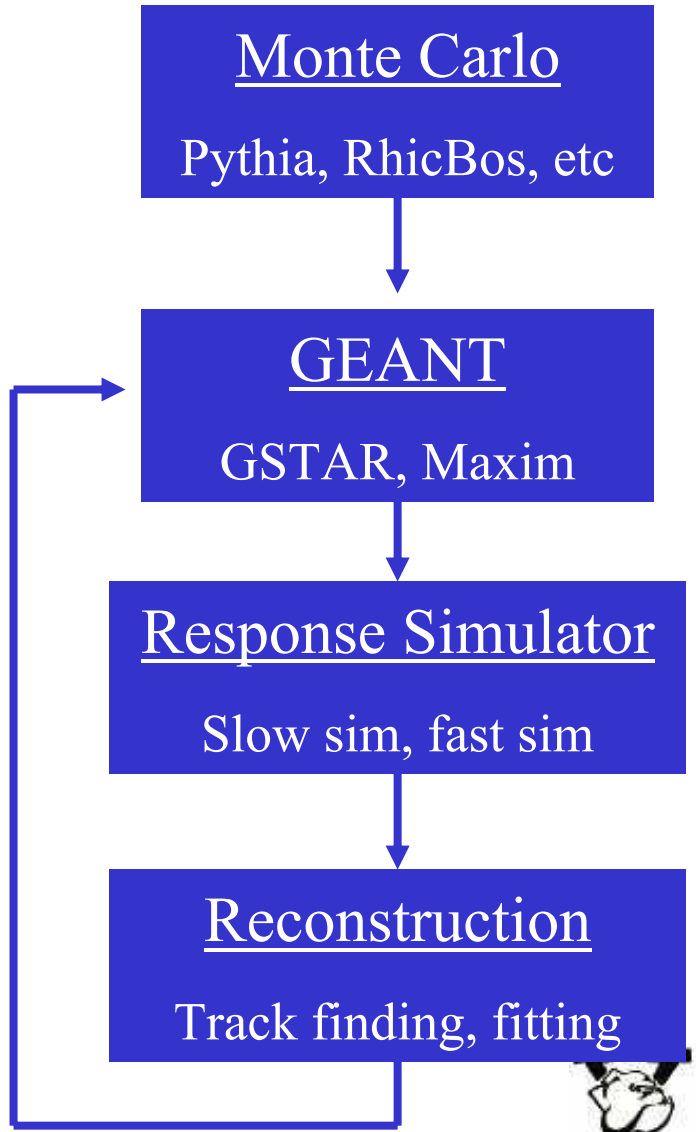
Forward Tracking: Software

1. New components: reconstruction and simulation
2. Current software components
3. Current status of ITTF tools

Why Do I Care About Software?

- R&D requires rapid feedback between hardware and end-physics
- Software environment:
  - Flexible, realistic, easy-to-use
- Event Simulation:
  - GEANT3
- Event Reconstruction:
  - Transition from Global chain to ITTF



Event Simulation

- New components in GSTAR (GeantSTAR)
  - Straightforward, fast
  - Realm of Maxim (1 day?)
  - Feedback from Pixel Group?
- Large scale simulation
  - Limited cpu and disk at BNL
  - Compete for resources with data production
  - PDSF is the place. cpu, and disk, disk … disk

Detector Simulators

- Straightforward task
- Fast simulator:
  - Smear hits by resolution
- Slow simulator:
  - Drift charges, amplify signals, add noise, etc…
- Silicon:
  - Plenty of expertise, experience
- Gem:
  - Nikolai, “under control”

Current Functioning Software

- Still TPC-centric:
  - TPT + global +primary, black-box Kalman
- FTPC:
  - Completely separate code
- SVT:
  - First physics production: RunII p+p (the 2nd)
- BEMC/EEMC:
  - No clear way to start tracking from EMC into TPC

Where's ITTF?

- ITTF in central region works well
- Not functionally integrated into bfc.C (reco. Chain)
- No clear path to production worthy tune
- But, flexibility demonstrated

- Fully integrated systems:
  - TPC, SVT, SSD, Pixel
- Partially integrated:
  - BEMC, FTPC
- Scattering planes:
  - Beampipe, IFC, SVT support, gas, …
- Can track Monte Carlo and data

How much ITTF do you need? It provides:

Sorted detector model, navigation/propagation methods, pattern recognition, easy integration … kalman filter

An Integration How To

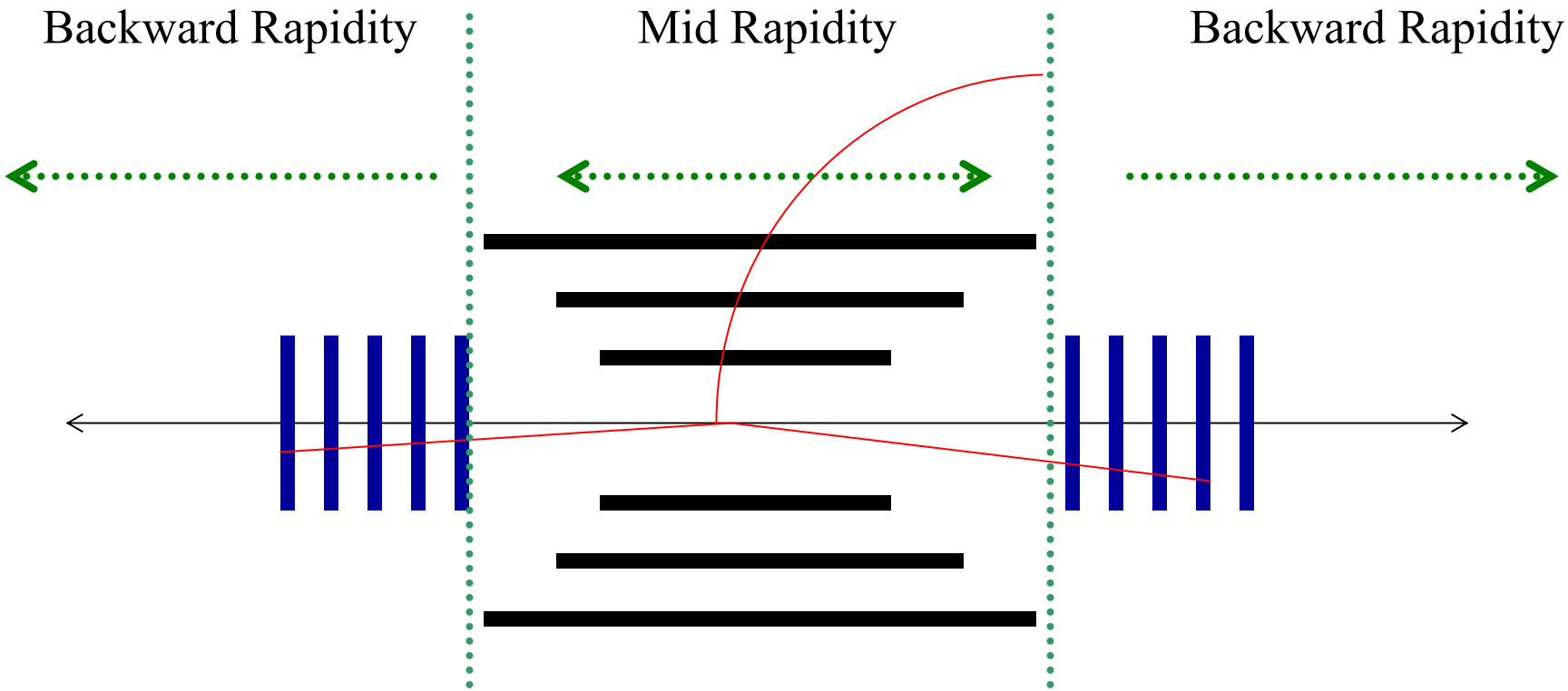
- What's expected:
  - Derive new directory StRoot/StXXXDetectorGroup
- What's for free:
  - Inherit from StiDetectorGroup
  - Base classes take care of c++ specifics
- Code you have to write:
  - Build shapes
  - Build detector objects
  - Load Hits

What Happens “Underneath”?

- StiXXXDetectorBuilder.cxx
  - Create c++ objects representing detector material and location.
  - Add these to internal container (handled by a call to base class add(StiDetector*)
  - StiDetectorTreeBuilder organizes these objects into a unique, extremely sorted tree structure.
  - This “sort” relies on:
    1. Region: kBackwardRapidity, kMidRapidity, kForwardRapidity
    2. Radial position
    3. Azimuth

“sort” happens only once/reco and defines “rules” of navigation

Why do we care about the sort?



Kalman Navigation Choices:

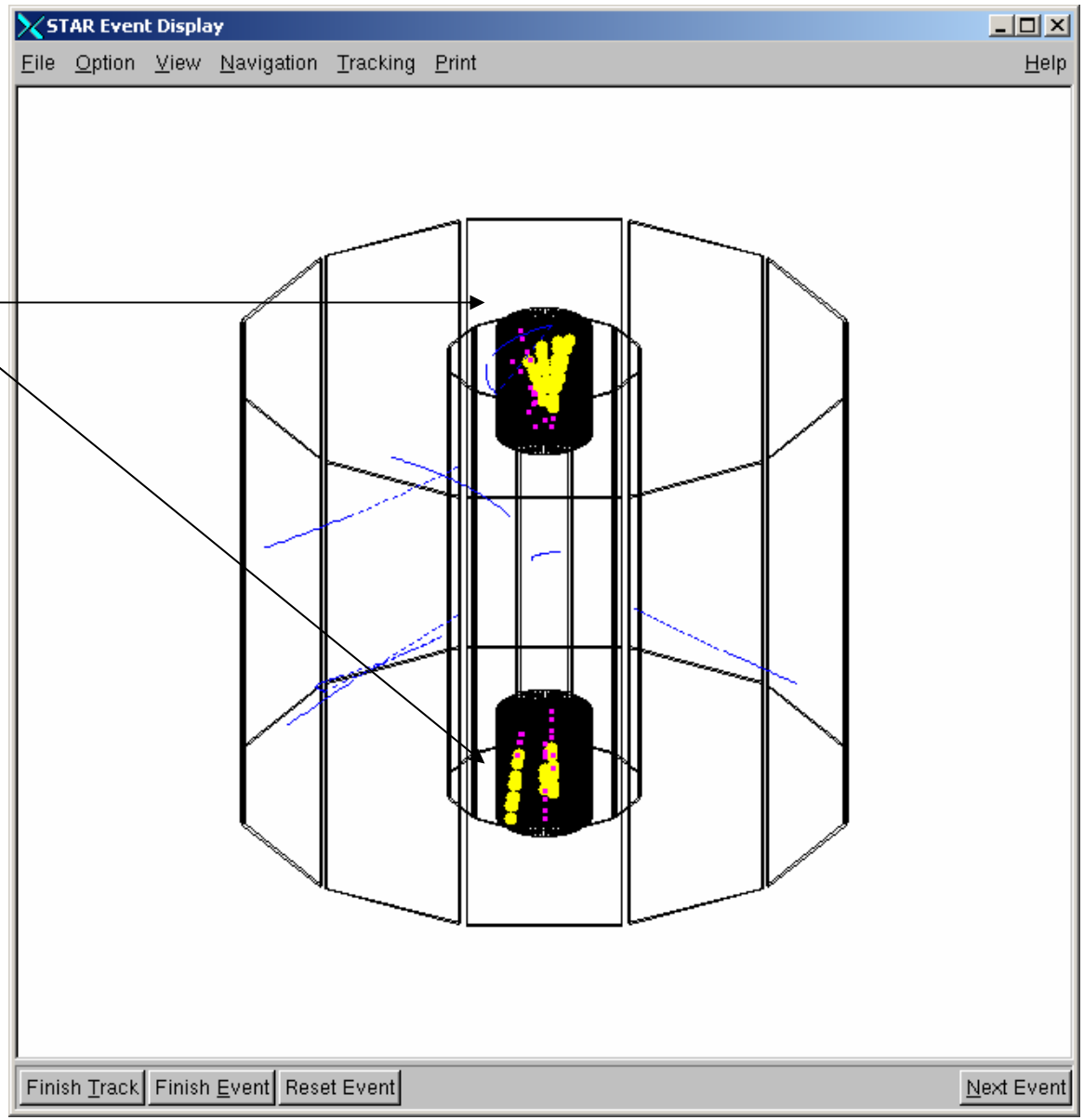
In/Out, +Phi/-Phi, +Region/-Region

FTPC: Proof Of Principle

FTPC Volumes

FTPC Hits

FTPC Hits Assigned to Track



Extrapolation/Projection Tools

- Projective tracking (outer point to vertex)
- Track matching to Pixel, Barrel, Endcap
- Central region:
  - Helix extrapolation straightforward, usable
  - Kalman extrapolation to planar surface debugged, usable
- Forward region:
  - Helix extrapolation still straightforward
  - Kalman extrapolation must be generalized, *Claude Pruneau!*

Conclusions

- Where to invest effort:
  - Nikolai, ITTF, some intermediate?
- New detectors in GEANT
  - Maxim adds, you can vary parameters
- Where to compute: PDSF
- Reconstruction:
  - Modifications needed in ITTF (Claude and friend)
  - Amount depends on specifics (*do we need Kalman?)*
- Realistic simulations needed
  - Efficiency, purity depend on multiplicity, background