated/reconstructed data to complement Fast MC studies
 - Interoperability
 - LCIO works nicely
 - Geometry interoperability remains elusive but highly desirable
 - Ability to call C++ (MarlinReco) modules from org.lcsim
 - Perhaps more possible with new version of SWIG
-

Conclusion

- org.lcsim Framework is mostly complete
 - If there are limitations which are impeding your work, let us know!
 - User contributed reconstruction software growing rapidly
 - Several more contributions promised soon
-