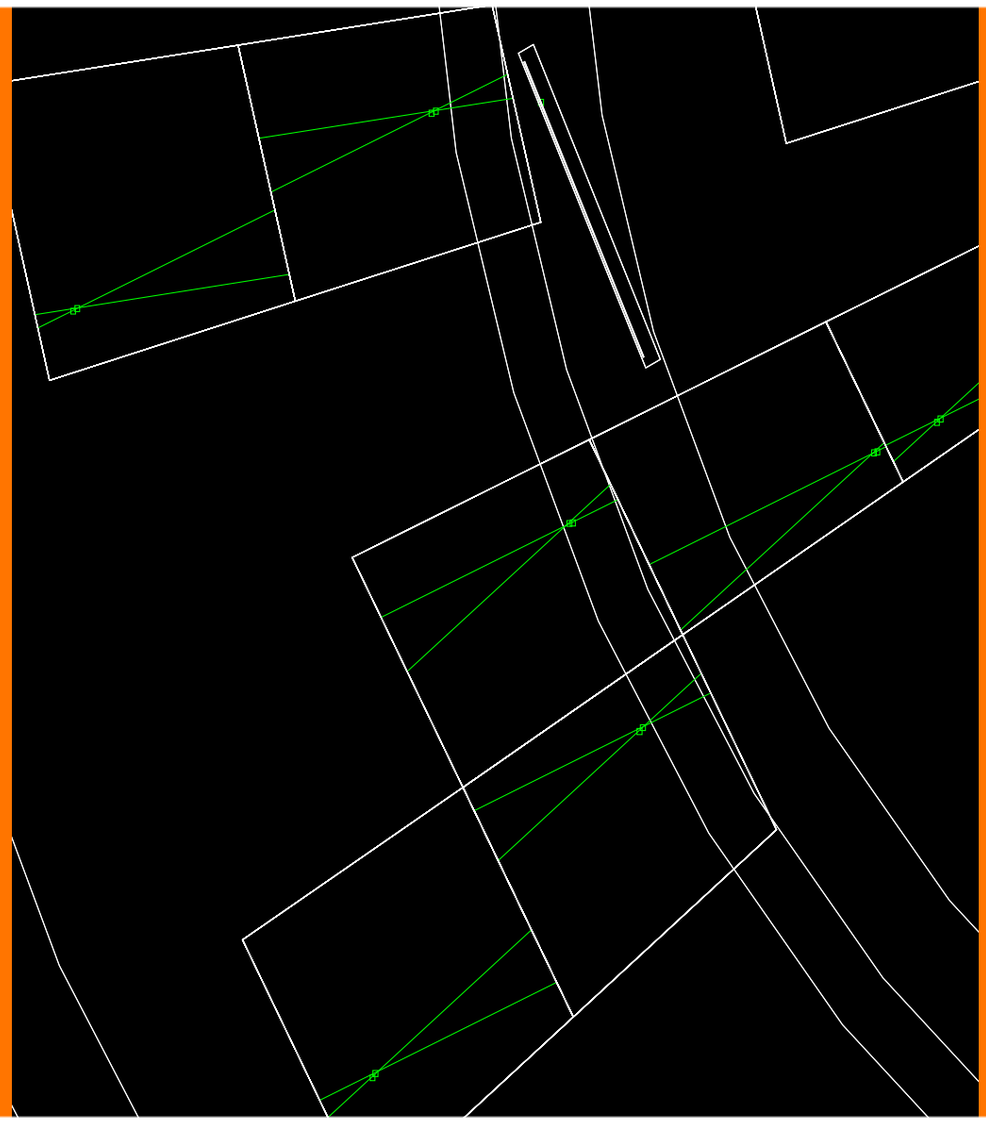
Status of Tracker Digitization and Hit Reconstruction



Tim Nelson, Jeremy McCormick, Cosmin Deaconu - SLAC
Nick Sinev - **U. OREGON**

SiD Workshop

RAL - April, 14 2008

A Little Caffeination

A little help for those who know some C++, but not used to Java-speak:

- Class: same as C++... some data and some methods that operate on it to do something
- SubClass *extends* BaseClass
- Interface: defines a set of methods for a class, but no method bodies (no "implementations": i.e. it only defines what kinds of actions can be done)
- Class *implements* Interface: means Class has all the methods that are defined for Interface (with method bodies: i.e. it actually ***does*** the actions defined by the interface).

➡ **Minimal interfaces are good... allow one to easily define which class gets used without changing much code**

```
Interface Coffemaker
{
   public Coffee brew(CoffeeBeans beans);
}

Class MrCoffee implements Coffeemaker
Class LaMarzocco implements Coffeemaker
```

```
Class Kitchen
{
   Coffeemaker _coffeemaker = new MrCoffee();
}
```

then, next year, after a raise...

```
Class Kitchen
{
   Coffeemaker _coffeemaker = new LaMarzocco();
}
```

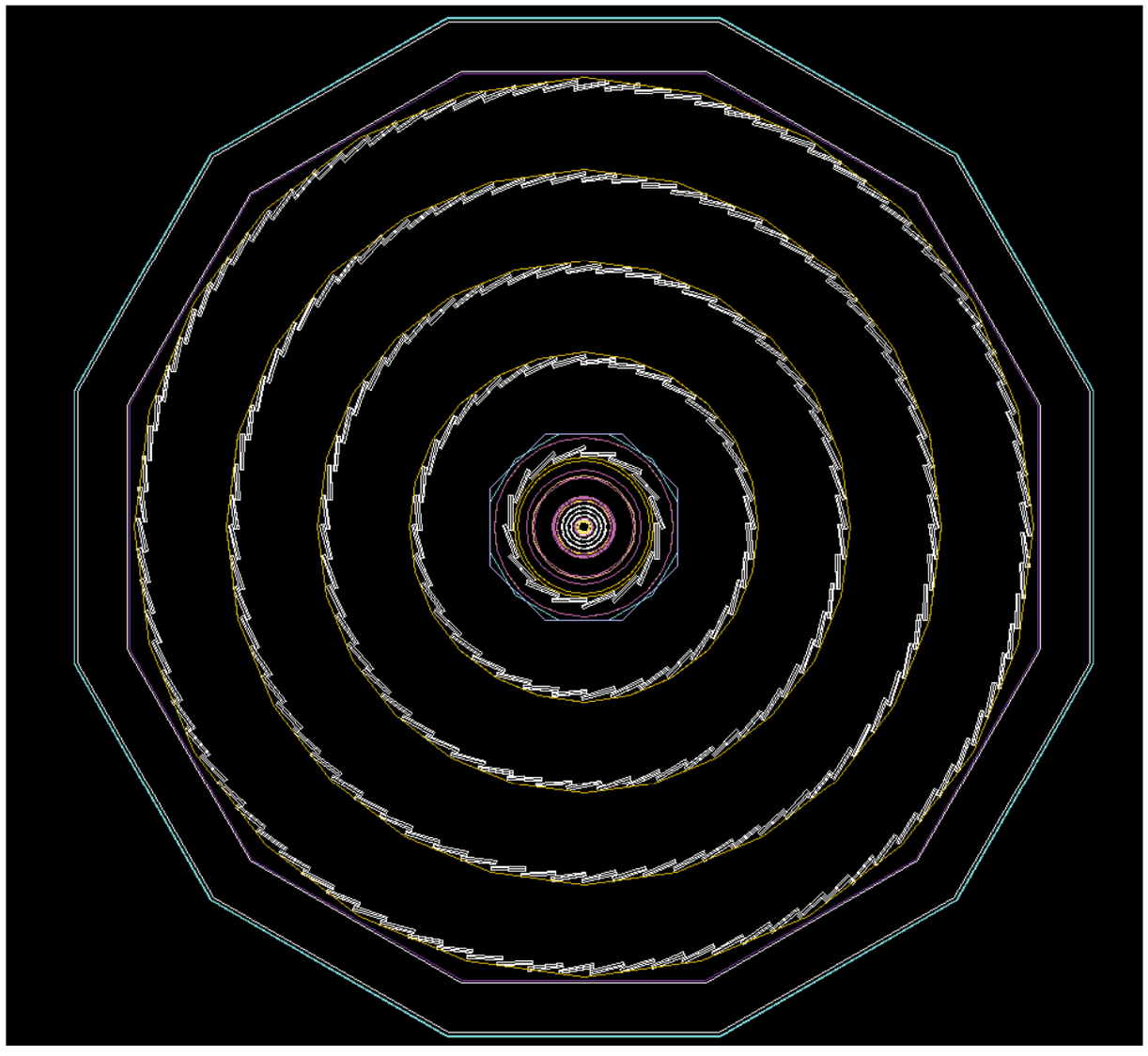
Overview

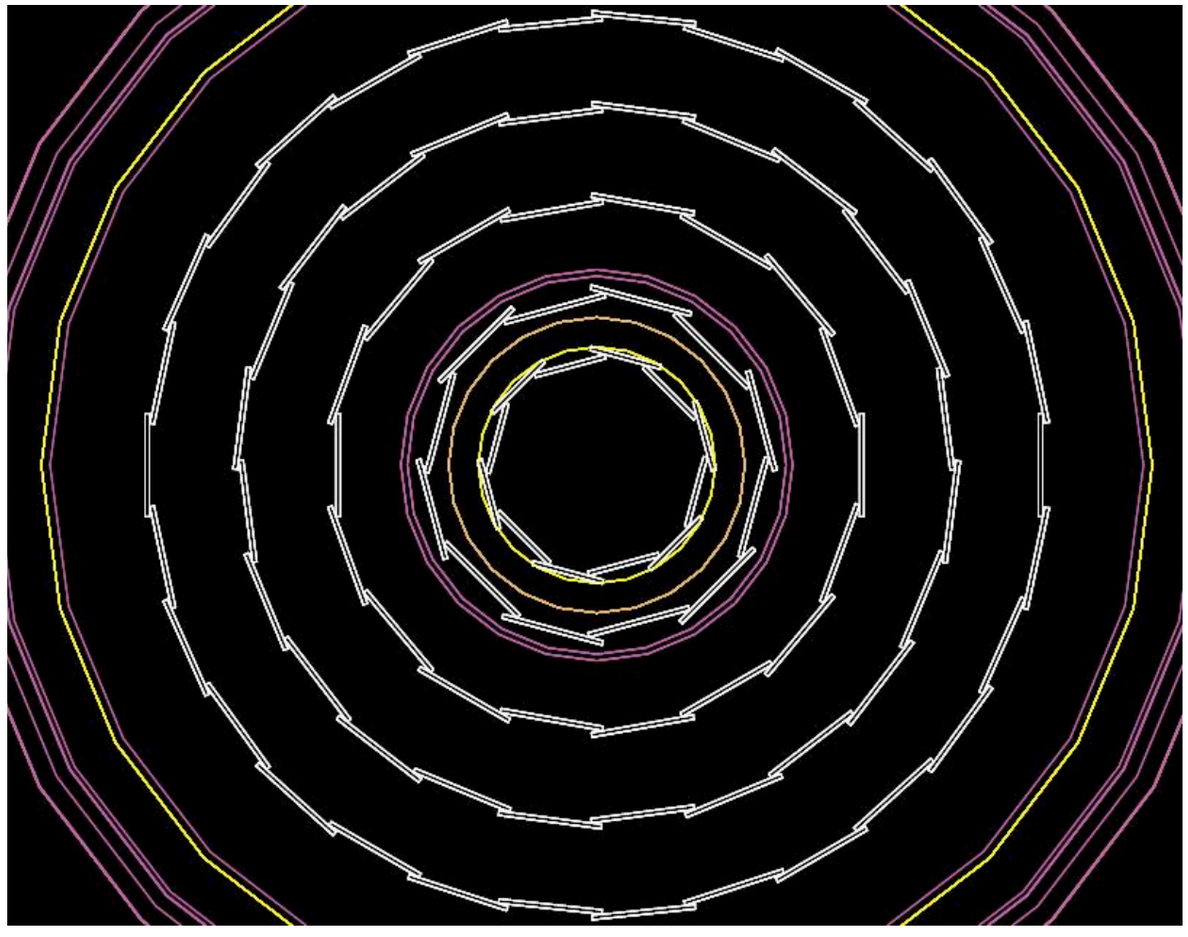
- Detector modeling / simulation ➡ SimTrackerHits (energy depositions as generated by GEANT)
- Silicon simulation / digitization ➡ RawTrackerHits (single-channel ADC values from detector readout)
- Hit clustering / reconstruction ➡ TrackerHits (clustered raw hits w/ position measurements)
- Pattern recognition / fitting ➡ Tracks (see next talk!)

Detector Modeling

- *Compact description for planar barrel trackers - SiTrackerBarrel*
  - Accommodates all layouts that have been discussed (including double-sided)
  - Specific models exist for outer tracker and vertex detector of baseline SiD
- *Compact description for planar endcap trackers - SiTrackerEndcap*
  - Can simulate any wedge-type geometry with any stereo angle (e.g. 90-degree)
  - Can be easily modified to to include tilts/overlaps for more realistic model
  - Specific “straw layouts” exist for endcap/forward trackers and vertex detector
- *Models are currently being updated to latest layouts and material estimates: these are drop-in replacements for the current sid01 tracking and vertexing*

SiTrackerBarrel Example

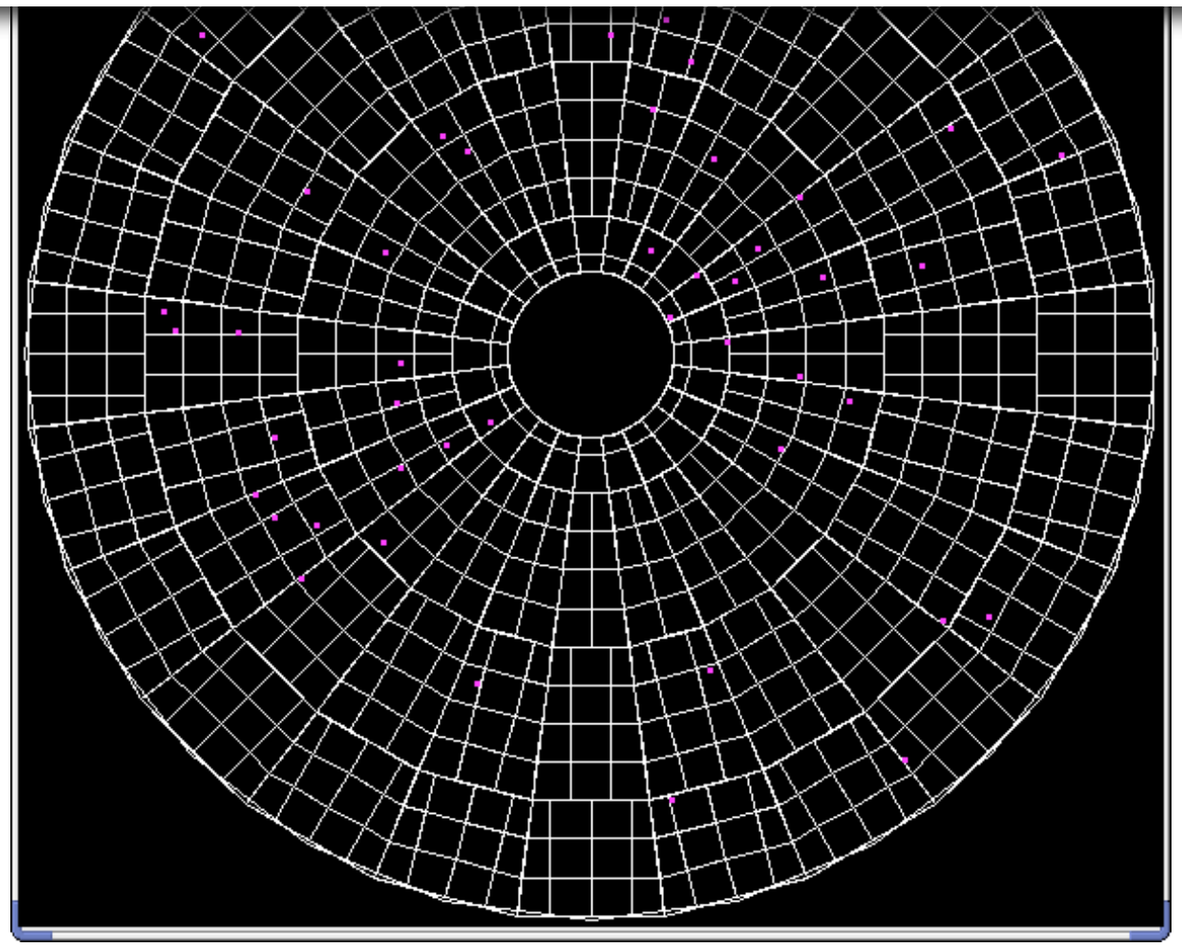


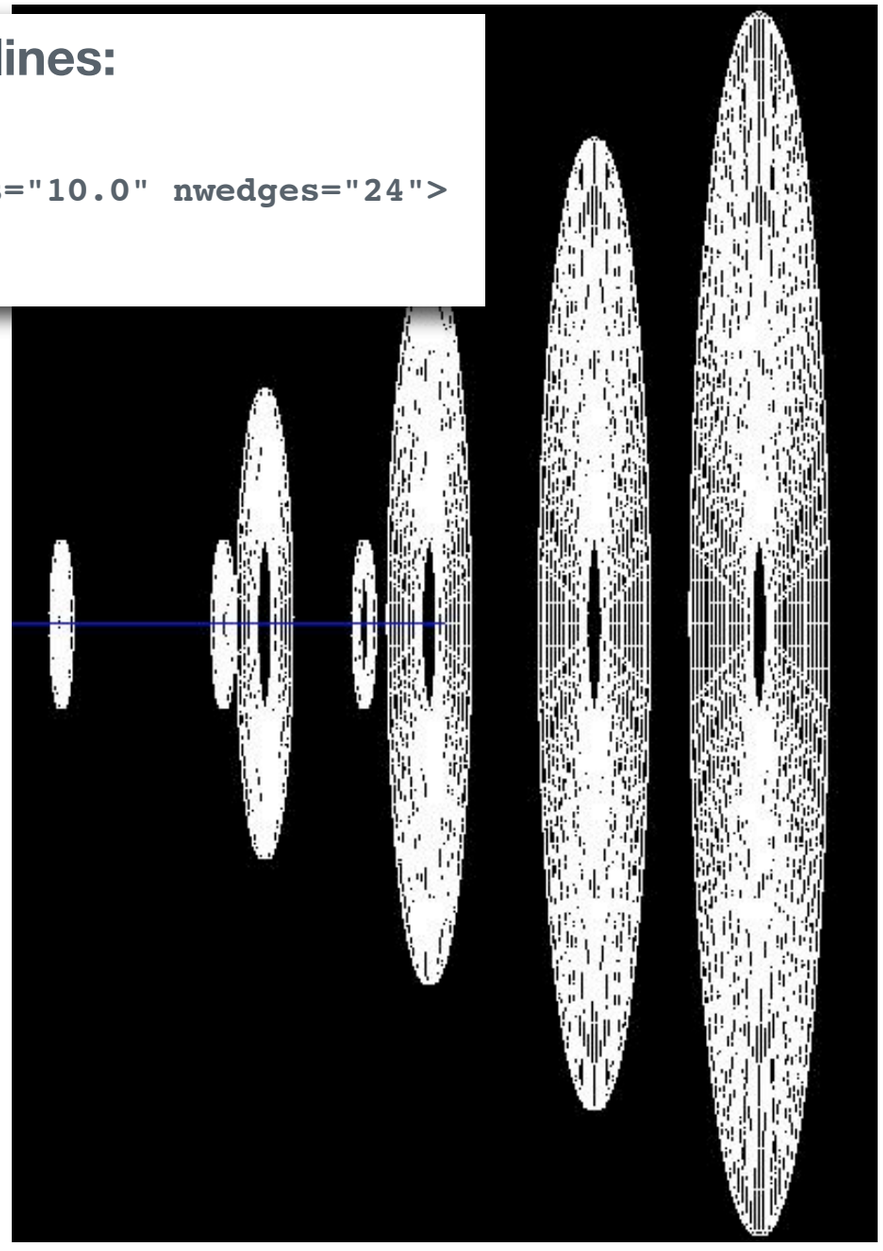


SiTrackerEndcap Example

Any wedge-type layout can be coded in a few, human-readable lines:

```
<!-- Endcap Tracker Layers -->
<layer id="4" inner_r="185.0" outer_r="478.0" inner_z="626.0" thickness="10.0" nwedges="24">
    <module_parameters r_size="85.333" phi_size_max="80" />
</layer>
```





Detector Descriptions

One representation for GEANT4 simulation,

- org.lcsim.geometry.compact.converter.lcdd.SiTrackerBarrel and SiTrackerEndcap convert the compact.xml to lcdd for SLIC (GEANT4)

and a second, consistent one for reconstruction, along with framework for managing the detector description: (inspired by Gaudi, ATLAS)

- org.lcsim.detector.converter.compact.SiTrackerBarrelConverter and .SiTrackerEndcapConverter convert compact.xml to objects in org.lcsim detector description and geometry framework
  - org.lcsim.detector - kernel of detector description and geometry framework
  - org.lcsim.detector.solids - geometry objects and operations
  - org.lcsim.detector.identifier - identification of elements

Silicon Simulation/Digitization

Description of the attributes of silicon sensors (in GeomConverter):

- org.lcsim.detector.tracker.silicon
  - SiSensor describes a generic silicon detector
    - DopedSilicon and BiasSurface define attributes determining charge generation/drift
  - SiStrips and SiPixels that implement SiSensorElectrodes on the bias surfaces

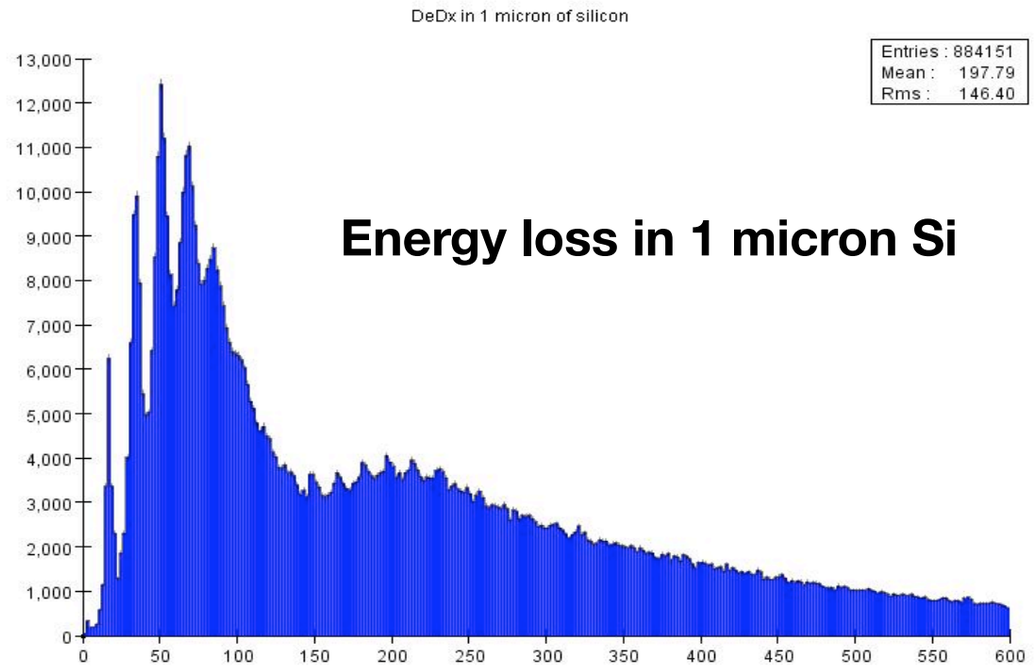
Charge deposition in sensors (in lcsim):

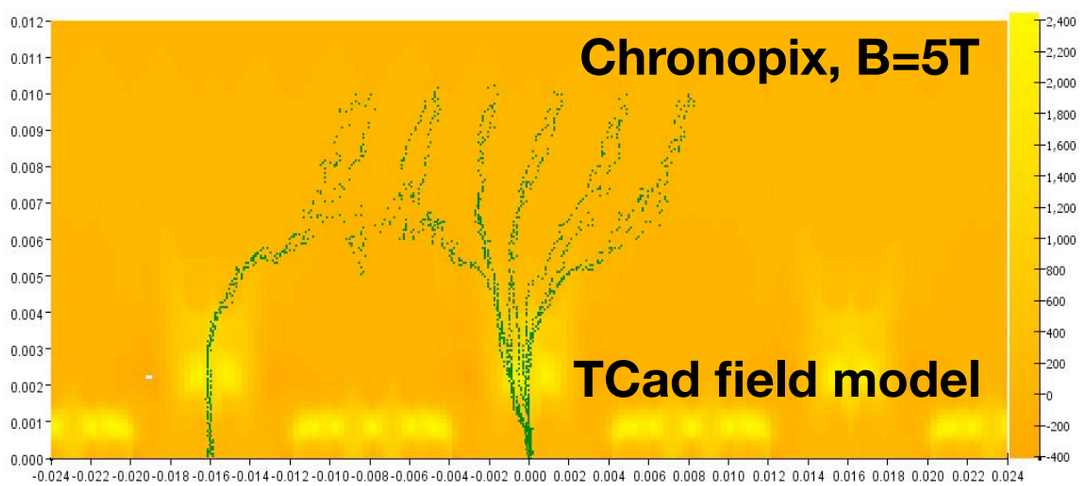
- org.lcsim.contrib.SiStripSim
  - CDFSiSensorSim implements SiSensorSim: improved CDF deposition model
  - fast, adequate for thicker silicon sensors of outer tracker

Detailed Pixel Simulation

CDF deposition model inaccurate for thin pixel sensors: new model by Nick Sinev

- Full implementation of shell effects
- Models arbitrarily complex pixel structures
- Drifts individual electrons: detailed results but very slow when there is region with no electric field (~1 second/hit)
- Produces a 3D lookup grid from detailed simulation of single pixel. Grid then used for fast simulation of physics events
- Integration of Nick's work into the hit reconstruction framework is a priority





Digitization and Readout

- ReadoutChip interface, outputs RawTrackerHits
- Implementation can be as generic or detailed as desired
- An example implementation of ReadoutChip, KPiX, is complete
  - register-level simulation of KPiX chip
  - includes encoding and decoding of ADC values (range bit, etc...)
  - *full, optimized noise simulation when attached to a StripClusterer: only generates noise hits that will result in clusters*

TrackerHit Class Heirarchy

- BaseTrackerHit (no-frills implementation of TrackerHit)
  - TransformableTrackerHit (+ability to transform coords)
    - SiTrackerHit (+references to sensors and electrodes)
      - SiTrackerHitStrip1D (+info about unmeasured coordinate)
      - SiTrackerHitStrip2D (+crossing-angle dependent position)
      - SiPixelHit

➡ *Nothing is lost when these types are persisted*

TrackerHitType

Encodes information about TrackerHit into an integer type that is persistable in LCIO, everything else one could want is regenerated on the fly after instantiation.

```
public class TrackerHitType
{
    public enum CoordinateSystem
    {
        GLOBAL,
        SENSOR,
        UNKNOWN;
    }

    public enum MeasurementType
    {
        STRIP_1D,
        STRIP_2D,
        PIXEL;
    }

    private static class Decoder
    {
        private static TrackerHitType decoded(int raw_type)
        private static int encoded(TrackerHitType type)
    }
}
```

Hit Reconstruction Framework

- class TrackerHitDriver extends Driver, makes use of four algorithm classes:
  - interface SiDigitizer (SimTrackerHits to RawTrackerHits)
  - interface StripClusterer (RawTrackerHits to SiStripHits1D)
  - interface PixelClusterer (RawTrackerHits to SiPixelHits)
  - interface StripHitCombiner (pairs of SiStripHits1D to SiStripHits2D)
- User can configure TrackerHitDriver to use any algorithm classes that implement these very minimal interfaces. A set of four default implementations are provided:
  - RawTrackerHitMaker implements SiDigitizer
  - StripHitMaker implements StripClusterer
  - PixelHitMaker implements PixelClusterer
  - StripHit2DMaker implements StripHitCombiner
- All hits are placed back on detector readouts *and* written to event header

TrackerHitDriver_User

- Example driver for configuring and running TrackerHitDriver
- Does default end-to-end digitization and hit reconstruction of any SimTrackerHits on SiSensors
- TrackerHitDriverUser_Test; runs on Z-pole→uds data
  - ~2.5 seconds/event on my laptop
  - ~15000 sensors, ~30×10^6 channels
  - ~2500 RawTrackerHits
  - ~200 SiTrackerHitStrip1D
  - ~40 SiTrackerHitStrip2D

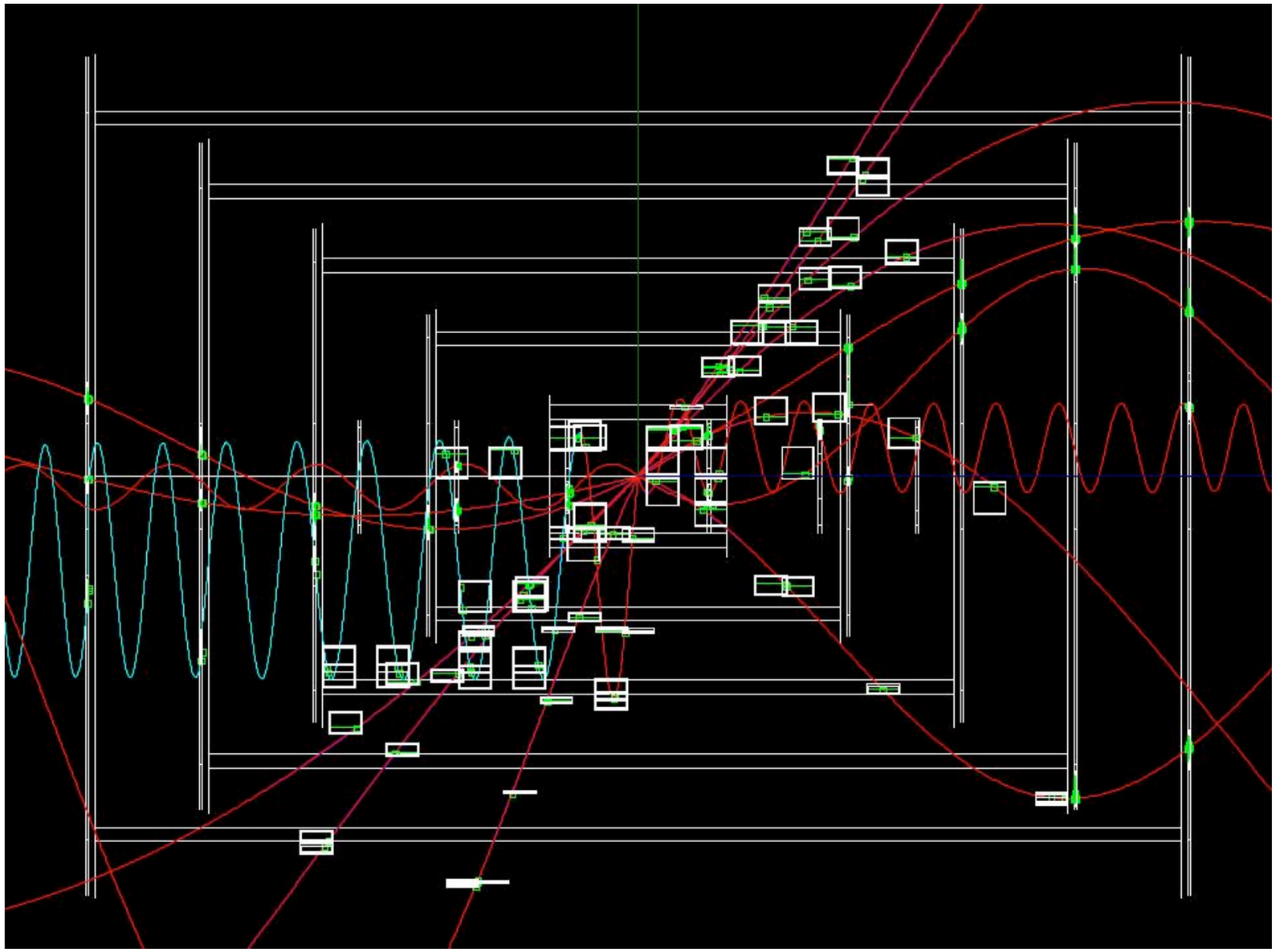
```
public class TrackerHitDriver_User extends Driver
{

    TrackerHitDriver _trackerhit_driver;

    /** Creates a new instance of TrackerHitDriver_User */
    public TrackerHitDriver_User()
    {
        _trackerhit_driver = new TrackerHitDriver();

        _trackerhit_driver.addReadout("SiTrackerBarrel_RO");
        _trackerhit_driver.addReadout("SiTrackerEndcap_RO");

        super.add( _trackerhit_driver );
    }
}
```

How Hard is It?

- Get/update code from cvs
  - If you don't already have the source distribution, see: **http://confluence.slac.stanford.edu/display/ilc/Building+org.lcsim+software**
  - If you already have it, then...
- **`cvs update GeomConverter`**
- **`cvs update lcsim`**
- **`GeomConverter/build.sh`**
- **`lcsim/build.sh`**
- download **http://www.lcsim.org/test/lcio/pythiaZPoleuds-0-1000_SLIC-v2r4p2_geant4-v9r1p0_LCPhys_SiTrackerTest01.slcio**
- open JAS3
- File -> New - Wired4 View
- File -> Load **`org.lcsim.contrib.SiStripSim.TrackerHitDriver_User`**
- File -> Open File -> **data file downloaded above**
- *Start looking at events with hits!*

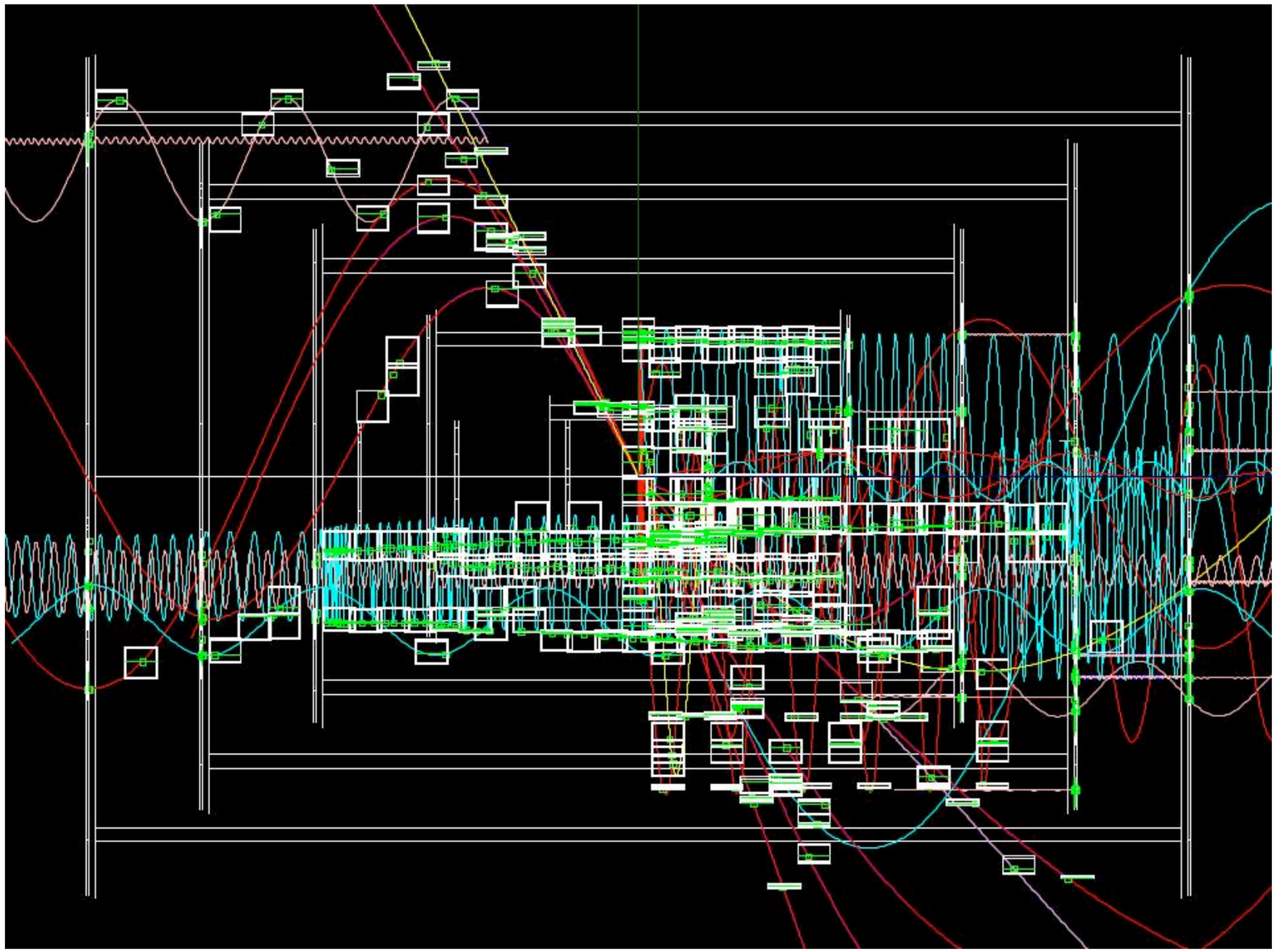
Looking at Events

- For clarity, event viewer shows only hit modules without any module substructure
- Strip hits are shown as the appropriate line segment
- Now possible to turn on all particles creating SimTrackerHits, color-coded by particle type



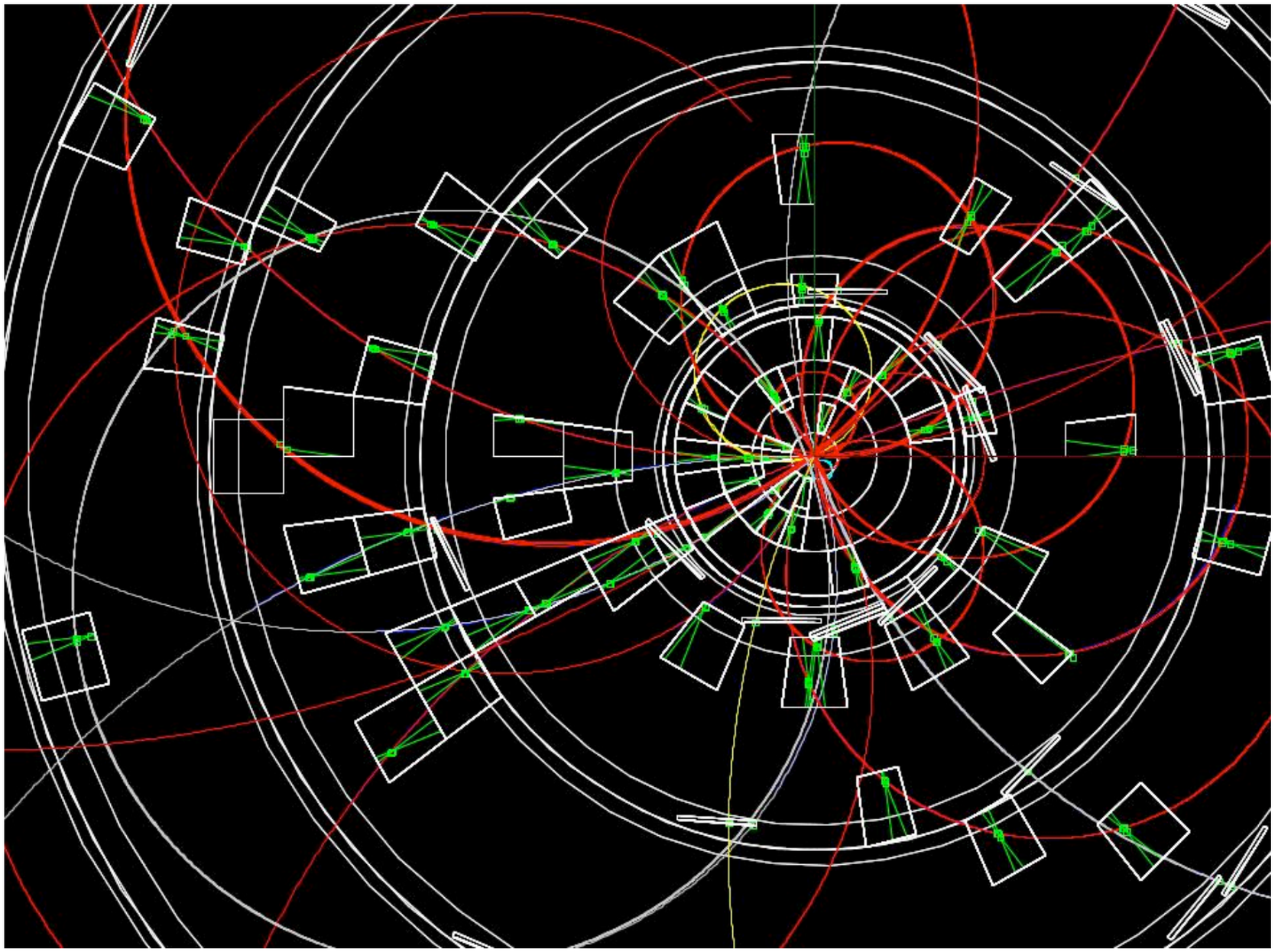
Looking at Events

- On rare occasions it can be a real zoo!
- This is clearly the worst of 100 events looked at
- Per-sensor occupancies in densest regions are still of order 1
- The blue are muons



Looking at Events

- Typical forward event showing SimTrackerHits and reconstructed stereo strip hits



Conclusions

- A complete and easy to use geometry, digitization and reconstruction architecture is in place for silicon tracking detectors in org.lcsim and ready **now** for simulation ***and*** reconstruction of tracking in fully detailed tracker models
- Improved tools for pixels in soon. Older models/tools can still be used, but as with tracking tools, the new tools are leaps and bounds better than the old in both fidelity and utility in performing tracking reconstruction.
- The to-do list is getting very, very short:
  - Updated detector model by ~May 1, followed by generation of data samples
  - Release of fully tested pixel reconstruction by ~May 15, pending attempts to integrate Nick's pixel simulation with the code
- Please try the code and help us improve it!
- The attention now must shift to track pattern-recognition and fitting...

A Quick Advertisement

- Nothing about hardware??
- Some new developments will be covered in the talk on K*PiX* in the calorimetry session tomorrow morning.