# Particles and tracks from simulation

- Reminders:
  - The Geant4 detector-simulation module is LArG4.
  - LArG4 writes two objects per event:
    - ParticleList
      - Track ID
      - Particle kinematics
      - Trajectory
    - LArVoxelList
      - Energy in each voxel
      - A list of how much energy each particle contributed to the voxel
  - SimulationParameters.xml controls the simulation output
    - Minimum energy for a particle to be output (presently 0.1 MeV)
    - Minimum energy for a voxel to be output (presently 1 keV)
    - Voxel sizes (presently 0.3 mm cubes)

# Disk space

- ParticleList seemed to take up a lot of space.
- Suggestion from Brian Rebel: An option to turn off particle trajectory information.
- While implementing this, I discovered a bug in the trajectory code: it was adding a point for every voxel.

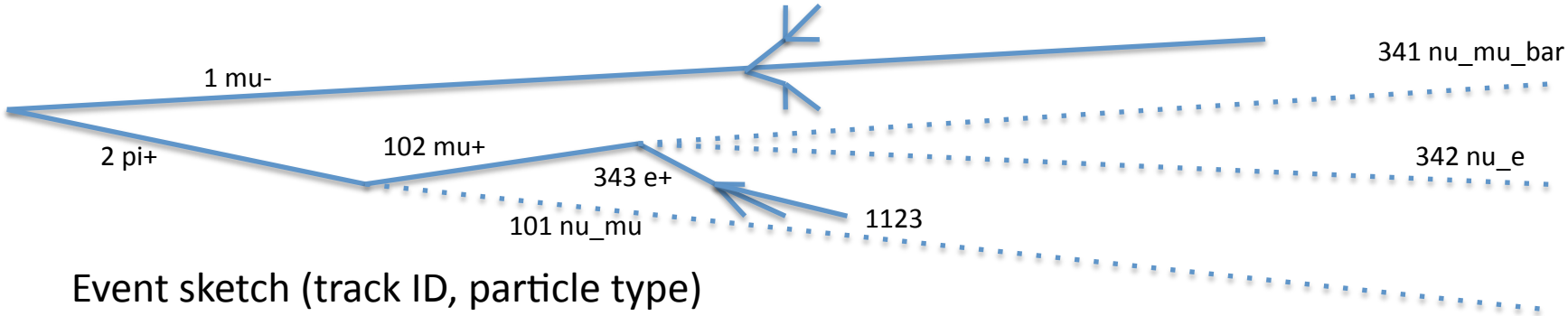
| prodgeniesim; 20 500-MeV $\nu_e$ | File size |
|---|---|
| With trajectories, before fix | 435MB |
| Without trajectories | 362MB |
| With trajectories, after fix | 369MB |

- Trajectory output controlled via SimulationParameters.xml
- Need studies to determine if we can change simulation parameters (voxel sizes, cuts) to reduce space without affecting reconstruction.

# Change in sim::Particle class

- ParticleList
  - Particle
    - Track ID
    - *Physics process that created the particle*
    - Particle kinematics
    - Trajectory
- "Process" is a text string
  - For primary particles (those created by EventGenerator), the value is "primary".
    - For EventGenerator process that created the primary, see simb::MCTruth
  - For other particles, this string comes from the Geant4 physics list.
- After this change, LArSoft won't be able to read ParticleList objects created in earlier files.

# ParticleHistory – a tool/container for understanding simulated showers

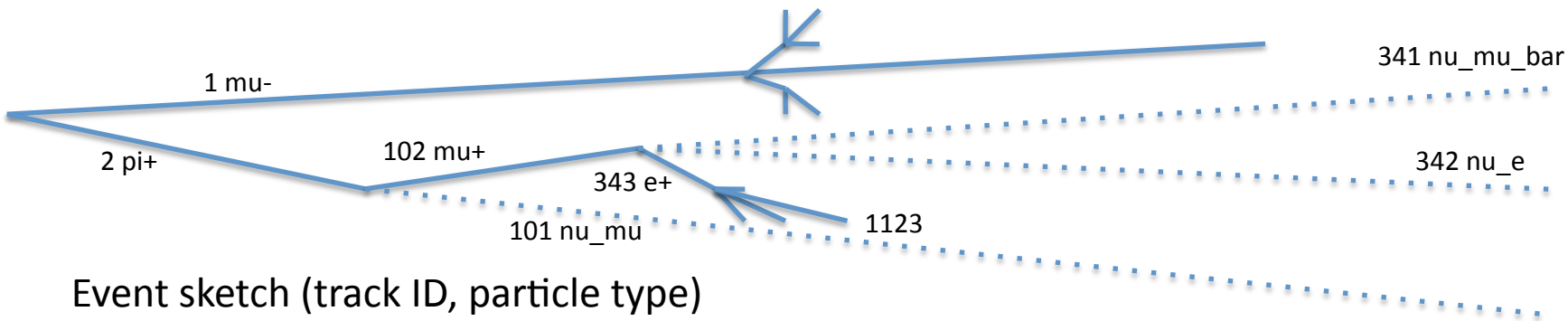



Event sketch (track ID, particle type)

```
sim::ParticleList* particleList = // ... from somewhere
int trackID = 343;
sim::ParticleHistory particleHistory( particleList, trackID );

// In the above example:
// particleHistory.size() == 3
// particleHistory[i] has the type const sim::Particle*
// particleHistory[0] points to the particle with track ID 2
// particleHistory[1] points to the particle with track ID 102
// particleHistory[2] points to the particle with track ID 343
```

# eve ID = “ultimate mother”




Event sketch (track ID, particle type)

- What is a useful eve ID definition for track 1123?
  - eve ID = track ID of primary particle = 2
  - eve ID = track ID of “significant” particle before e-m shower = 343
  - if 1123 is an e+, it might be the “same” positron => eve ID = 343? 2?
- Don’t debate!
  - Instead of creating a field in sim::Particle, use the strategy design pattern
  - I’ll supply a simple default algorithm
  - If you need your own definition, use your own method

# How it will work

```
sim::ParticleList* particleList = // ... from somewhere
int trackId = 343;
int eveId = particleList->EveId( trackId );



// If you need your own method to calculate the eve ID:
sim::ParticleList::SetEveCalculator( new myRoutine );



// Example myRoutine:
class myRoutine : public sim::EveIdCalculator {
public:
   myRoutine() {}
   ~myRoutine() {}
   int CalculateEveId( sim::ParticleList* list, int trackId )
   {
      sim::ParticleHistory particleHistory( list, trackID );
      // This just returns the track ID of the primary particle, but
      // you can loop over the history, make decisions based on
      // the process, etc.
      return particleHistory[0]->TrackId();
   }
};
```